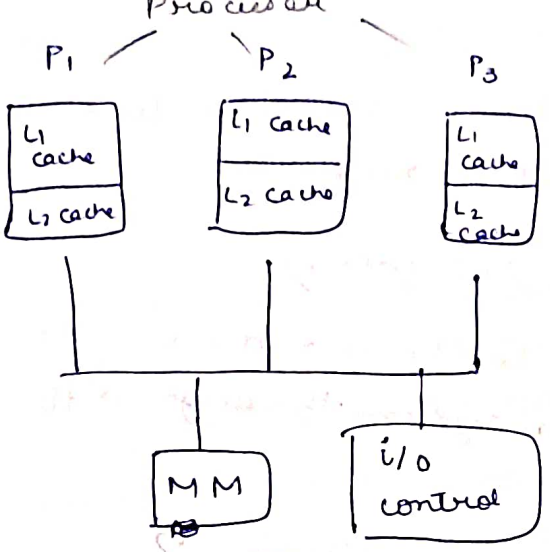
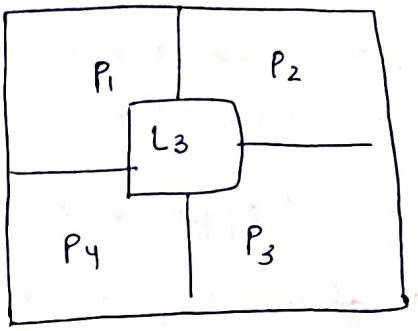
20/9/23



SMP



Multicore processor

CH 2 - Operating System Overview

Goals :

→ Convinience
→ Efficiency
→ Ability to evolve

How OS acts as user interface?

1. Program development provides services of editors & debugger
2. For the execution of program instr^{n} & data must be loaded. i/o devices & files must be initialized & other resources must be prepared.
3. Access to i/o devices
4. control access to files
   In a system with multiple users the OS must provide protection mechanism to control access to the files.
5. Error detection & respons : Os must provide a response that clears the error cond^{n} with the least impact on the running program.

- ISA (Instruction Set Architecture)

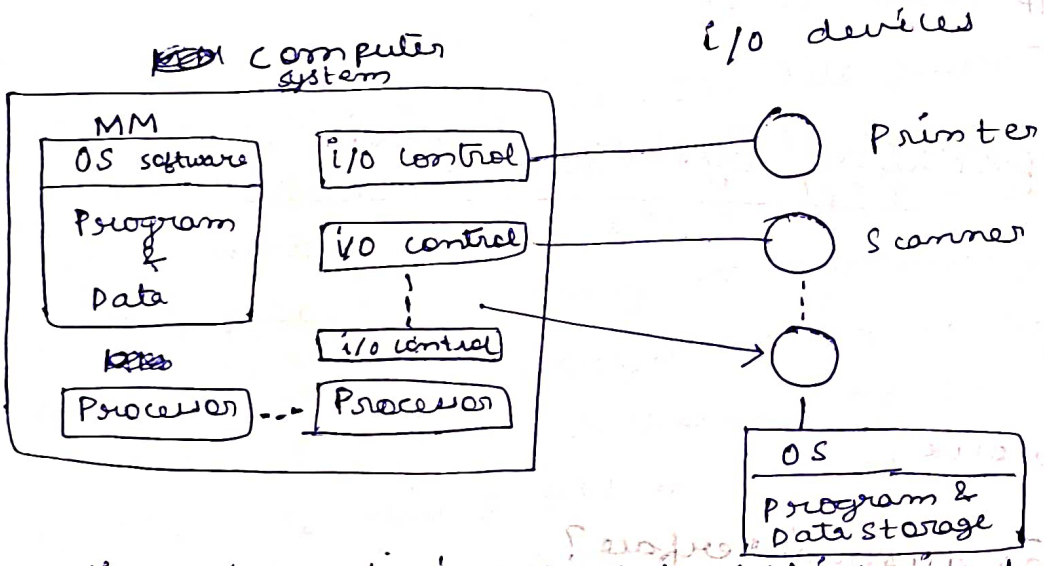
→ The OS has access to additional machine level language instruction that manage system resources

A -

→ API (Application program Interface.)

It gives a program access to the hardware resources & services available in a system through diff service calls

How OS acts as a resource manager ?



A portion of OS is in the MM which includes the kernel which contains most frequently used functions in the OS. The remainder of main memory contains user & utility programs & data. The OS & memory management hardware in the processor, jointly control the allocation of main memory. The OS decides when an i/o device can be used in a program in execution & control & access to use of files. The processor itself is a resource & OS determines how much processor time is to be given for the execution of any program

Types of OS

1) Batch OS :

In Batch OS the user has no direct access to the processor, instead the programs are submitted in form of cards / tapes to a computer operator who batches the prog. sequentially & places a batch on an input devices. Each program is constructed to branch back to the computer when it completes processing at which pt. the monitor automatically loads the next program. The programs are executed in system mode / kernel mode.

2) Multiprogramming OS

More than 1 processor are present in RAM. This is the non-primitive approach (one after another). The goal is to minimize CPU idle time.

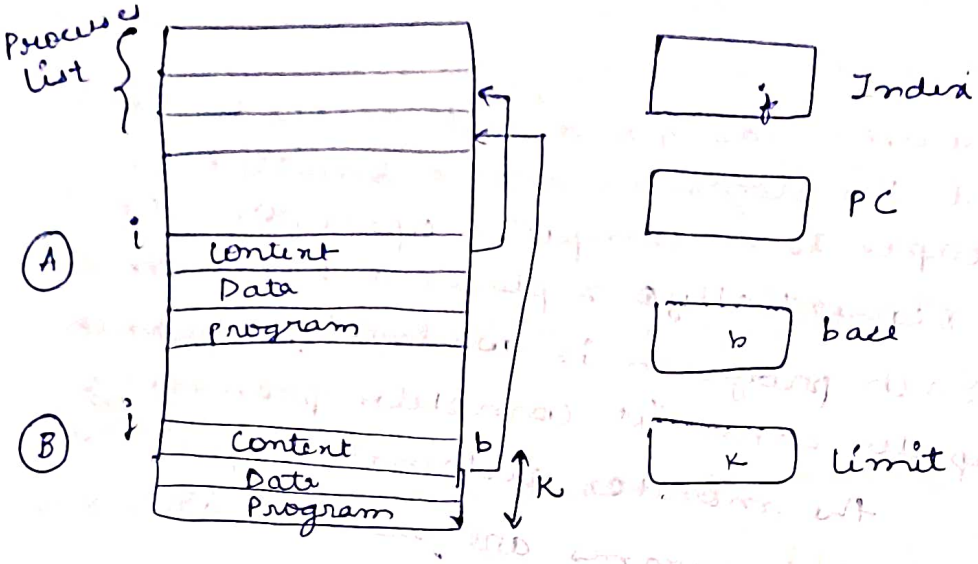
3) Multitasking OS

Similar to multiprogramming but premetive approach. The goal is to minimize response time.

Major achievements of any OS : advances in the development

1) Processes
2) Memory management
3) Information & security measures
4) Scheduling

1) Process



Def^n

A program in execution / an instance of a program currently running on the computer.

In the diagram 2 processes A & B exist in portions of main memory.

Each process is recorded in a process list, build & maintained by the OS. Generally there are 3 components of processes;

i) Executable program

ii) Associated Data

iii) Execution context of the program (State i.e. eg- waiting for i/o, etc)

The process list contains 1 entry for each process which includes a pointer to the location of the block of memory that contains the process.

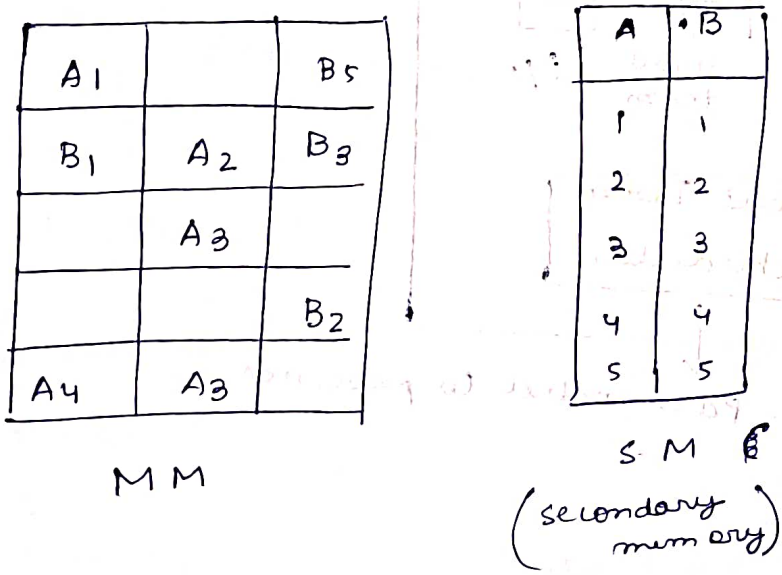
The process index register contains the index in the process list of the process currently controlling the processor.

The base & limit register define the region in memory occupied by the process.

