19.07.2023
Wednesday

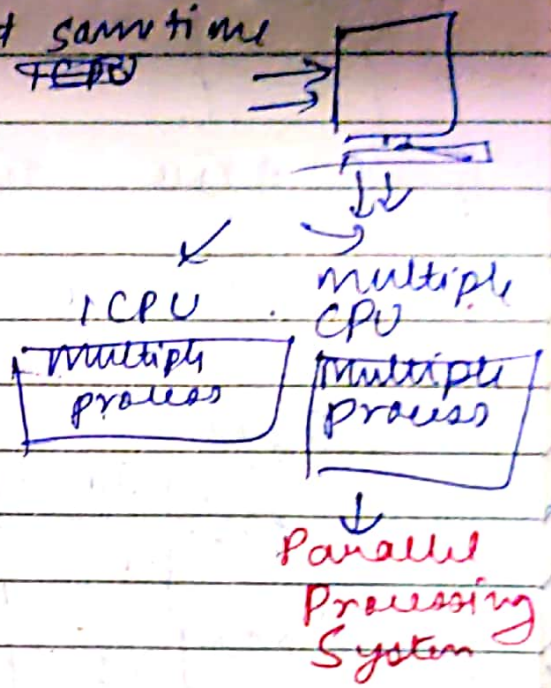
Lecture - 2

~~Batch Processing~~

Uni processing :- single CPU 1 CPU 1 process

Multi Processing :- Multiple process at same time

Time sharing



Thread

Multi threading

Process

Process control

State Transistion Diagram

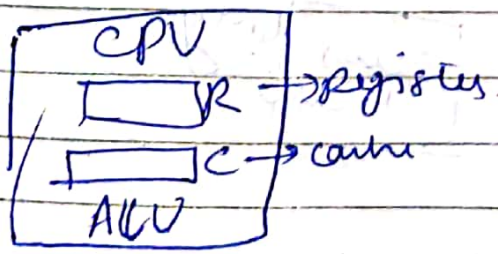
Time Sharing :-

Concept dividing execution time of CPU

user will not get feel. Switching is frequent

helps to achieve multiple processing

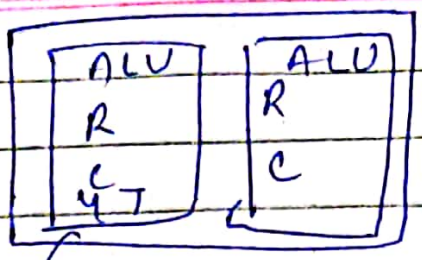
Threads :- Lost time element of execution



1 ALU = Single core

Scanned with CamScanner

RAM

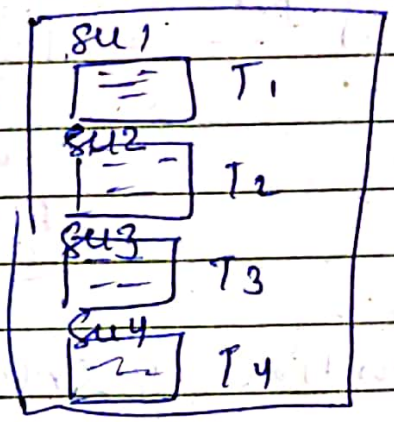


2 ALU = 2 cores
Dual core CPU

4 Threads } Parallel processing can be achieved

Basic entities for parallel processing

Independent instr which can be executed



In single core
= 0 speed up

In Dual core
= Speed up ↑

2 Threads at the same time.

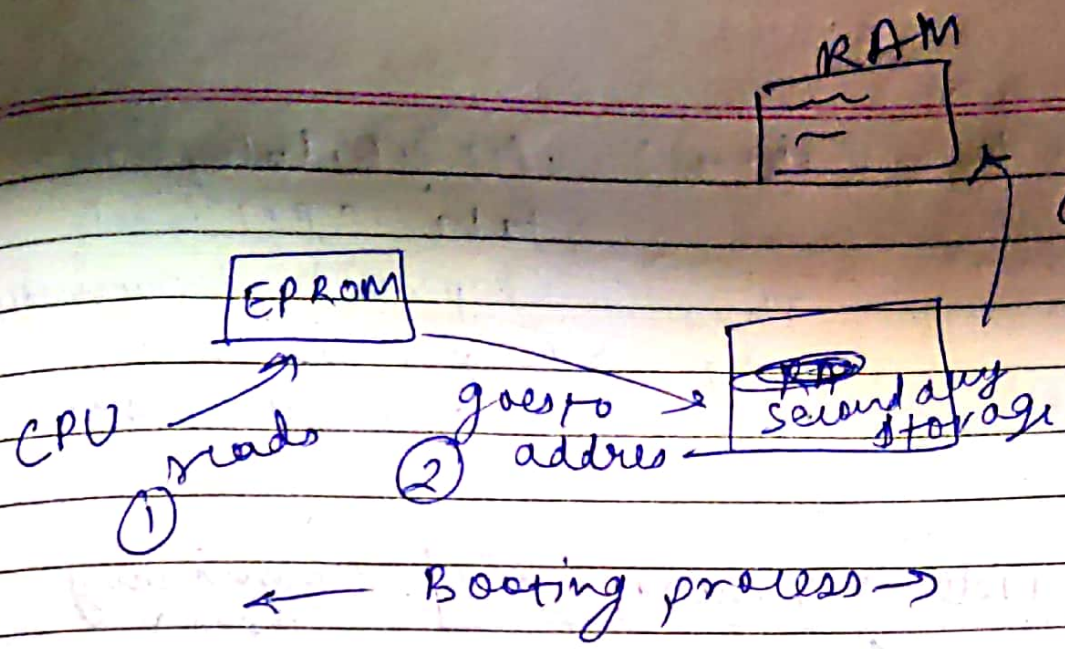
Process :-
CPU runs any program

EPROM
Electronically Programmable ROM

Primary RAM

Secondary device

When installing OS → these addresses are stored in EPROM



← Booting process →

In course, Time Sharing
(Single CPU, Single Core, multiple process)
(most primary resource)

OS : efficiently use of resources of comp.

Process Control : Do the process control
CPU scheduling Algos

State Transition Diagram :-
Process Creation
Execution
Termination
→ multiple time it goes in waiting state

CPU scheduling

Primitive
CPU will be taken away from the process.

Non Primitive
Process holds the CPU till it terminates or going in waiting state.

Scanned with CamScanner

CPU scheduling can't take CPU back
↳ process terminates
↳ process is in waiting state

Then only take CPU

Memory Management (managing RAM)

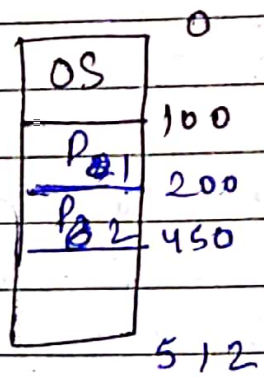
Continuous

Non Continuous.

