CPU SHEDULING

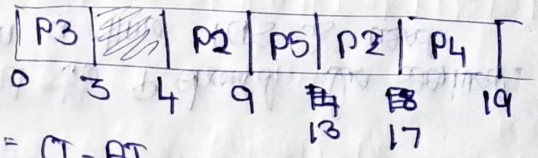
MODULE - 4

FCFS

MODE: Non-preemptive
CRITERIA: Arrival time.

PID	AT	BT	CT	TAT	WT
1	4	5	9	5	0
2	6	4	19	11	7
3	0	3	3	3	0
4	6	2	19	13	11
5	5	4	13	8	4

Gantt-Chart



TAT = CT - AT
WT = TAT - BT

CONVOY EFFECT?

Happens when longer Job (BT) is in front of ready queue of smaller jobs.

* When this longer job in ready queue, smaller jobs will finish the I/O operation & move into ready queue & sits idle.
* Also now the I/O device is idle
* Eventually, longer one finishes & goes to I/O device
* But quickly, shorter job of small BT finishes & goes to the I/O queue.
* Now, the CPU sits idle.
* This cycle continues.

LEADS TO: Inc in WT, inefficency; decrease in throughput & CPU utilisation.

SOLUTION:

1. SJB
2. RR
3. PS
4. Multi-level queue scheduling.

SJF

@ NP WT = RT

"SHORTEST JOB FIRST"

MODE: Non-preemptive.

CRITERIA: BT sorted, least BT is chose.

① First the P should be arrived.
② on a time FCFS is used.

P	AT	BT	CT	TAT	WT	RT
1	0	7	7	7	0	0
2	2	4	12	10	6	6
3	4	1	8	4	3	3
4	5	4	16	11	7	7

Gantt-chart

P1	P3	P2	P4
0	7	8	12	16

Shortest - NEXT - CPU - BT algorithm.

"The goal difficulty, finding LENGTH/BT of next process"

It's fine with LONG TERM shedulers but NOT with SHORT TERM.

Next BT → Exponential avg of previous BT.

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

↑ Predicte ↑ actual BT ↑ Predicted $0 < \alpha < 1$ Mostly $\alpha = 1/2$

SJF is also a priority algorithm. Where priority is given to inverse of next BT or the shortest BT.

RT = First CT − AT

SRTF "SHORTEST REMAINING TIME FIRST"

MODE: Preemptive
CRITERIA: Priority to SHORTEST BT.

P	AT	BT	CT	TAT	WT	RT
P1	0	~~8~~ 7	17	17	9	0
P2	1	4 — ①	5	4	0	0
P3	2	9	26	24	15	15 starts 17 arrive 2
P4	3	5 — ②	10	7	2	2 starts 5 arrive 3

GANTT:

| P1 | P2 | P4 | P1 | P3 |
0 1 5 10 17 26

P	AT	BT	CT	TAT	WT	RT
P1	0	~~12~~ 10	27	27	15	0
P2	2	4	6	4	0	0
P3	3	6	12	~~14~~ 9	3	3
P4	8	5	17	~~3~~ 9	4	4

GANTT CHART:

| P1 | P2 | ~~P4 | P3 | P1~~ P3 | P4 | P1
0 2 6 ~~12 17 27~~ 12 17 27

P	AT	BT	CT	TAT	WT	RT
A	0	~~10~~ ~~7~~ 6	20	20	10	
B	3	~~6~~ 2	10	7	1	
C	7	1	8	1	0	
D	8	3	13	5	12	

GANTT CHART:

~~| A | B | C | B | A | D | A |~~
~~0 3 4 5 7 8 10 11 17~~

| A | B | C | B | D | A |
0 3 7 8 10 13 20

After arriving of each job we compare & swap each process.
Thus sort by AT to easy doing.

NP - PRIORITY SHEDULING [NPPS]

MODE: NON - Preemptive
CRITERIA: Highest priority no.

Generally lower P. no has higher priority.

P	BT	P
1	10	3
2	1	1
3	2	4
4	1	5
5	5	2

GANTT

P2	P5	P1	P3	P4
0	1	6	16	18	19

P	AT	BT	P
1	0	3	2
2	2	5	6
3	1	4	3
4	4	2	5
5	6	9	7
6	5	4	4
7	7	10	10

P1	P3	P5	P4	P2	P
0	3	6	10	12	17

P	AT	BT	P
1	0	3	2
3	1	4	3
2	2	5	6
4	4	2	5
6	5	4	4
5	6	9	7
7	7	10	10

P1	P3	P6	P4	P2	P5	P7
0	3	7	11	13	18	20	27

PROBLEM:

* Indefinite blocking / Starvation.
* A process that is ready to run but blocked by CPU.
* Here some low priority process maybe left waiting indefinitely.
* In a heavily loaded process a stream of HP process can prevent a LP from ever getting shedulled.

SOLUTION:

* Aging.
* This increases the priority (Low priority) of those process that are waiting for a long time.

PPS

PID	AT	BT	P
P1	0	11 9	2
P2	5	20	0
P3	12	2	3
P4	2	10 7	1
P5	9	16	4

P1	P4	P2	P4	P1	P3	P5
0	2	5	25	32	41	44	48

When we see a process with higher priority we preempt that for the given AT.