The base register is the starting address of the region of memory & limit is the size of region

2) Memory management

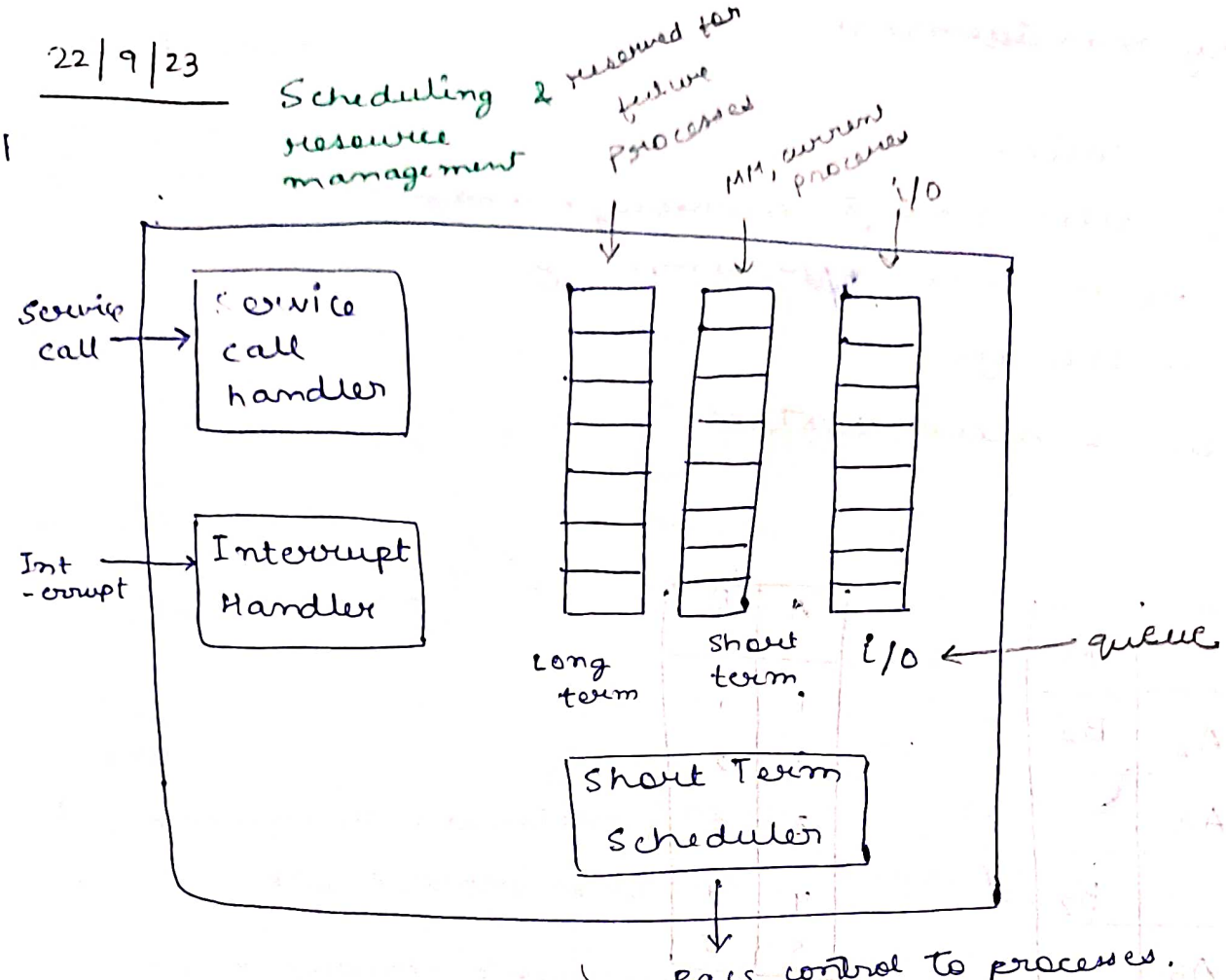
→ Process isolation
→ Automatic allocation & memory management
→ Support of modular programming
→ Long term storage
→ Protection & access control



Processes comprise of fixed size block called pages which are stored in secondary memory.
MM consists of no. of frames each equal to the size of a page. For a program to execute, some / all pages should be present in MM.

22/9/23

Scheduling & resource management



Information & security

1. Availability
2. Confidentiality
3. Data Integrity
4. Authentication

The OS maintains a no. of queues which contains a list of processes waiting for some resource

- The short term queue consists of process that are in MM & are ready to run as soon as processor is made available.
- The long term queue, is a list of new jobs waiting to use the processor. The OS adds the jobs to the system by transfering the process from the long term queue to short term queue. There is an I/O queue for each

i/o device ~~and~~. More than 1 process may request the use of same i/o device. The OS receives control of processor at the interrupt handler when an interrupt is raised / when a ~~so~~ service call is invoked from any application.

Development that lead to modern operating system

1) Microkernel Architecture
2) Multithreading interrupt
3) (SMP) Symmetric multiprocessing
4) Distributed environment
5) Object-oriented Design

Features of SMP

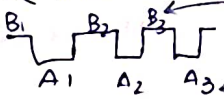
→ Performance
→ Incremental growth
→ Scaling
→ Availability, even after 1 processor shuts down.

Faults

Fault is the hardware/software state resulting from component failure, operator error, design error, data structure ~~f~~ or physical interference, ~~for~~ from the environment. Types → Permanent
→ Temporary

Fault tolerance

1) → ~~tolerance~~ Reliability ← uptime
2) → MTTF
3) → MTTR



$$MTTF = \frac{B_1 + B_2 + B_3}{3}$$

$$MTTR = \frac{A_1 + A_2 + A_3}{3}$$

→ Fault Tolerance refers to the availability of system to continue normal operation despite the presence of hardware & software faults

3 basic measures are

1) → Reliability

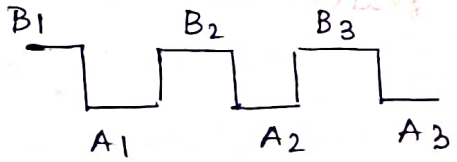
It is the probability of current operation at upto time t, given that the system was operating at time t = 0

2) MTTF : Mean time to failure

→ It is the average time available of the system when it was executing correctly

3) MTTR : " " " repair

→ The avg. time it takes to repair / replace a faulty element. The time during which system is not available is downtime while when sys. is available is uptime



$$MTTF = \frac{B1 + B2 + B3}{3} \qquad MTTR = \frac{A1 + A2 + A3}{3}$$

Design consideration for Multiprocessors

A multiprocessor OS must provide all the functionality of multiprogramming system some of the design considerations are :

1) Simultaneous concurrent process / threads
2) Scheduling
3) Synchronization
4) Memory management : It must be guaranteed that old data can no longer be accessed & needs to be updated is changed by any process

5) Reliability & Fault tolerance.

Multicore OS Considerations PP-99

26/9/23

(1) Parallelism :

Many applications can be subdivided into multiple tasks that can execute in ||^{el}. It is the duty of developer to decide how to split up the application work into independent executable tasks.

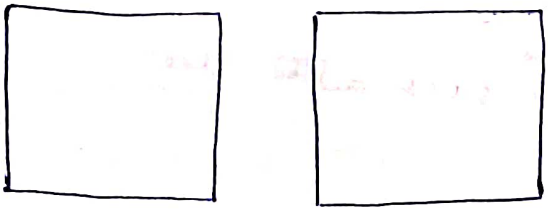
Grand Central Dispatch — GCD is a multicore support capability in which the OS maps tasks onto the threads.

(2) Virtual Machine Approach :

One / more cores are dedicated to a particular process then processor devotes all its efforts to that process. It is the duty of OS that assigns an application to a processor some memory. The program itself uses meta data generated by the compiler.

OS → windows
Linux

Windows Architecture



Kernel
1) Executive services
2) Microkernel
3) H/w abstraction layer
4) Device drivers
5) Windows Graphic Interface

User
1) special system process
2) Environment Subsystem
3) user Application

Kernel Mode

1. Executive Services

It makes the low level of kernel mode. There are diff. modules such as i/o manager, cache manager, object manager, security manager, power manager, virtual memory manager & process manager.

2. Micro Kernel

It is the heart of the kernel & it is central element of new technology. It's primary task is to schedule all active system threads while maintaining a procedure, operating environments & its highest performance level.

3. Hardware Abstraction Layer (HAL)

It maps b/w generate hardware commands & responses & those ~~who need~~ unique to a ~~special program~~ specific platform.

4. Device Drivers

These include hardware device drivers that, translate user i/o function calls into specific hardware device i/o request

5. Windows Graphic ~~sy~~ Interface

Implements the GUI ~~systems~~ functions such as User Interface Controls.

User Mode processes

1. Special System processes

They manage the system such as session management, authentication subsystem, service manager & log in process.

2. Environment ~~Sys~~ Subsystem

It includes the syetesk process that deals with responsiveness of application

3. User Application

Executables i.e. exe & DLL(s) that provide the functionality users runs to make use of systems.

Client / server Mode

A client which can be the application program / another server program request a service by sending the message. The message is routed through executive to the appropriate server.
Here server is a subsystem of process that provide services. The server performs the requested operation & returns the result / status information by means of another message which is routed through the executive back to the client.