→ Virtual memory

P_3 will break & store

P_1
P_3
P_2
P_3



{ fix for OS process

← P_3 (500) addr locations

Can't be store in RAM

① Complex mechanism for finding next address

② Good space allocation

Getting the next address

① so easy

Start addr + offset = easy Algo

Fast

② Poor space Allocation

2.11.07.2023
Tuesday

Virtual Memory Type of Non continuous Allocation

↳ Concept.

Game of 20 GB in 8 GB RAM.

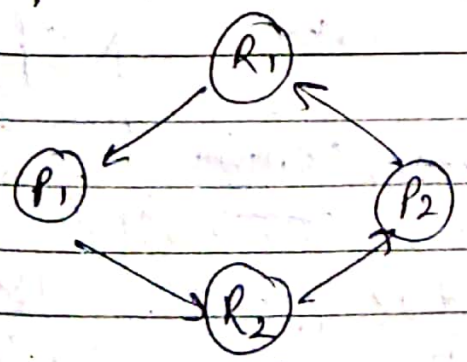
How?

With the help of VM
concept of Paging & Segmentation

Technique to ~~RAM~~ break process

Part of the Game on RAM which is executed ~~AN~~ by ALU & then keep it in Hard Disc & then ~~keep~~ next part get ~~at~~ again in RAM

Dead Lock :- The state or loop from which are unable to come out or not further execution is possible.



$R_1 \longrightarrow P_1$

P_1 is having the resource R_1

and

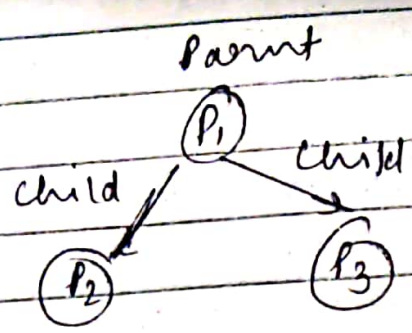
$R_2 \longleftarrow P_2$

P_2 is requesting for R_2

Scanned with CamScanner

Process Synchronization :-

→ (IPC) Inter Process communication



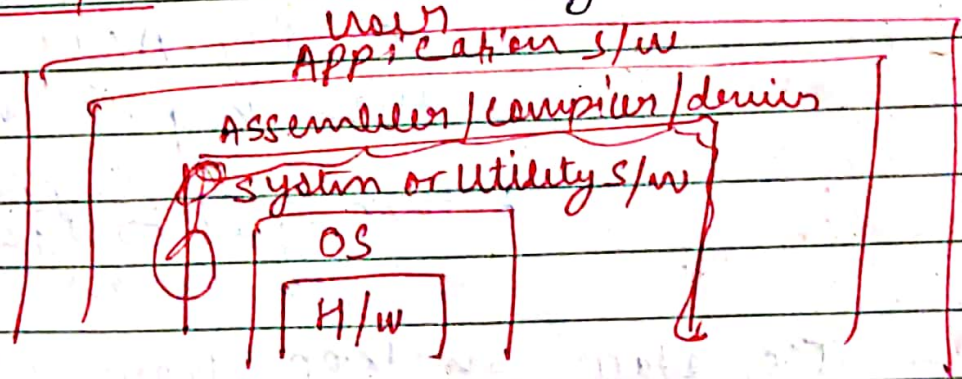
All the resources are allocated to P_1

P_2 & P_3 will request resources from P_1

This is called IPC.

Application s/w :- Not mandatory to run OS on computer

System s/w :- Mandatory to run OS on computer



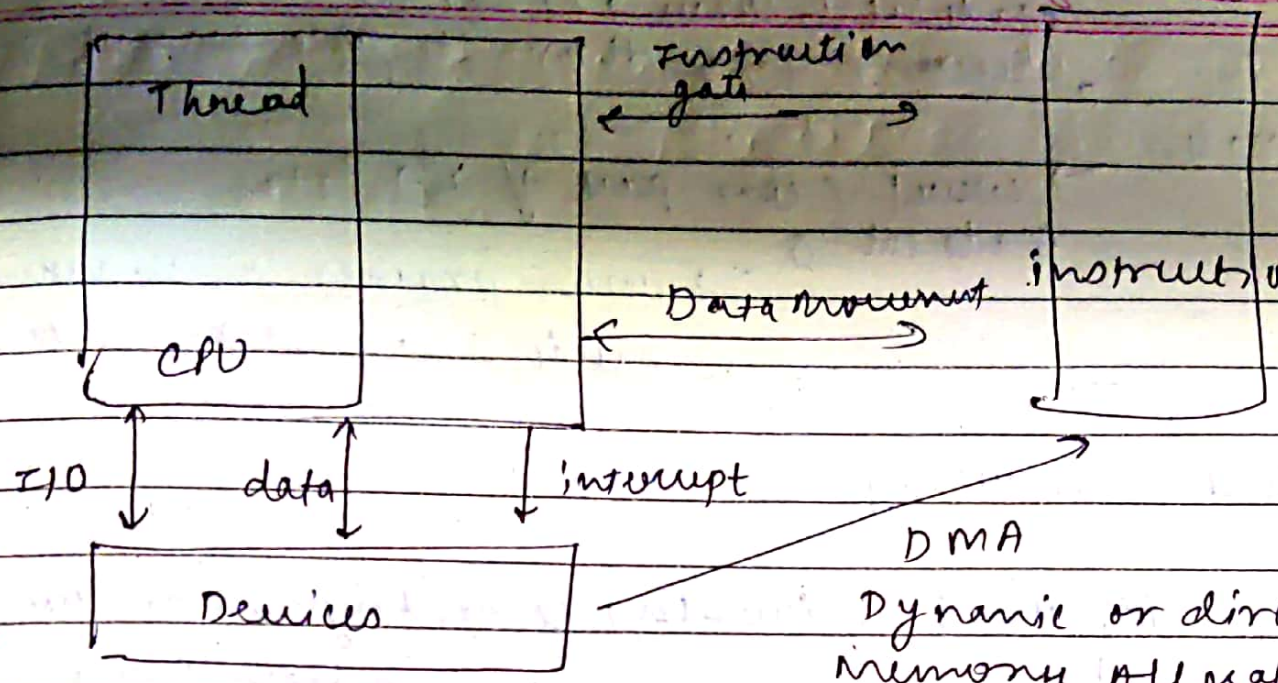
Memory Hierarchy

Flip flop (But not in OS)
↓
Register 1KB
↓
Cache
↓
Main Memory (RAM)
↓
Electronic device (SSD)
↓
Magnetic disk (HDD)
↓
Optical Disk
↓
Magnetic Tape

↑ Speed ↑ Size ↓

Secondary memory

Obsolete



DMA
Dynamic or direct
Memory Allocation.

Process Managment → modes → user, kernel.