PI	AT	BT	P
1	0	1	2
2	1	76	6
3	2	3	3
4	3	6	5
5	4	5	4
6	5	15	10
7	15	8	9

P1	P2	P3	P5	P4	P2	P7	P6
0	1	2	5	10	16	22	30	40	55

RR — ROUND-ROBIN

MODE : PREHPTIVE
CRITERIA : TQ & ready queue, AT
(modified FCFS)

TQ = 2

PID	AT	BT
P1	0	5 3 1
P2	1	4 2
P3	2	2
P4	4	1

GANTT CHART :

P1 P2 P3 P1 P4 P2 P1

P1	P2	P3	P1	P4	P2	P1
0	2	4	6	8	9	11	12

@t=0 RQ: P1
@t=2 RQ: P2 P3 P1
@t=4 RQ: P3 P1 P4 P2
@t=6 RQ: P1 P4 P2
@t=8 RQ: P4 P2 P1
@t=9 RQ: P2 P1
@t=11 RQ: P1
@t=12 RQ: ∅

NOTE that first the waiting process comes then only incomple process comes

PID	AT	BT
P1	0	5
P2	1	3
P3	2	1
P4	3	2
P5	4	8

P1 P2 P3 P1 P4 P5 P2 P1 P5

|P1|P2|P3|P1|P4|P5|P2|P1|P5|
0 2 4 5 7 9 11 12 13 14

#TQ = 2

@ t=0 : RQ : P1
@ t=2 : RQ : P2 P3 P1
@ t=4 : RQ : P3 P1 P4 P5 P2
@ t=5 : RQ : P1 P4 P5 P2
@ t=7 : RQ : P4 P5 P2 P1
@ t=9 : RQ : P5 P2 P1
@ t=11 : RQ : P2 P1 P5
@ t=12 : RQ : P1 P5
@ t=13 : RQ : P5
@ t=14 : RQ : ∅

When the TQ is large ⇒ act as FIFO.
TQ is small ⇒ TQ > S (context Switch) Else overhead is high. → not too large

TAT ↑ SJF, BUT AT is better.

ADVANTAGE:

fairness on each process.

Time sharing.

DISADVANTAGE:

Lot of Contex switches.

MULTILEVEL QUEUE SCHEDULING.

A class of scheduling algorithm are created for situations of process that can be classified into different queues.

* This portions the ready queue into several seperate queues.

* The process are grouped based on a certain criteria or idea.

Eg: Foreground - user interactive - faster - RR
Background - batch - can be slow - FCFS

* Classified into diff q based on
  → RESPONSE TIME
  → Scheduling needs.

* Scheduling is done within a q and also among the q.

* Among the q ⇒ Pre-emptive priority

Here FG > BG, priority wise. PPS

Process permenantley assigned to one queue, based on memory size, process size & priority.

* Each q has own scheduling algorithms.

Higher priority

→ System process →
→ Interactive process →
→ Interactive editing process →
→ Batch process →
→ Student process →

* "Each queue has absolute priority over lower-priority".

* We can also time-slice among the qs.
  80% - RR
  20% - FF.

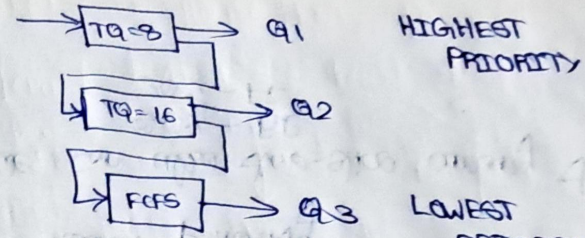
MULTILEVEL FEEDBACK QUEUE SCHEDULING.

Here also the same idea but little diff (Each Q with their own algo)

* This algo. allows a process to move between the queues.
* The idea is to separate the process on BT.
* If P use too much CPU time, it can be pushed down to a lower priority queue.
* If a P is too long in a low priority queue it can be pushed upwards.

~aging prevents STARVATION.

* I/O & interactive P are in Hihger priority queue.



* Pushed from Q1 to tail of Q2. → If P not completes.
They form feedback from each queue.

"Each Q has absolute priority over low-priority too." + PPS

PARAMETERS

1. No of Qs
2. Scheduling algo. of each Qs
3. When to upgrade/downgrade a process to which
4. The method used which Q we are moving the process to.

Scheduling Criteria

① Process (RUNNING → WAITING state), wait() used during I/O.

② Process (RUNNING → READY state), during interrupt.

③ Process (WAITING → READY state), completion I/O.

④ Process termination.

①, ④ → Non-preemptive / Cooperative.

②, ③ → Preemptive.

Dispatcher latency is the t taken by D to stop 1 & start 1 process

DISPATCHER

Dispatcher gives control to CPU to the process selected by SHORT-TERM scheduler.

↳ this selects from the processes that the short-term scheduler already choose.

Via 1) Context switch
2) User mode

MEMORY MANAGEMENT STRATERGIES

MODULE - 7

* Purpose of computer to execute programs ie) create processes.

* Program in exec must be in main memory (RAM)

⇒ Physical address ≡ Main memory ≡ RAM.

* Into are swapped btw 2° mem & MM.

Instruction - execution cycle
- Instruction fetching
- Instruction decoding
- Execution
- Memory access
- Feedback.

CPU can directly access → Registers in the processor, Main memory

* Machine instruction takes memory addr. as arguments.