Advantages

1. It simplifies the executive
2. Improves reliability
3. Provides a suitable base for distributed computing

Windows Objects

1. Encapsulation

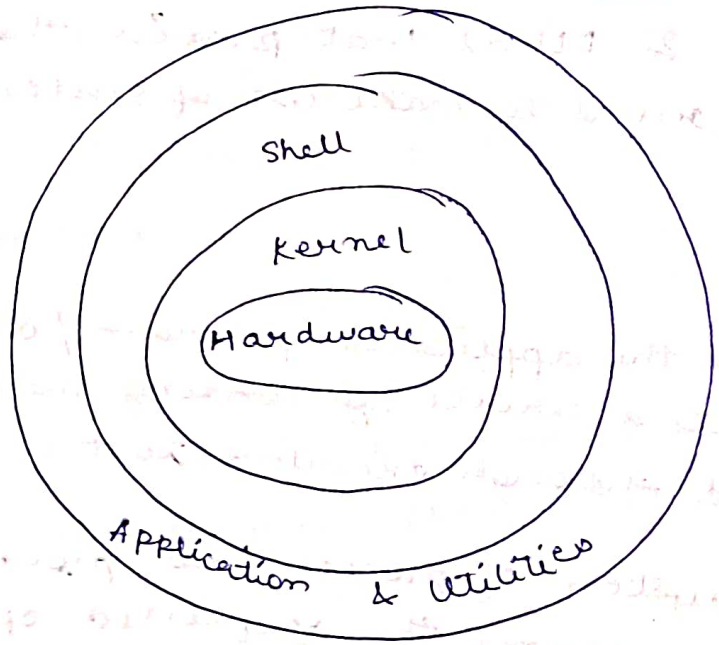
2. The only way to access the data in an object is by invoking one of the object services

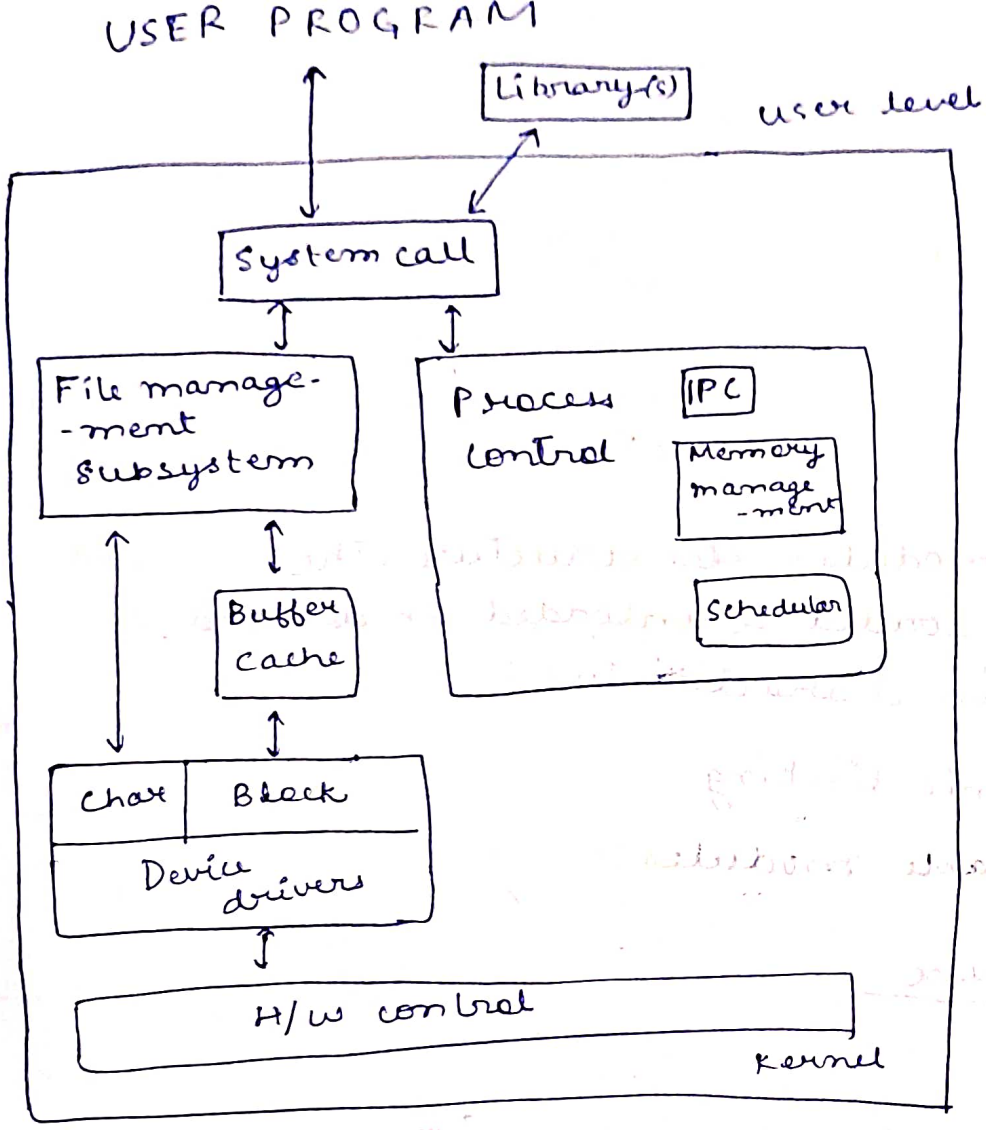
2. Inheritence

3. Object class & Instance

4. Polymorphism.

Unix ~~Linux~~ Architecture





User programs can invoke OS services either directly or through library programs. The system call interface is the boundary with the user & allows software to again access to specific kernel function. System has 2 main parts:

1. Process control
2. File management & i/O

1. The Process control subsystem is responsible for memory management, IPC & scheduling & dispatching of processes
2. The file system exchanges data b/w memory & external devices either as ~~from~~ stream of characters or in blocks. For this variety of device drivers are

used. like

i) SVR4
ii) BSD
iii) Solaris 11

Linux

Linux follows modular structure. The modules can be loaded & unloaded on demand. They have 2 main characteristics:

i) Dynamic linking
ii) Stackable modules

Components of Linux

1. Signals

The kernel uses signals to call into a process

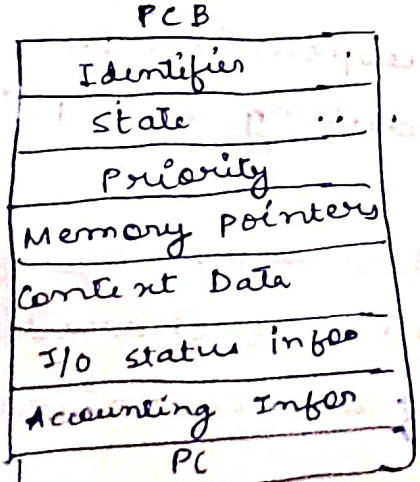
2. System call

It is the means by which a process requests a specific kernel service

3. Process & Scheduler

4. Virtual Memory

27/9/23 Process & Process control blocks



The process can be uniquely characterized by a no. of elements.

1. Identifier
→ Unique id associated with the process,

2. State of the process

3. Priority
   Priority level relative to other processes

4. Memory Pointers

→ Pointers do the program code & data associated

5. Context Data

→ Data that are present in registers while the process is executing

6. i/o status information

→ Includes outstanding i/o request, i/o devices assigned to this process. A list of file in use of this process.

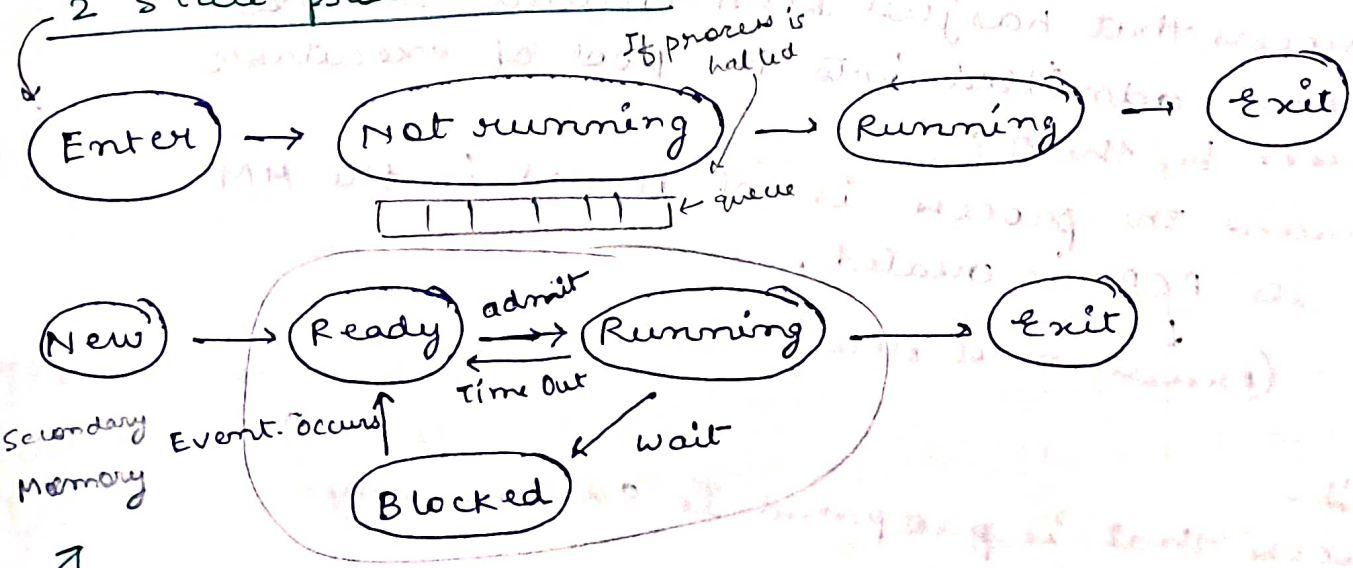
7. Accounting info:

→ Amount of processor time & clock time or time limits for this process

8. PC

→ Address of next instruction

2 State process model



5 state process model

→ When OS creates a new process, it creates a process control block for the process & enter that process into the system in the "not running state". The process exists & is waiting for its turn to execute. From time to time the currently running process will be interrupted & the scheduler will select some other process to run. The process in not running state is moved to running to state when it gets the turn.