File Protection & security → Type of Systems
Distributed
Special purpose
computer Environment
Peer to ~~Peer~~ Peer computing
Master-Slave computing
Web-Based Computing

Types of OS:-

Batch

Interactive

Time sharing

Real Time

Scanned with CamScanner

mode bit = 1 → when process ~~ota~~ uses user code then it goes in user mode
user

mode → kernel (core part of OS)
mode bit = 0 → when process ← → kernel
mode ← → kernel

There is switching betn modes.

MS word (writing & reading in MS word)
↳ user mode
↓
same fn ( )
↓
process kernel mode (same in h/w
only OS can do it
∴ use same fn OS)

trace cmd in linux → process ~~stry~~ history

File Protection & Security

Three types of user (in linux terms)
① User (u)
② Group (g) → consists of several users
③ Other (o)
↳ Remaining

user.. user login in system

r w x
read write execute

↳ For any file there are 3 types of permissions

→ For this

Part 1

Group 1 Group 2 Group 3

{2, 3}
↳ others

these are others

Scanned with CamScanner

Folder → also as file

Granting & Removing Permission [chmod]
r w x

~~Distribu~~ Types of systems

Distributed
- Different heterogenous machines are connected with each other
diff. architecture / specifications.
- No central power.

Ex: Lab

Special purpose
Machine device for any special purpose only
ex. ATM

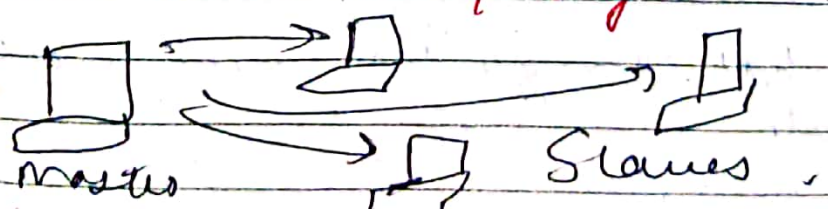
Computing Environment

Online — Offline

Online IDE
(Accessed through network)
internet

Peer to Peer Computing :- No central (Special case Distributed)
computing , distributed Computing
All are equal ~~capab~~ power of execution

Master Slave Computing :- ~~All~~ (Special case of Distributed)



Scanned with CamScanner

(Special case of Online computing)

Web Based computing : Online computing

Cloud computing (taking services + data)

Types of OS (IBM 650)

① Batch :- Entire job at a time we need to submit

② Interacting :- ATM Response & interaction

Opp. Passive/ Not interactive

③ Time sharing :- Share P_1 P_2 P_3 time by time gives time to all.

Ex. printer

④ Real Time :- Response time must be very high

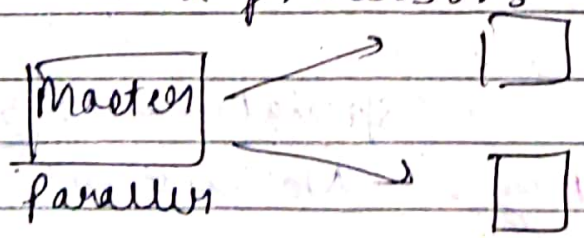
Ex: Missile launching system, car system, Air Traffic Controller

⑤ Network :- In Distributed system (Network)

Ex: Window NT (Networking) (Mac OS X Window NT)

⑥ Parallel :- work done in Parallel.

↳ 2 processors + 2 machines



Parallel

⑦ Distributed Advanced Network OS (LOCUS)

⑧ Clustered (in Distributed)

⑨ Handheld : OS that can go in small machine

Ex. Mobile.

Scanned with CamScanner

9th + 10th

Concept of OS by Galvin.

→ 90%

S C Dhamidhare

Time sharing — Single CPU — Multiple process

Parallel — Multiple CPU — Parallel → Single process, Multiple process

OS Services:-

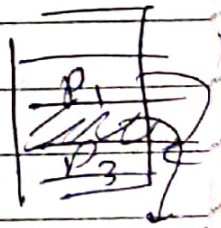
① Program Execution — CPU scheduling.
FCFS, STF, Priority. - Round Robin

Deadlock (RAG): Resource allocation graph

② Resource allocation — assing resources to the process
→ H/w resource : printer, scanner
→ S/w Resource : Semaphore

③ I/O operation.

④ Communication → methods: Shared memory, Message passing



Shared memory

⑤ Error ~~Detection~~ Detection / Handling (Debugging)

⑥ Protection : describes a method or a way to manage user, process designated by a comp. system.

⑦ Security — The main protection related activity of an OS are
(i) The OS makes sure that all system resources have controlled access
(ii) The OS offers password based authentication features for user.

⑧ File system Manipulation

Scanned with CamScanner

- The OS make sure external input output devices are secured against unauthorized access attempts.

- OS ensures that all access to sys. resources must be monitored and controlled,

It also ensures that the peripherals must be protected from invalid access. It provides authentication by using username & passwords.

File System /

Need to read & write files, directories and create, delete file & directory, list file info. and list permit info.

Functions of OS

- Process managing management
- Memory managing
- File managing
- Device mang

(1) CPU scheduling
(2) Process & PCB
(3) Deadlock
(4) Process synchronization Problems
(5) Reader - Writer problem
(6) Sleeping Barber problem
(7) Dinning Philosophies Problems.

Tough Numerical

Program up process

continuous memory Allocation
(i) First Fit
(ii) Best Fit
(iii) Worst Fit

Non - continuous Memory Allocation

Scanned with CamScanner

31.07.2023
Monday.

(i) Paging
(ii) Segmentation
(famous) { Big Topic & Numerical

Files & Directories
(i) File Allocation Methods
(ii) File Scheduling

$ ls -l

		u g o
→ ordinary file		rwx rwx rwx
- rwx rwx rwx ...	file1	All the type of users have all types of permission
d rwx rwx rwx ...	folder	
- r - x	file3	

→ dictionary / directory.

By default, execute is disabled ~~for~~.

x is not there then we can "search" cmd.

Layered Approach in OS Design

Module → code very very ~~f~~ ~~f~~ more

Remove unnecessary modules or add modules to inside the kernel.

