* Operating Systems * [16th March, 2022]

#Syllabus of OS.

1. Basic Introduction → types, process diagram (Stages) → sys-call (Kernel)
2. ** Process Scheduling (CPU-sch) {FIFO, SJF, RR Algos}
3. Process Sync. → Semaphores
4. * Deadlock & Threads → {Banker's Algo}
   - Theory
5. ** Memory Management → [(Virtual Memory) Paging, Segment? Fragmentation (Page repla Algo.)]
6. * Disk Scheduling (Num.) {C-scan Algo, FCFS ...}
7. UNIX commands → Linux cmds {ls, mkdir, cd, chmod, open (sys-call)}
8. File Management and Security → {seqⁿ, Rand, Linked Access allocation} → vulerablitjie Attacks.

Process block ← Starvation

Big process takes T. → Processes pace slowed! ← Convoy Effect

* Starvation & Convoy Effect ☑ Last Class.

Pre-emptive Scheduling (Power to stop running proc)

* CPU is allocated to process for a limited time.

Non-Pre-emptive scheduler.

* CPU is allocated to process till it terminates or switches to wait for state

Starvation / Convoy Effect:

Starvation ↑ indefinite time

Convoy Effect ↑ Place slow

Time	1000	2000	100	10	20
Process	P_1	P_2	P_3	P_4	P_5
Priority	5	4	3	1	2

1 → low priority

P_1 → 1000, P_2 → 2000, P_3 → 100 } Big Processes → Wait 3100 sec.

P_4 → 10 ← then!

(long term scheduler)

Scheduling Algorithms:

① FCFS (first-come-first-serve)

* whichever process LTS brings to ready Queue gets CPU time first

(1) FCFS manner
(2) Order of requests →
(3) Non-preemptive → terminate, I/O o/p

Adv {Simple, No Starvation}

Disadv {Convoy}

② SJF (Shortest Job First)

adv. (No convoy)

dis-adv (Starvation, Not implementable)

(1) Based on selⁿ of smallest burst time of process
(2) Non-Pre-emptive.
(3) Service time is Known before execution!

NOTE:

✓ SJF is not implementable as we don't know burst time! (service-time)

* can't forecast accurately.

Even if we can is basically based on prev. results!!

Starvation for Big processes in SJF.

Numerical: (First Come First Serve)

Completion Time · Turn Around Time · Waiting Time

C.S. Context switching ↓ {State store extra ops.}

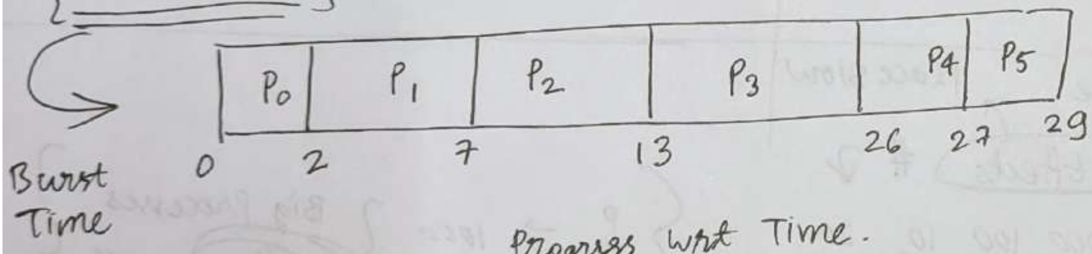
Switching ~~overhead~~

Processes	Arrival Time	Burst Time	C.T.	TAT	WT
P_0	0	2	2	0	0
P_1	2	5	7	5	0
P_2	4	6	13	9	3
P_3	5	13	26	21	8
P_4	6	1	27	21	20
P_5	7	2	29	22	20

T.A.T Turn Around Time

Waiting Time

{Gantt chart} → work(process) wrt Time.



Progress wrt Time.

Turn Around Time formula.

$\boxed{TAT = (CT - AT)}$

$\boxed{WT = TAT - BT}$

Amt it had to occupy - ~~actual time it took~~

* SJF based (Shortest Job First based Numerical)

Process	Arrival Time	Burst Time
P_4 wrong	6	1
P_5	7	2 } same?
P_0	0	2
P_1	2	5
P_2	4	6
P_3	5	13

Process	Arrival Time	Burst Time
P_0	0	2
P_1	2	5
P_2	4	6
P_3	5	13
P_4	6	1
P_5	7	2

RQ

P_0 available

0th sec.

How can P_4 come ??

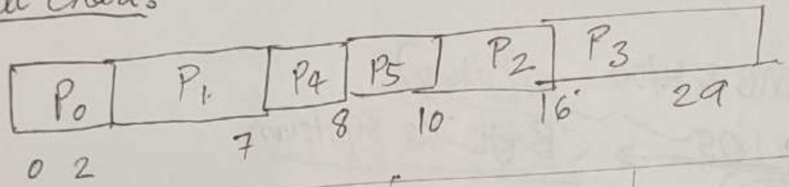
only process available so it's shortest itself ✓

when they arrive, only then they can be compared on pro-Burst time basis!!

Check acc, to Arrival time

give CPU acc to Burst time!

Gantt chart:



CT-AT = TAT-BT =

Table

Process	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
P_0	0	2	2	2	0
P_1	2	5	7	5	0
P_2	4	6	16	12	6
P_3	5	13	29	24	11
P4	6	1	8	2	1
P5	7	2	10	3	1

$\frac{\Sigma}{n} = 48/6 = 8$ $= 19/6 \sim 3$

Comparing on basis of burst time & check basis of arrival time!

SRTF - SJF's pre-emptive version!
{Shortest Remaining Time First}

* pre- emptive

* Simmilar to SJF, but pre-emptive
If any smaller Job comes during executoo, it pre-empts & executes the SJF (compared to rem. time)

Adv - No convoy ; Avg. WT ↓

Dis - Starvation
Not implementable ✗.

Booting (What happens when Computer turns on?)

Step1: Power ON
power supply to MB & H/W → (chip)

Step2: CPU loads BIOS → Basit I/O system

Step3: BIOS runs test ✓
→ loads some settings from mem. area
→ POST - Power on self test OK → enables Bootloader

Step4: BIOS - hands ff to boot device → MBR {master Boot record}

Step5: Boot loads OS!