→ All the new processes & the process that are interrupted are transfered to the queue of not running state, so we split the not running state to ready & block state.

Fig - 5 state model

New

A process that has just been created but has not been admitted into the pool of executable processes by the OS.
It means the process is not loaded in the MM but its PCB is created.
(Process control block)

Ready

A process that is prepared to execute when given the opportunity

Running state

The process that is currently being executed

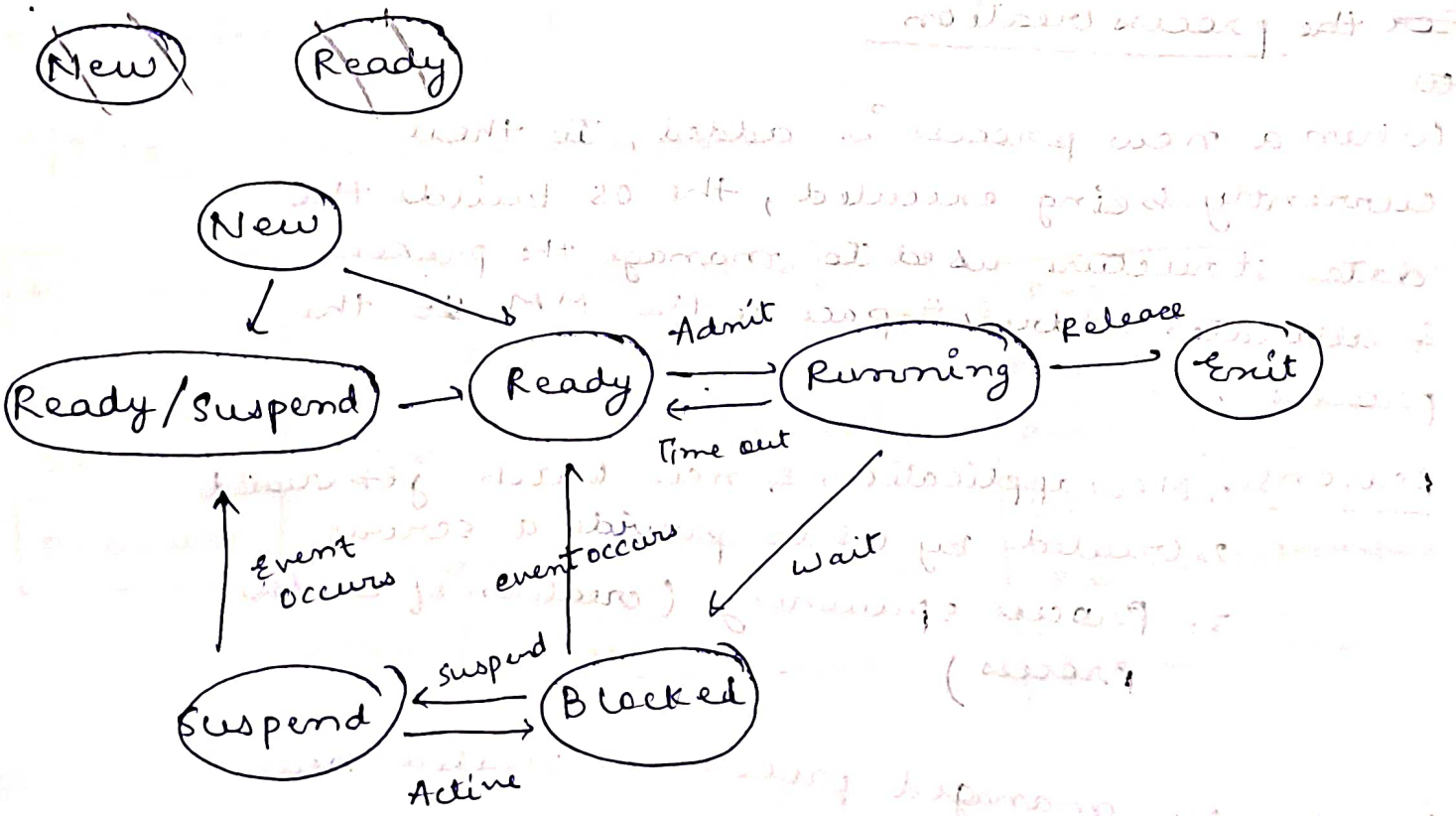
Blocked

A process that cannot execute until some event occurs such as completion of i/o

Exit

A process that has completed execution & has been released by the processor

* While a process is in new state, information regarding the process is maintained in control tables in MM, however the process itself is not in MM



The processor is much faster than i/o, that it will be common, for all the processes in memory waiting for i/o, hence processor would be idle for most of the time. When all the processes in MM are in blocked state the OS can suspend one process by putting it in suspended state & transfering it to secondary memory.

Ready

→ The process is in MM & ready for execution

Blocked

The process is in MM & awaiting an event

Suspend

The process is in SM & awaiting an event

Ready / Suspend

The process is in SM but is available for execution as soon as it is loaded in the MM

For the process creation

When a new process is added, To those currently being executed, the OS builds the data structure used To manage the process & allocates address space To the MM in the process.

Reasons
1. New application & new batch job created
2. Created by OS To provide a service
3. Process spawning (creation of child process)

A currently managed process can create a new process

4. Interactive log on.

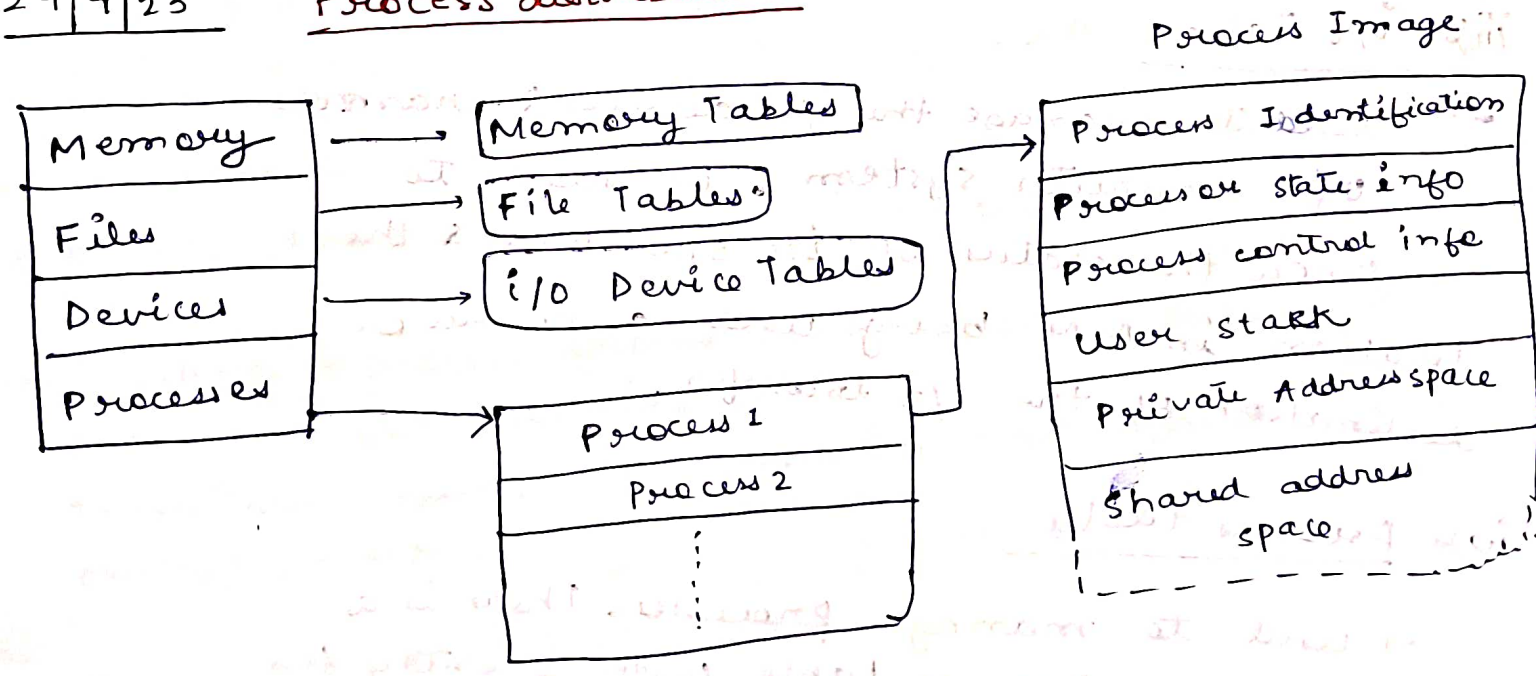
A user in terminal logs on To the system.

Reasons of process termination

1. Normal completion
2. Requires more memory than the system can provide
3. Bounce violation
   → The process tries to access a memory location that is not to allowed to access
4. Arithmetic error
5. i/o failure
6. Invalid instruction
   → The process attempts to execute a non-existent instruction
7. Parent process termination
8. Time limit exceeded

29|9|23

Process distribution



If OS wants to manage processes & resources, it must have information about the current status of each process & resource. The OS manages this information in 4 categories :-

i) Memory Tables

These are used to keep track of both main &

secondary memory.

The memory tables include the following information.

→ The allocation of MM to processes

→ " " of SM to "

→ Any information needed to manage virtual memory

→ Any protection attributes of blocks of MM.

ii) Files Tables

They provide information about the existence of file, their location on secondary memory & their current status.

iii) i/o Tables

→ used to manage the i/o devices & channels of computer system. Os needs to know the status of i/o operation & the location in MM being used as source or destination of the i/o transfer.

iv) process tables

→ used to manage processes. There is a primary process table with 1 entry for each process.