~~+ ~~ .ko → kernel loaded file
↳ C program file (kernel)

Scanned with CamScanner

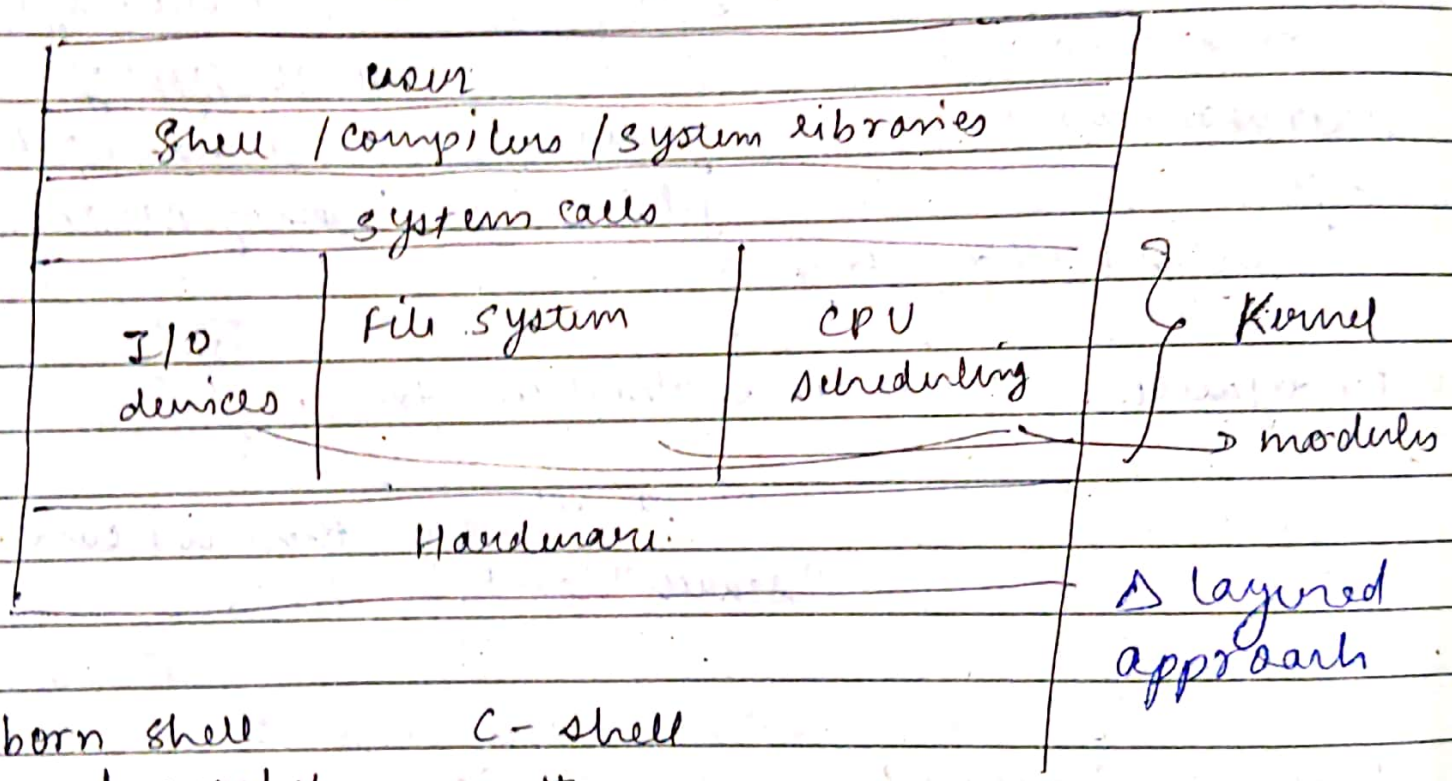
What is the name of Indian OS?
Name of kernel.

Open Source : Source code ~~of the prog~~ also available.
GUI is not included in kernel.

Kernel + GUI → Different flavours of
(same) different linux

(origin)
Unix → Not an open src.] 99% same
Linux → Open Src.

BSD → Berkeley S/W Distributions.



born shell
$ symbol is there

C - shell
#

Shell Scripting :-

We can club the kernel functionalities & system calls in program

System calls interact with modules.
read, write, open, → cmds in file

Scanned with CamScanner

01.08.2023
Tuesday

Microkernel → Related to no. of modules

Macro kernel

Shell script is not compiled. It contains system calls (like function calls) to interact with

System Calls

API (Application Program Interface) → small procedure that can be included in code.
Weather API, Google Location API

Small Procedures that help to kernel

Types of System Calls :-

① Process Control: (Lab)

	Windows	Unix
initialize the process	Create Process( )	fork( )
terminate the process	Exit Process( )	exit( )
	Wait for Single object( )	write( )

Parent will not die until child does not die.
Need to write wait( ) in parent & child both.

② File Management :- (Lab)

Create file( )	open( )
Read file( )	Read( )
Write file( )	Write( )
Close file( )	Close( )

for successfully saving the content — File handler (Unix)

Scanned with CamScanner

③ Device :-

set console mode ( )	ioctl ( ) → input output control mode
Read console mode ( )	read ( )
Write consolemode ( )	write ( )

These are same as file
In Unix, everything is same as file.

④ Information Maintanence :-

Get current Process ID ( )	getpid ( )
Set Timer ( )	alarm ( ) → specific event output
Sleep ( )	sleep ( )

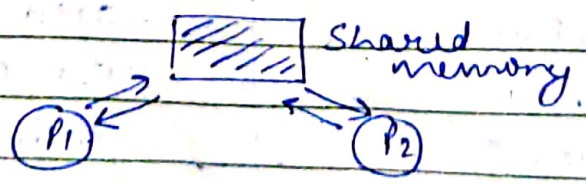
When used in shell scripting.

→ Intentionally sending some process to stop state.

⑤ Communication :-

Create Pipe ( )	pipe ( )
Create file mapping ( )	shmget ( ) Shared memory get ( )
Map view of file ( )	mmap ( ) memory mapping

→ O/p of first process can be send to i/p to 2nd process



mapping of memory (figure out the addr, which addr, conv. to other addr shift process)

Scanned with CamScanner

⑥ Protection :-

set file security( )	chmod( )
set Security Descriptor Group( )	chown( )

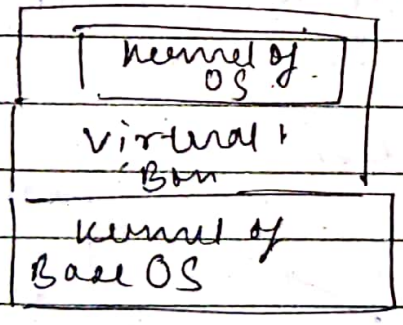
System call( ) & not command

change owner of the file.

System Calls :- C program in Unix
Benefit of Unix → Simple system calls.

VIRTUAL MACHINES

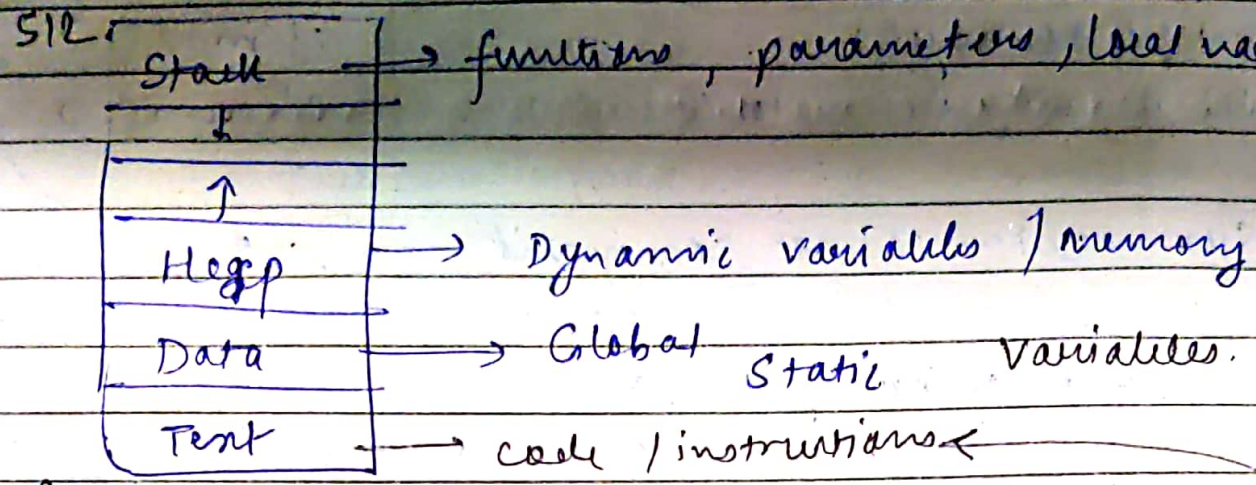
On same h/w architecture we can use different OS.
~~More~~
Multiple kernels on the same machine
copy of h/w to virtual machine



more secure.

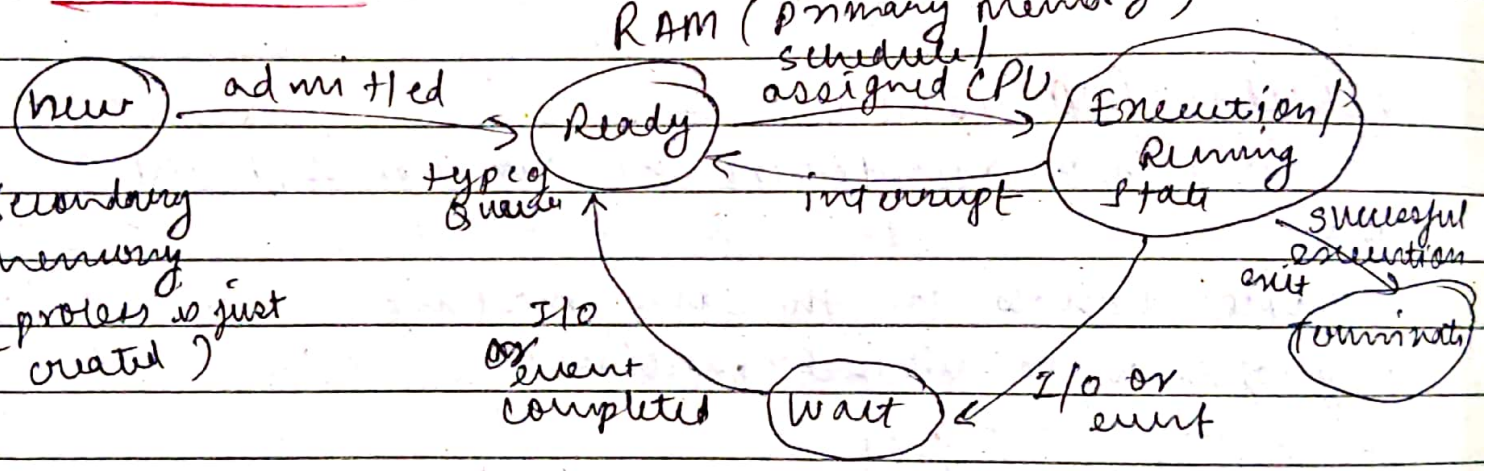
Rolling Robin
High priority process runs. ← preemptive
Non

Visualization of process in RAM



Process states (State Transition diagram) Simplum

(limited ones)



CPU should be allotted to other process while our process goes in "wait" state.

PCB (Process Control Block)

each process has corresponding PCB to it.

↳ contains all info. ~~around~~ of the process

1. Process state (new / ready / execution / wait / terminate)
2. Process number / Process ID CPU uses to find particular process
3. program counter → Register that stores add^n of next instruction
4. CPU scheduling → multiple process in Ready queue, which process to take?

Scanned with CamScanner

freq. (CPU) >> freq(i/o)

07.09.2023
Monday

First ~~Can~~ One First Serve

priority based, Round Robin, etc.

(5) Memory Management Information
↳ Base add^n of memory, Base table & segment table, size of memory, ~~Load time remaining~~

(6) Accounting Information
↳ Time limit, job or process ~~no~~ related to process no., Time for which CPU is allocated to process

(7) I/O Status Info^n
↳ available or allocated or required

(8) List of open files → whatever files are open are stored in it.

PROCESS~~OR~~ IN MEMORY

No fixed block is assigned Stack & Heap.
Why? Value of Stack & Heap can change
Stack → decrement the address
Heap → increment the address.

Registers are also there { holding address corresp-
-onding the stack.

Role of PCB: When we take back the CPU from a process & want to resume the process again then old info might get deleted. PCB helps in saving that info.