→ Each entry contains a pointer to a process image.

→ Process image consists of

* Process identification
  Unique process identifier

★ Processor state infor.

It consist of content of processor registers which contain the information of a running process in the register.

★ Process control information
This consists of different flag, such as resume flag, interrupt enable flag & others to control the process

★ Private & shared address space.

Scheduling Algo

Preemptive (As the name suggests, it provides starving of processes by giving equal time to each) equal time to each process

- SRTF (shortest remaining time first)
- LRTF (Longest " " ")
- ★ Priority Level based
- ★ Round-Robin

Non-preemptive (1 process enters processor, 2nd enters when 1 is complete)

- FCFS (First come first serve)
- SJF (shortest job first)
- LJF (Longest job first)
- Multilevel Queue
- HRRN (Higest response ratio next)

→ Selecting a process from READY queue & putting it on processor for running. There are multiple processes in Ready queue which is present in MM.

Times used in processor

1. Arrival Time (Process arrives in ready queue)
2. Burst Time (time to run the process inside process)
3. completion Time (burst time + i/o time + etc) = process end time
4. Turn around Time = Completion Time - Arrival Time
5. Waiting Time = TAT - Burst Time
6. Responsive Time = (time when process gets CPU (for the 1st time)) - (Arrival time)

1, Arrival Time

Time at which process enters the ready queue / state

2. Burst Time

Time required by a process to get executed on CPU (time duration

3. Completion Time

Time at which process completes execution

4. Turn around time

Completion Time - Arrival time

5. Waiting Time

Turn around time - Burst.

6. Response Time

First time when process get CPU - Arrival time

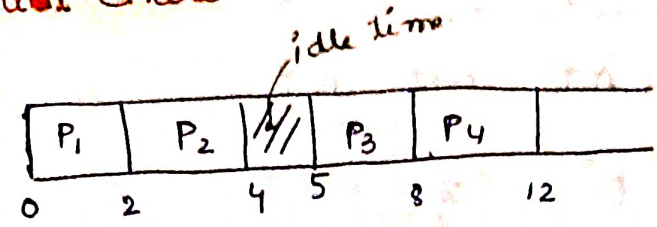
3/10/23 FCFS scheduling → non-premtive

P.no.	AT	BT	CT	TAT	WT	RT
P_1	0	2	2	2	0	0
P_2	1	2	4	3	1	1
P_3	5	3	8	3	0	0
P_4	6	4	12	6	2	2

given — calculated

★ In non premtive scheduling MWT = RT

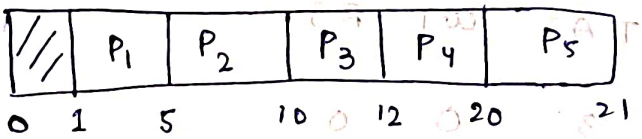
Gantt Chart



Average TAT $= \frac{2+3+3+6}{4} = \frac{14}{4} = 3.5$

Q(2)

P.NO	AT	BT	CT	TAT	WT	RT
P_1	1	4	5	4	0	0
P_2	3	5	10	7	2	2
P_3	4	2	12	8	6	6
P_4	7	8	20	13	5	5
P_5	8	1	21	13	12	12



Avg TAT $= \frac{4+7+8+13+13}{5} = \frac{19+26}{5}$

$= \frac{45}{5} = 9$

Avg WT $= \frac{0+2+6+5+12}{5} = \frac{25}{5} = 5$

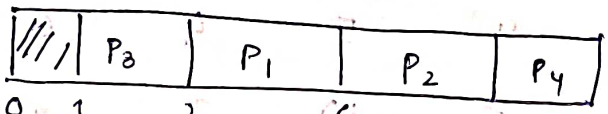
Non-premtive

~~Q③ P.No.~~ SJF (shortest job first)

Q①

P.NO.	AT	BT	CT	TAT	WT	RT
P_1	1	3	6	5	2	2
P_2	2	4	10	8	4	4
P_3	1	2	3	2	0	0
P_4	4	4	14	10	6	6

Gantt Chart :

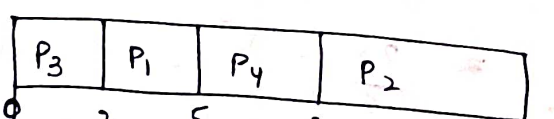


{P1, P3} {P1, P2} {P2, P4} {P4}

$$\text{Avg TAT} = \frac{5+8+2+10}{4} = \frac{25}{4} = 6.25$$

Q②

P.NO.	AT	BT	CT	TAT	WT	RT
P_1	2	3	5	3	0	0
P_2	0	4	12	12	8	8
P_3	0	2	2	2	0	0
P_4	3	3	8	5	2	2



{P3, P1} {P1, P4} {P4, P2} {P2}

$$\text{Avg CT} = \frac{5+12+2+8}{4} = \frac{27}{4}$$

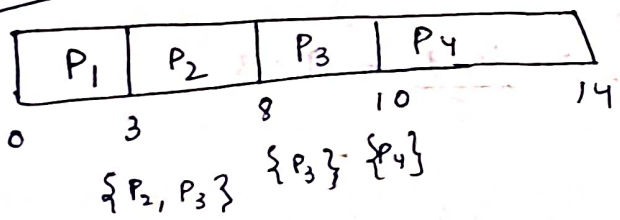
$$\text{Avg TAT} = \frac{3+12+2+5}{4} = \frac{22}{4} = 5.5$$

$$\text{Avg WT} = \frac{0+8+0+2}{4} = \frac{10}{4} = 2.5$$

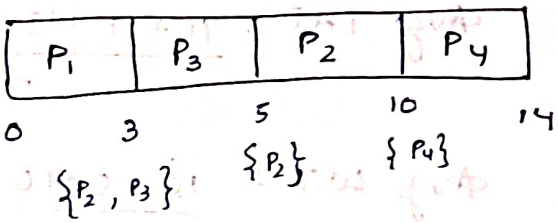
Q ③ Solve using SJF & FCFS

PNO	AT	BT	CT (FCFS)	CT (SJF)	TAT (FCFS)	TAT (SJF)	WT (FCFS)	WT (SJF)
P_1	0	3	3	3	3	3	0	0
P_2	1	5	8	10	7	9	2	4
P_3	3	2	10	5	7	2	5	0
P_4	9	4	14	14	5	5	1	1

FCFS



SJF



Avg TAT $= \frac{3+7+7+5}{4}$

$= \frac{22}{4} = 5.5$

Avg TAT $= \frac{3+9+2+5}{4}$

$= \frac{19}{4} = 4.75$

Avg WT $= \frac{0+2+5+1}{4}$

$= \frac{8}{4} = 2$

Avg WT $= \frac{0+4+0+1}{4}$

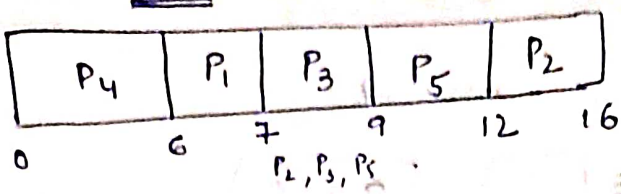
$= \frac{5}{4} = 1.25$

Q ④ Solve using SJF & FCFS

P. No.	AT	BT	CT (FCFS)	CT (SJF)	TAT (FCFS)	TAT (SJF)	WT (FCFS)	WT (SJF)
P_1	3	1	14	7	11	4	10	3
P_2	1	4	10	16	9	15	5	11
P_3	4	2	16	9	12	5	10	3
P_4	0	6	6	6	6	6	0	0
P_5	2	3	13	12	11	10	8	7

Ans

SJF



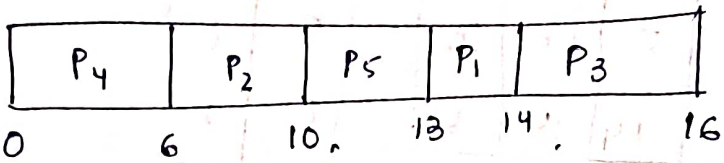
P_2, P_3, P_5

P_4 $\{P_1, P_2, P_3, P_5\}$

Avg TAT = $\frac{4+15+5+8+8}{5} = \frac{40}{5} = 8$

Avg WT = $\frac{3+11+3+0+7}{5} = \frac{24}{5} = 4.8$

FCFS



P_4 P_2, P_5, P_1, P_3

Avg TAT = $\frac{11+9+12+6+11}{5} = \frac{49}{5} = 9.8$

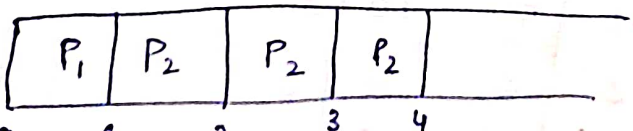
Avg WT = $\frac{10+5+10+0+8}{5} = \frac{33}{5} = 6.6$

4|10|23 SRTF (Shortest remaining Time first) → Premtive

Q① ~~P.No~~ It is a premtive SJF that will preamt the currently executing process where as non-premtive SJF will allow currently running proces ~~with~~ to complete CPU time / CPU burst.

Q①

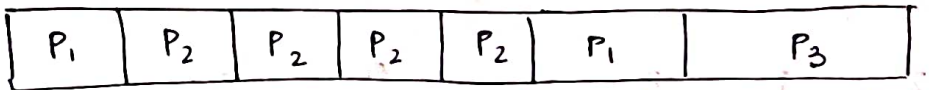
P.No.	AT	BT	CT	TAT	WT	RT
P_1	0	5 4				
P_2	1	3 2				
P_3	2	4				
P_4	4	1				



P_1 P_1, P_2 $P_1 P_2 P_3$

Q ②

P. NO.	AT	BT	CT	TAT	WT	RT
P_1	0	9	13	13	4	0
P_2	1	4	5	4	0	0
P_3	2	9	22	20	11	11



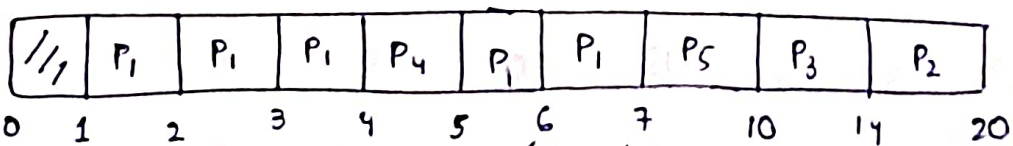
P_1 P_1P_2 $P_1P_2P_3$ P_1P_3 P_3

$$\text{avg TAT} = \frac{13+4+20}{3} = \frac{37}{3}$$

$$\text{avg WT} = \frac{4+0+11}{3} = \frac{15}{3} = 5$$

Q ③

P. NO	AT	BT	CT	TAT	WT	RT
P_1	1	5	7	6	1	0
P_2	2	6	20	18	12	12
P_3	2	4	14	12	8	8
P_4	4	1	5	1	0	0
P_5	6	3	10	4	1	1



{P1} {P1 P2 P3} {P1 P2 P3 P4} {P1 P2 P3} {P1 P2 P3 P5} {P2 P3} {P2 P3}

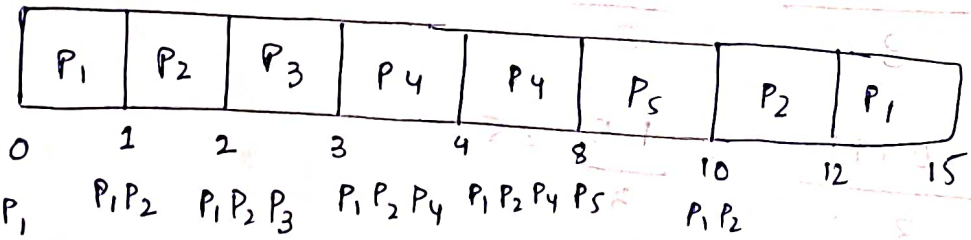
$$\text{Avg TAT} = \frac{41}{5}$$

$$\text{Avg WT} = \frac{22}{5}$$

Priority based Scheduling

* Higher no, higher priority (in this ques)

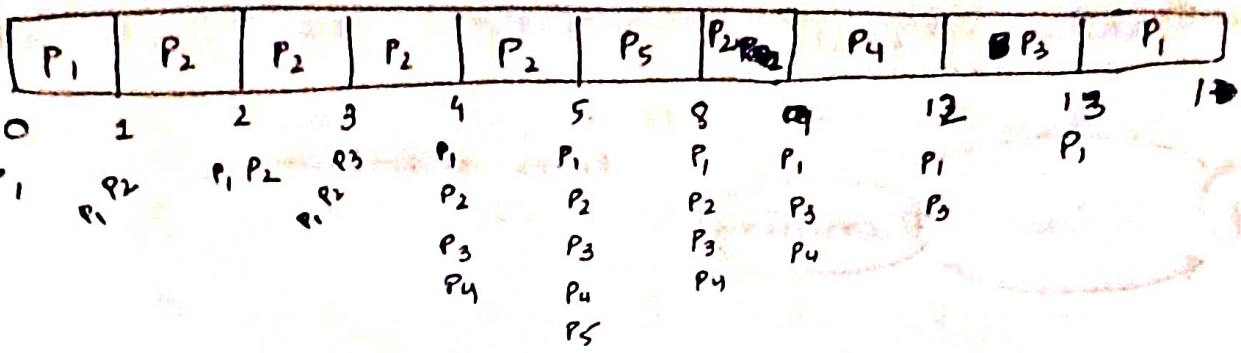
Priority	P.No.	AT	BT	CT	TAT	WT	RT
2	P_1	0	3	15	15	12	0
3	P_2	1	2	12	11	8	0
4	P_3	2	1	3	1	0	0
5	P_4	3	4	8	5	0	0
5	P_5	4	2	10	6	4	4



In this technique, a priority is associated to the process & CPU is allocated to the procus with highest priority. Equal priority processes are scheduled in FCFS.

Q ②

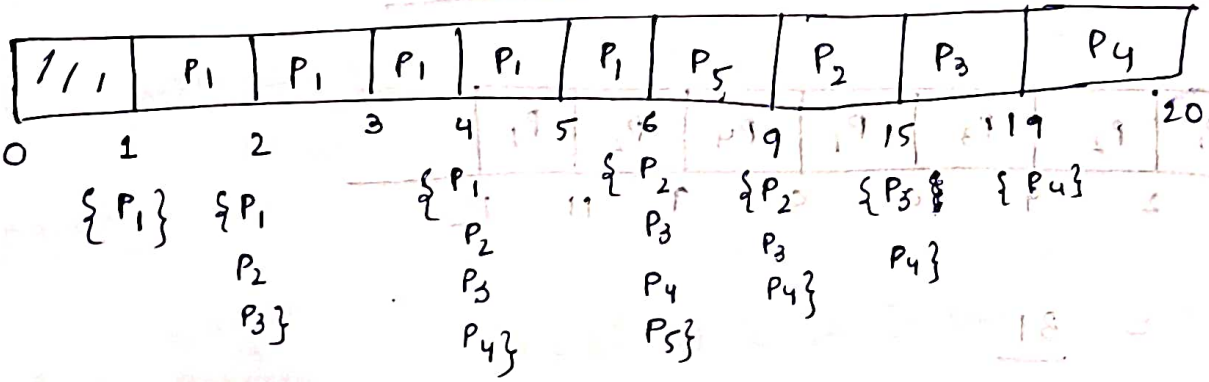
Priority	P.No.	AT	BT	CT	TAT	WT	RT
1	P_1	0	3	16	16	12	0
4	P_2	1	2	9	8	3	0
2	P_3	3	1	13	10	9	9
3	P_4	4	3	12	8	5	5
5	P_5	5	3	8	3	0	0



$$\text{Avg TAT} = \frac{16 + 9 + 10 + 8 + 3}{5} = \frac{45}{5} = 9$$

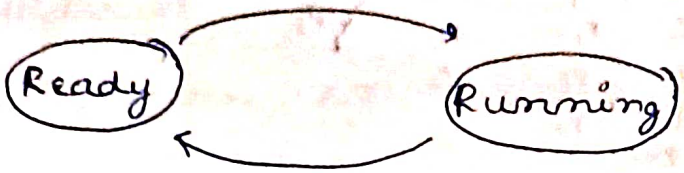
Q③ Priority

PID/ P.NO.	AT	BT	CT	TAT	WT	RT
P1 (4)	1	5	6	5	0	0
P2 (2)	2	6	15	13	7	7
P3 (2)	2	4	19	17	13	13
P4 (1)	4	1	20	16	15	15
P5 (3)	6	3	9	3	0	0



$$\text{Avg TAT} = \frac{54}{5} \qquad \text{Avg WT} = \frac{35}{5} = 7$$

Round-Robin scheduling

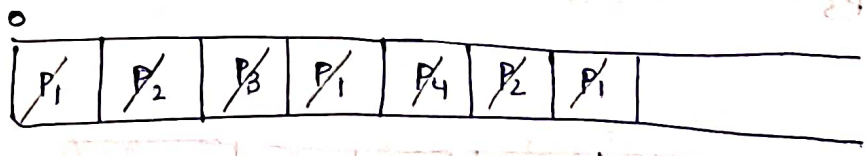


time quantum = q

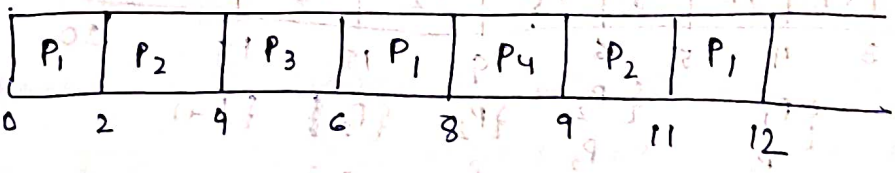
PID	AT	BT	CT	TAT	WT	RT
P_1	0	5 ~~8~~ 1	12	12	7	0
P_2	1	~~4~~ 2	11	10	6	1
P_3	2	~~2~~ 0	6	4	2	2
P_4	4	1	9	5	4	4

q = 2 (each process will run for 2 units)

Ready



Gantt chart



Avg TAT = $\frac{31}{4}$

Avg WT = $\frac{19}{4}$

②

PID	AT	BT	CT	TAT	WT	RT
P_1	0	~~5 2 10~~	13	13	8	0
P_2	1	~~8 1 0~~	12	11	8	1
P_3	2	1 0	5	3	2	2
P_4	3	2 0	9	6	4	4
P_5	4	~~3 1 0~~	14	10	7	5

$q = 2$

Ready | ~~P_1~~ | ~~P_2~~ | ~~P_3~~ | ~~P_1~~ | ~~P_4~~ | ~~P_5~~ | ~~P_2~~ | ~~P_1~~ | P_5 |

Gantt Chart

P_1	P_2	P_3	P_1	P_4	P_5	P_2	P_1	P_5

0 2 4 5 7 9 11 12 13 14

Avg TAT $= \frac{43}{5}$

Avg RT $= \frac{12}{5}$

Avg WT $= \frac{29}{5}$

~~HRRN~~

A time quantum 'q' is maintained & it is the criteria for choosing the next process.