Load the PCB first in CPU

When CPU is taken back
Save PCB
↓
Load PCB
CPU is given

Switching of the CPU from process

Code & Text } are in RAM

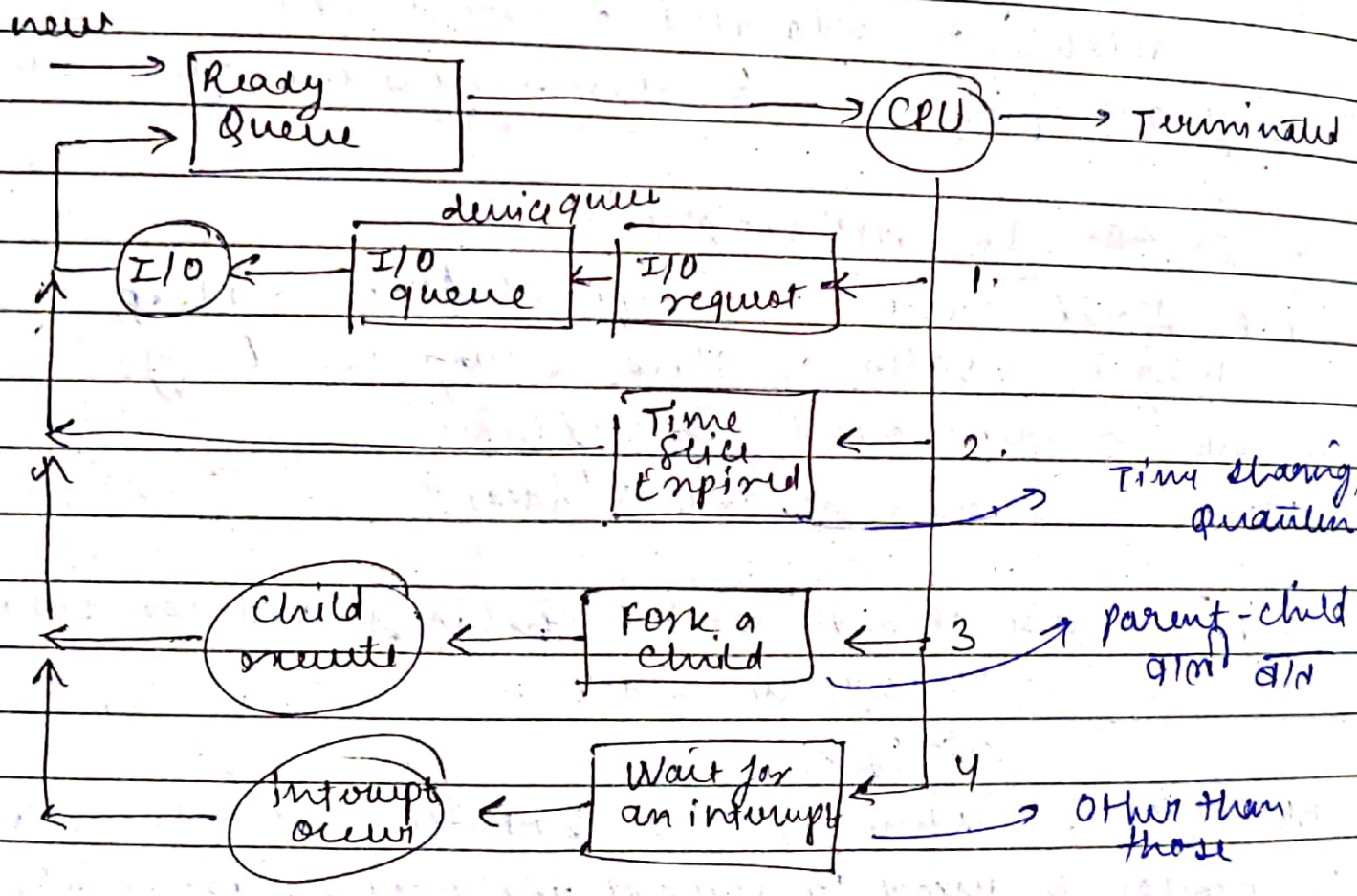
Scanned with CamScanner

Job Queue All the process in system residing in Job Queue

Ready Queue Process that are residing in main memory (waiting for CPU)

Device Queue Process that are waiting for i/o device

Different devices ⇒ different queues.



Two types of schedulers are there

① Long Term Scheduler (Also known as Job Scheduler) Responsibility to ~~to bring~~ decide which process will go from new → ready

freq. of this → less

② Short Term Scheduler

ready → executes

freq of this → high

freq(short) > freq(long)

Once p°

③ Medium Term scheduler :- Used when we are swapping the process

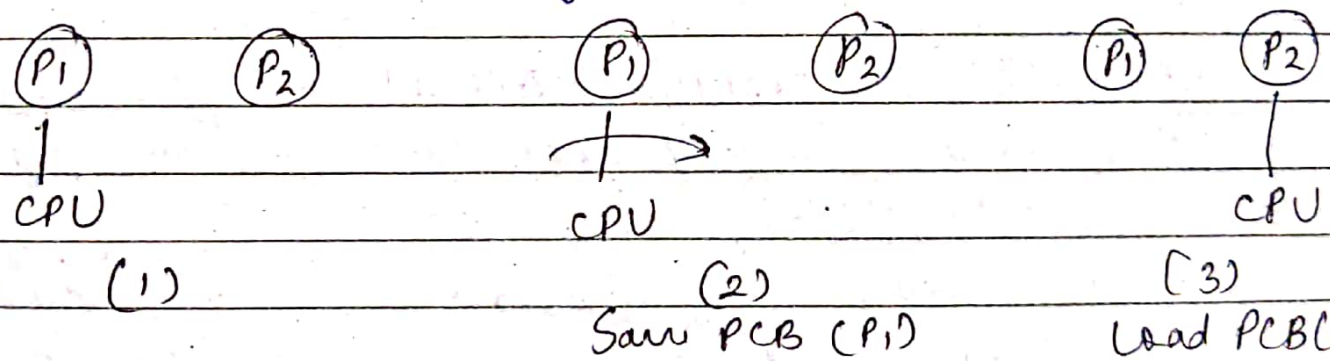
process main memory → secondary memory

due to lack of memory & ~~main~~ high priority process ~~to~~ come & there is requirement of memory then partially executed process is removed from main memory to secondary memory.

Very rare.

→ saving & loading the PCB depends on CPU scheduling algorithms

CONTEXT SWITCH of P_1



~~Suu~~ If context switching is done continuously

↳ not good

Time ↑ as calculation of add^n.

Overhead type.

∴ Need to avoid such CPU scheduling algorithms

Waiting time

Scanned with CamScanner

08.08.2023
Two day.

BOSS
kernel of linux (GUI)

parent is waiting for signal that child is completed

PROCESS CREATION

→ fork() → wait →

System call

fork did this? copy of parent process (with diff. process Id) [code + text]

child → exec() → exit()

System Call

New code of child

[All the process are child of Shell process]

init()

→ Who is the parent of Shell?

INTERPROCESS COMMUNICATION

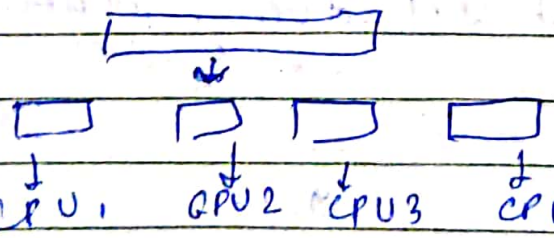
↳ Betn process change of data through memory sharing & message passing

Reasons:- (Normal)

1. → Information sharing
2. → computation speedup
3. → modularity
4. → convenience

computation speedup ↓ fast execution of process, reduce time of execution

(2) Speedup



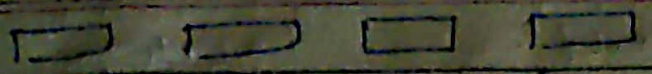
Divide

(2 s)

(Communicate with CPU)

Parallel = 2s

Scanned with CamScanner



Combine. (Exchange of data & information)

Interprocess communication is required

③ Dividing process into modules → communication is required

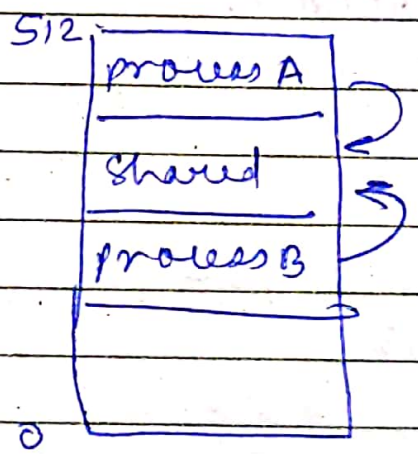
④ To handle multitasking we require

Methods to implement Interprocess Communication

(I) Shared memory
(II) Message passing

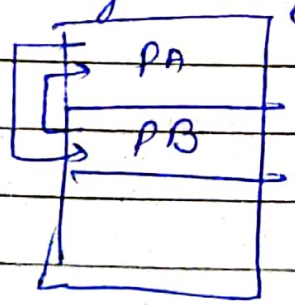
(I) Shared passing

(Semaphore)



R R OK
W W X
R W X
How to maintain this sequence

(II) Message Passing



Take Time

Time (MP) >> Time (SP)

Types of process :-

① Independent process → Not affected by any other process in the system

② Cooperating process :→ May be affected by any other process in system.