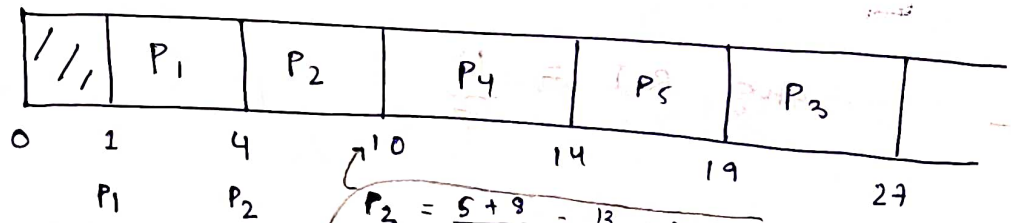
It's mode is preemptive i.e. after the time quantum the process will be context switched

HRRN (Higher response ratio next)

Mode - Non-premtive

$$\text{Response ratio} = \frac{WT + BT}{BT}$$

PID	AT	BT	CT	TAT	WT	RT
P_1	1	3	4			
P_2	3	6	10			
P_3	5	8	27			
P_4	7	4	14			
P_5	9	5	19			



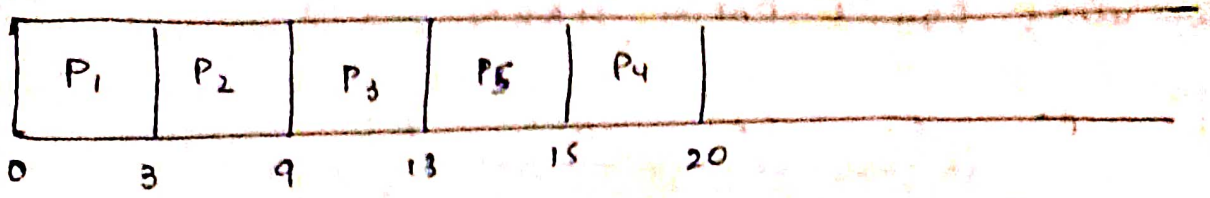
$P_3 = \frac{5+8}{8} = \frac{13}{8} = 1.6$

$P_4 = \frac{3+4}{4} = \frac{7}{4} = 1.7$

$P_5 = \frac{2+5}{5} = \frac{7}{5} = 1.4$

Q ②

PID	AT	BT	CT	TAT	WT	RT
P_1	0	3	3	3	0	
P_2	2	6	9	7		
P_3	4	4	13	9		
P_4	6	5	20	14		
P_5	8	2	15	7		



9 : $P_3 = \frac{5+4}{4} = \frac{9}{4} = 2.25$

$P_4 = \frac{3+5}{5} = \frac{8}{5} = 1.6$

$P_5 = \frac{1+2}{2} = \frac{3}{2} = 1.5$

13 : $P_4 \rightarrow \frac{7+5}{5} = \frac{12}{5} = 2.$

$P_5 \rightarrow \frac{5+2}{2} = \frac{7}{2} = 3.$

Key pts.

- Highest response ratio next
- In this algorithm, we consider the response ratio on

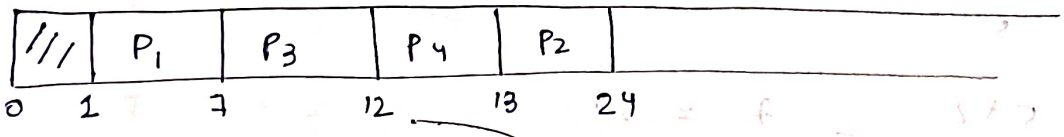
$$= \frac{WT + BT}{BT}$$

- using SJF algorithm, we face the problem of starvation, i.e. the process with longer time do not get their chance to get executed on the CPU, Hence in HRRN, we consider WT as well as BT.
- Its mode is non-preemtive in nature

HW Solve using Round-Robin, HRRN. q=3

PID	AT	BT	CT		TAT		WT		RT	
			RR	HPRN	RR	HRRN	RR	HPRN	RR	HRRN
P_1	1	6 3	13	7	12	6		0		0
P_2	2	11 4	23	24	21	22		10		11
P_3	4	5 2	19	12	15	8		3		3
P_4	7	1	14	13	7	6		5		5

(1) HRRN:



P_1

$P_2 = \frac{5+11}{11} = 1.4$

$P_3 = \frac{3+5}{5} = 1.6$

$P_4 = \frac{0+1}{1} = 1$

$P_2 = \frac{10+11}{11} = 1.$

$P_4 = \frac{5+1}{1} = 5$

Avg TAT $= \frac{42}{4} = 10.5$

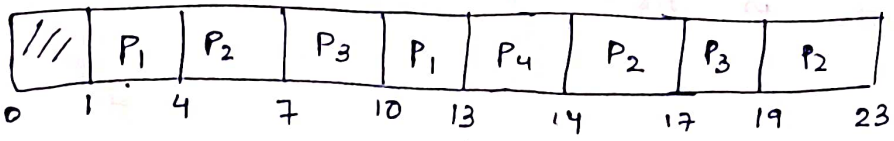
Avg WT $= \frac{19}{4} = 4.5$

Avg RT =

(2) Round-Robin:

Ready queue: ~~P_1~~ ~~P_2~~ ~~P_3~~ ~~P_1~~ ~~P_4~~ ~~P_2~~ ~~P_3~~ P_2

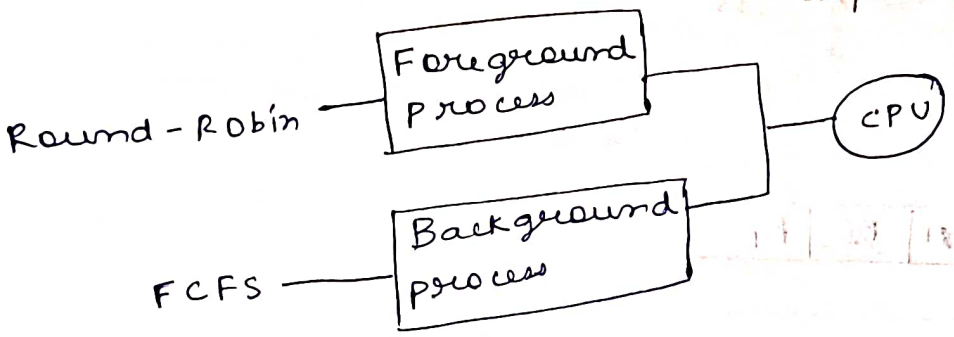
Gantt chart:



Avg TAT $= \frac{55}{4} =$

10/10/23 **Multilevel queue scheduling**

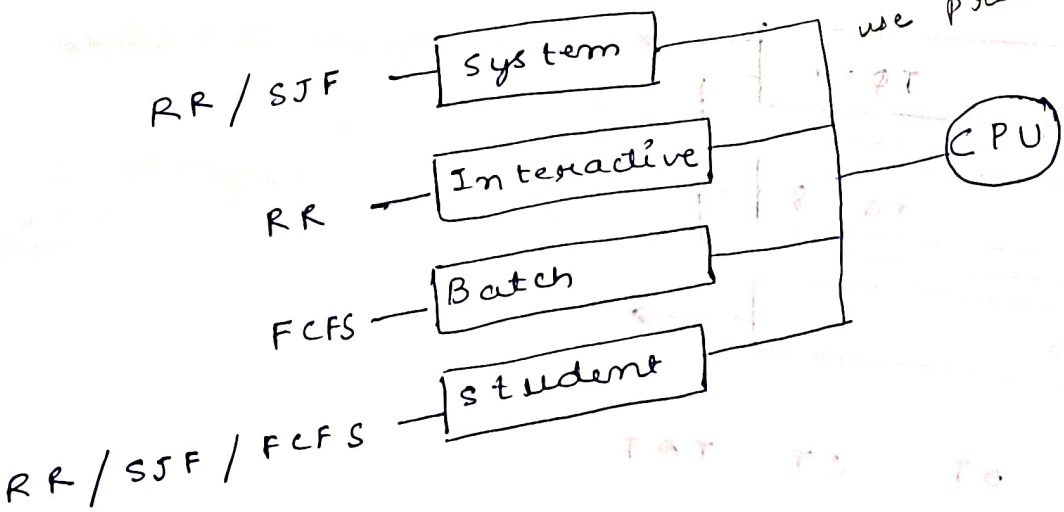
1. Foreground - Interactive process (Like applications, you interact with the application)
2. Background - Batch process (all commands are provided at once, CPU takes time to process it)



Priority wise processes

1. System process
2. Interactive process
3. Batch process
4. student process → notepad, etc.

More priority processes tend to use preemptive scheduling



Def^n Only after completion of all processes from from top level queue, the bottom level ready queue would be executed.