Scanned with CamScanner

Core = ALU + Cache + Regis

Parent → Child location sharing

Bigger task many processes (needs communicate in)

eg:
"Producer Consumer Problem"

P1 P2

items
items consume

P1 produce items
P2 consumes items

location to store items

P1 produces only when there is atleast.

MULTI PROGRAMMING

Multiple programs are running at the same time
seems to
MS word, Excell open at the same time

MULTI TASKING

Specific program/process is divided into multiple tasks
MS word → Spell check
→ typing

No talking of multiple cores.

Ready Queue

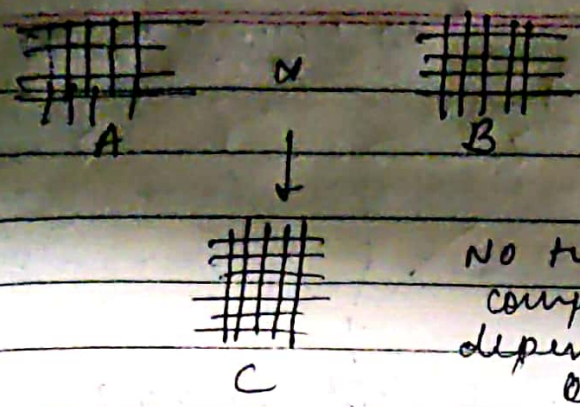
MULTI CORE - PROGRAMMING.

↳ multi-programming / multitasking

independently execute

Threads

Scanned with CamScanner



No two cells computation depends on each other } independent } multi core programming

Divide A & B into 4 to 5 parts

T_1	T_2	T_3
T_4	T_5	T_6
T_7	T_8	T_9

T_i $i = 1$ to 9 would be done in parallel
Task $_i$

Whenever we try to solve problem using multi-core programming keep these points in mind

Dividing activity → Dividing work
Balance → no. of cores. (equal partition)
Data Splitting → proper splitting of given data
Data Dependency → writing dependency inc or not?
Testing & Debugging

NO
No two cells depends on each other

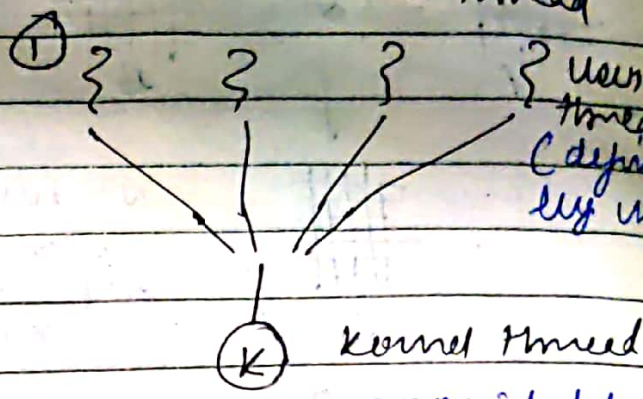
Check the load of core/CPU

problem:
how to store matrix in ~~two~~ main memory
↳ only req. no. of rows & no. of columns

→ mapping between user thread to kernel thread

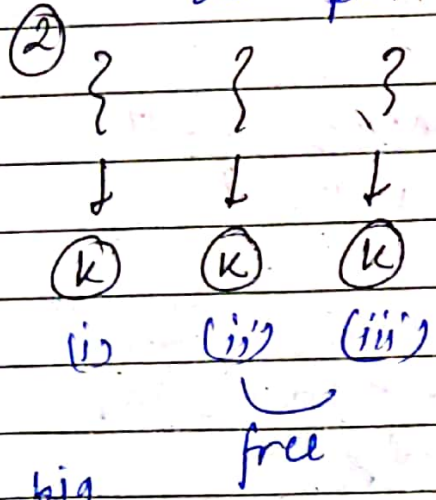
Multithreading models

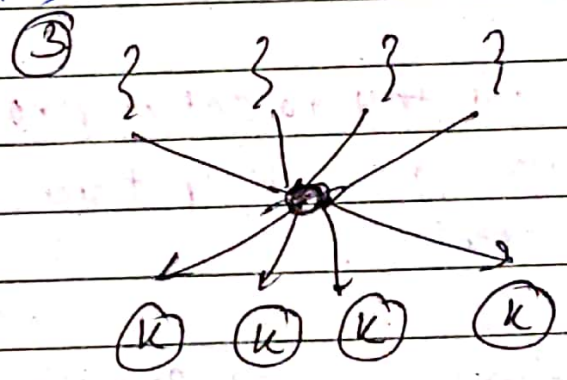
1. Many to one
2. One to one
3. Many to many
4. Two level model



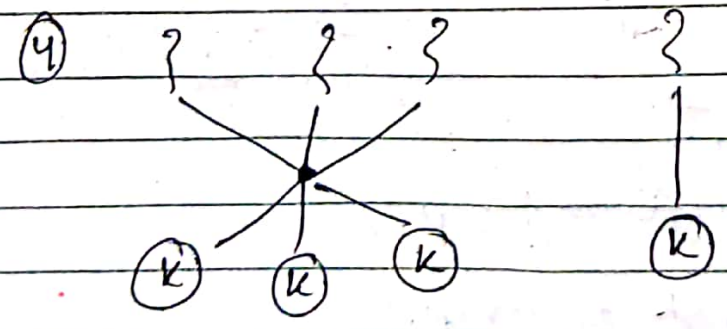
Many thread of user is managed by single kernel thread.

(User can create n no. of threads but managing as per kernel)





big work can't assign to (ii) & (iii) which can be in 3



user get the kernel thread & then assign this/the thread to kernel thread

or

if randomly assigned thread then it will go to any one thread (free one)

Scanned with CamScanner

09.08.2023
Wednesday

CPU Utilization

CPU burst → process is using CPU
I/O burst → process is using I/O devices

Process = combⁿ of (CPU + I/O) burst.
↳ all process starts from CPU burst.

~~CPU~~ I/O bound processes — major → I/O burst
CPU bound processes — major → CPU burst

CPU Scheduler

maximize the CPU utilization ↑

4 condⁿ based on which CPU scheduler takes

1. Process switches from running to waiting state.
2. Process switches from running to ready state.
3. Process switches from waiting to ready state.
4. Process terminates.

scheduling under 1 & 4 is known non-preemptive

non-preemptive.
↳ not allowed to take back the CPU
process is voluntarily giving the CPU

2 & 3 under preemptive.

Scheduling Criteria

1. CPU utilization
2. Throughput.

→ entire system not process

3. Turn Around time.
4. Response time.
5. Waiting time.

for each process & take average afterwards

↑ CPU Utilization ↑ :- Need to keep CPU busy all the time.

↑ Throughput :- No. of process that complete their execution per unit of time.

↓ Turn around Time :- This is the Amount of time executes a particular process (Time betn time of submission to the time of process termination)

↓ Waiting time :- This is the amt. of time that a process spends in the Ready Queue. (not in IO waiting time)
time wrt. CPU

↓ Response time :- The amt. of time process takes from the request submitted to the first response given by the CPU. (first time your process is getting CPU)

FCFS (FIRST COME FIRST SERVE)

Q)

Process	Burst Time
P_1	24
P_2	3
P_3	3

If i/o not given By default CPU burst time.

Gantt Chart ✓ (IMP)

P_1	P_2	P_3

0 24 27 30

Turn Around Time (TAT) = Termination time − Arrival Time

If Arr. time not given assume 0

$TAT_{P_1} = 24 - 0 = 24$

$TAT_{P_2} = 27 - 0 = 27$

$TAT_{P_3} = 30 - 0 = 30$

$\frac{24 + 27 + 30}{3}$

Avg. TAT = 27

Waiting Time (WT) = TAT − Burst

$WT_{P_1} = 24 - 24 = 0$

$WT_{P_2} = 27 - 3 = 24$

$WT_{P_3} = 30 - 3 = 27$

P_1 started then P_2 was in Ready Queue.

Avg. WT = 17

$\frac{0 + 24 + 27}{3}$

Q)

Process	Burst Time	Arrival
P_2	3	0
P_3	3	0
P_1	24	0

GC:

P_2	P_3	P_1

0 3 6 30

$TAT_{P_2} = 3 - 0 = 3$

$TAT_{P_3} = 6 - 0 = 6$

$TAT_{P_1} = 30 - 0 = 30$

Avg. TAT = 13

$WT_{P_2} = 3 - 3 = 0$

$WT_{P_3} = 6 - 3 = 3$

$WT_{P_1} = 30 - 24 = 6$

Avg. WT = 3

Why improvement in WT?

Change in order of process. P_1 is big process so we kept it in Last

Scanned with CamScanner

16.08.2023 Wednesday

In case of Non-preemptive (like FCFS)

$\boxed{\text{Waiting time} == \text{Response time}}$

Convoy ~~Conflict~~ Effect — small/Short processes are waiting for execution long process.

case 1

short process behind/after long process.

CPU Scheduling

Non Preemptive	Preemptive
① FCFS (First come first serve)	① Round-Robin (RR)
② SJF (Shortest Job First) ⟷ Same Principle	② SRTF (Shortest Remaining Time First) → preemptive version of SJF
③ Priority ⟷ Same	③ Preemptive Priority
④ Multilevel queue	④ Multilevel Feedback Queue
⑤ Future knowledge	

② Shortest Job First (SJF)

Q)

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

Sol: ⇒ Do not consider sequence.

Scanned with CamScanner

P_4	P_1	P_3	P_2

0 3 9 16 24

TAT = Termination - Arrival

$TAT_{P_4} = 3 - 0 = 3$

$TAT_{P_1} = 9 - 0 = 9$

$TAT_{P_3} = 16 - 0 = 16$

$TAT_{P_2} = 24 - 0 = 24$

$TAT_{avg} = \frac{52}{4} = 13$

~~WA~~ WT = TAT - Burst Time

$WT_{P_4} = TAT_{P_4} - BT_{P_4} = 3 - 3 = 0$

$WT_{P_1} = 9 - 6 = 3$

$WT_{P_3} = 16 - 7 = 9$

$WT_{P_2} = 24 - 8 = 16$

$WT_{avg} = \frac{28}{4} = 7$

Response Time = Waiting (Non Preemptive process)

Q)

Process	Burst Time	Arrival time
P_1	6	0
P_2	8	2
P_3	7	4
P_4	3	5

Sol: When arrival time check only $t = t_0$ for all process, All P available

P_1	P_4	P_3	P_2

0 6 9 16 24

Only process P_1 available

Non preemptive ∴ Can't take A bath

$TAT_1 = 6 - 0 = 6$

$TAT_2 = 24 - 2 = 22$

$TAT_3 = 16 - 4 = 12$

$TAT_4 = 9 - 5 = 4$

$TAT_{avg} = \frac{44}{4} = 11$

$WAT_1 = 6 - 6 = 0$

$WAT_2 = 22 - 8 = 14$

$WAT_3 = 12 - 7 = 5$

$WAT_4 = 4 - 3 = 1$

$WAT_{avg} = \frac{20}{4} = 5$

Response time = Waiting time (Non preemptive)

Q)

Process	Burst Time
P_1	20
P_2	12
P_3	2
P_4	6

Sol: ~~$T_1 = 20 - 0$~~ ~~$T_1 = 2 - 0 =$~~

P_3	P_4	P_2	P_1

0 2 8 20 40

$T_1 = \cancel{2 - 0 = 2}$ 40

$T_2 = \cancel{8 - 0 = 8}$ 20

$T_3 = 2$

$T_4 = 8$

$T_{avg} = \frac{70}{4} = 17.5$

$W_1 = 40 - 20 = 20$

$W_2 = 20 - 12 = 8$

$W_3 = 2 - 2 = 0$

$W_4 = 8 - 6 = 2$

$W_{avg} = \frac{30}{4} = 7.5$

= Response Time (Non preemptive)

Scanned with CamScanner

Advantage:-

$\boxed{WT(SJF) < WT(FCFS)}$ $\boxed{\text{Convoy Effect solved}}$

Disadvantage:

Randomly new process come & has less Burst time than other. then always we have to entertain for the process (small) to come. Big processes have to wait for long.

This is called "problem of Starvation"

Indefinate postponement of processes having larger Burst time.

→ Aging : how much time they are waiting Give this thing also to solve this problem.

We can overcome the problem of Starvation.

SJF is mainly used as long term scheduling algorithm.

Determination of Next CPU Burst :-

↳ mostly applied New process.

t_n = Actual length of nth CPU Burst

τ_{n+1} = predicted value for the next CPU burst

α , $0 \le \alpha \le 1$

$$\boxed{\tau_{n+1} = \alpha * t_n + (1-\alpha) * \tau_n}$$

$\alpha = 0.5$ → 50% for actual
→ 50% for predicted

If not correct then modify α.

$\alpha = 0$ → actual part = 0
→ totally depends on predicted

$\alpha = 1$ → depends on actual
→ No depend on prediction