P.No.	Q.No. (queue no)	AT	BT	CT	TAT	WT
P_1	1	0	~~4~~	~~6~~		
P_2	1	0	~~3~~	7		
P_3	2	0	~~8~~ 5	20		
P_4	1	10	5	15		

q = 2 time units

* starvation disadvantage is faced

RR, q_1 | ~~P_1~~ | ~~P_2~~ | ~~P_3~~ | ~~P_2~~ | P_4 |

FCFS, q_2 | ~~P_3~~ | P_3 |

Gantt chart

P_1	P_2	P_1	P_2	P_3	P_4	P_3

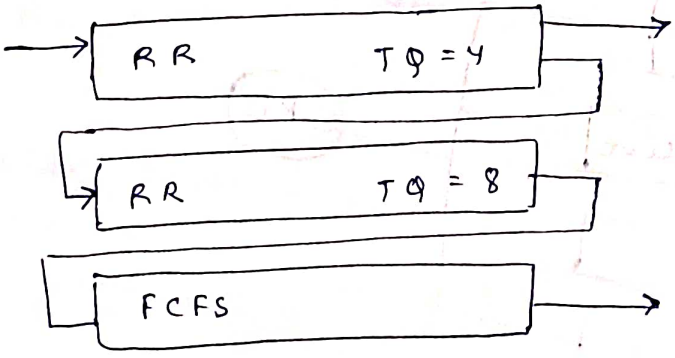
0 2 4 6 7 10 15 20

Avg TAT = $\frac{38}{4}$ = 9.5

Multilevel Feedback Queue

TQ = time quantum

By default level follows
RR
RR
FCFS



P.No.	AT	BT	CT	TAT
P_1	0	~~15~~ 10		
P_2	1	~~10~~ 5		
P_3	1	~~20~~ 15		
P_4	2	~~7~~ 2		

q_1 | ~~P_1~~ | ~~P_2~~ | ~~P_3~~ | ~~P_4~~ |

RR = 5 Gantt chart :

P_1	P_2	P_3	P_4	P_1	P_2	P_3

0 5 10 15 20 30 35 40

q_2 | ~~P_1~~ | ~~P_2~~ | ~~P_3~~ | ~~P_4~~ |

RR = 10

P_4	P_3

45 47 52

q_3 | P_3 |

If BT = 40, TQ = 2 } Increment by 5

TQ = 2 → TQ = 7 → TQ = 12 → TQ = 17 → TQ = 22

5 queues are used.

Q(2)

P.No.	AT	BT	CT	TAT	WT
P_1	0	53			
P_2	0	17			
P_3	0	68			
P_4	0	24			

11/10/23 **Traditional Unix Scheduling**

1) Kernel Non-Interruptable Mode
2) Kernel Interruptable Mode.
3) User Mode.

★ Priority = Base Priority of process + f(CPU time of process) + nice
↳ by user

★ Priority = Base priority of process + f(CPU time of process) + nice
↳ defined by user
↳ user can only dec priority not inc.

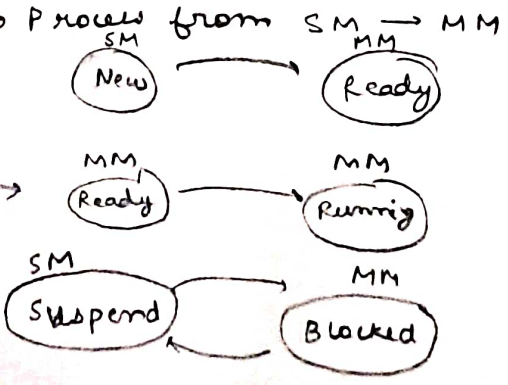
★ For fair scheduling, since multiple users in unix with multiple processes.

Priority = Base priority of process + f(CPU time of process) + nice + f(CPU time of process group)
↳ no. of process in that group

→ The UNIX scheduler ~~is~~ employs multilevel feedback using Round-Robin within each of the priority queues. Priority is based on process type & its execution history. The priority of each process is re computed once per sec at which time a new scheduling decision is made.

Types of Scheduling & Scheduler

1) Long Term Scheduling (Job scheduler) → Process from SM → MM
2) Short Term (CPU scheduler)
3) Medium (swapper)



- The aim of processor scheduling is to assign processes to be executed by processor in a way, that meets different objectives such as response time, throughput & processor efficiency

1. Long term scheduling :

It is performed when a new process is created, it is determined by a long term scheduler, which programs are admitted in the system for processing.

The scheduler must decide when the O.S can take 1/ more additional process. It controls degree of multiprogramming
↳ no. of processes in ready queue

2. Short Term Scheduling

The short Term scheduling makes the decision to transfer processes from ready state to running state.

The Short Term scheduler aka dispatcher executes most frequently & makes the decision which process to execute next

3. Medium Term Scheduler

The swapping in decision b/w suspend & block state is based on managing the degree of multiprogramming.

This is the duty of Medium Term Scheduler.

★ Convoy effect = smaller processes are stuck behind longer processes.

Algo	FCFS	~~SJF~~	~~SJF SRTF~~	~~Priority~~	~~SRTF Round-Robin~~
Criteria	AT	BT		~~Priority~~	BT
Mode	Non-preem.	Non-preemtive		Preemptive	
Advant.	1) Easy to implement	1) Minimizes RT		1) Beneficial if we predict next process	
Disadv.	2) Simplest	2) Improved WT			
Disadv.	1) Convoy effect	1) Difficult to predict Next process		1) Complex to execute	
		2) Starvation		2) Prediction difficult	

Priority	Round-Robin
Priority	Time Quantum
Preemptive	Pre
1) Priority gets chance	1) Every process gets chance
1) Starvation of Low priority process	2) NO starvation
	3) NO convoy effect
	1) Not suitable for less time quantum
	2) Overhead of context switching

13/10/23

Process Control

- Modes of Execution
  1) User Mode
  2) System Mode / Kernel Mode
- Process Creation
- Process Switching
- Mode Switching

Modes of Execution

Processor execution normally works in 2 modes:

1) User Mode - The less privileged mode, is often referred as user Mode as user programs typically execute in this mode.

2) The more privileged mode is Kernel/system mode. More privileged & imp. instructions are run in this mode. There is a bit in PSW that indicates the mode of execution. The bit is changed in response to certain events. For eg - When a user makes a call to OS service, or when an interrupt triggers, after execution of OS routine, mode is set to kernel mode, & upon return from the service to the user process the mode is set to the user mode.

2. Process Execution

- Process ID
- Allocate some space
  - Default
  - User
  - By parent process (process spawning)
- PCB initialization
- Setting appropriate links

(1) Assign unique process ID to the new process.

(2) Allocate space for the process

→ OS must know, how much space is needed for the private user address space & the user stack. These values can be assigned by default based on the type of process. Or they can be set based on user request or it's a

process is spawned by another process then space is allocated by parent process

(3) PCB initialization

All the entries are zero except program counter & system stack pointers. The priority may be set by default to the lowest priority unless a request is made for higher priority

(4) Setting appropriate links

If the OS maintains each scheduling queue then new process must be put in new/ready state

3. Process Switching

(1) INTERRUPTS

- Clock Interrupt
- i/o "
- Supervisor call

When to switch process :-

Two kinds of system interrupt are present which may need process switching :-

(1) INTERRUPTS

1) Clock Interrupt

→ When the time slice / quantum of any process expires then, so this process must be switched to the ready state

2) i/o interrupt :

→ If the i/o action contains an event for which processes are waiting then after the completion of i/o, OS moves the processes from block state to the ready state.

3) Supervisor call :

OS may be activated by a supervisor call from the program being executed. For eg - A user process is running & the instⁿ request an i/o operation. This call results in a transfer to a routine to the part of OS system part. The use of a system call may place the user process in the blocked state

~~This traps the~~

(2) TRAPS

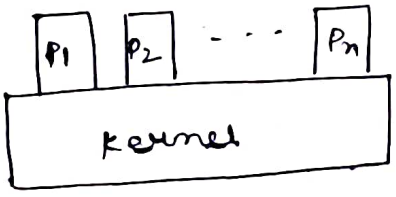
With the trap the OS determines if the error or exception will be handled / not. If it cannot be handled the process is moved to the exit state & if can be handled then OS envokes some recovery procedure.

4 Mode Switching

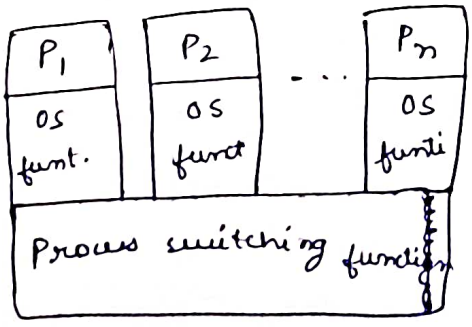
1) Interrupt occurs
2) user call to priviledged instruction.

EXECUTION OF OS

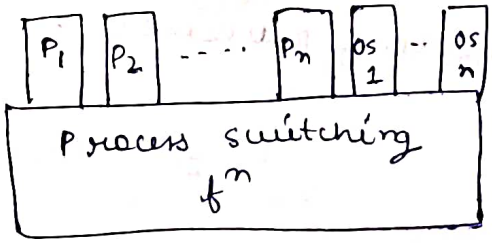
1)



2)



3)



Multiprocesses systems

1. Separate Kernel / Non process Kernel

The kernel is executed outside of any process. The process is considered to apply only to user programs & the operating system code is executed as a separate entity operating in privileged mode

2. Execution within user process

All OS software is executed in the context of user process. OS is a collection of routines. The user calls to perform different functions. A separate kernel stack, is used to manage calls while the process is in kernel mode.

3. Process based operating system

The major kernel f^n are organized as separate processes. There may be a small amt. of process switching code that is executed outside of any process. It is useful in multiprocessor or multicomputer environment in which some of the OS services can be shipped out to dedicated different processors.