GATE

Operating System

Intro

- OS is an interface b/w user and ~~hardware~~ hardware.
- Resource Allocator
- Manager → memory, processes, files, security etc.

Goals :-

- Primary → convenience
- Secondary → efficiency

Process requires two types of time –
- ↳ CPU time
- ↳ I/O time

Types of OS :-

- Batch OS → process will starve
- Multiprogramming
- Multitasking → process will not starve
- Multiprocessing → Preemption is allowed
- Real time → Parallelism → ∞ Multiple processors
  - ↳ Throughput is improved
  - ↳ Reliability through more CPU's
  - ↳ strict deadlines ~~of~~ for given jobs

01

Process Management

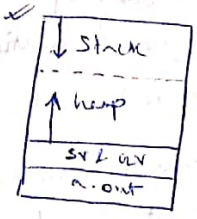
Attributes of a process :-

1) Process id
2) Program Counter
3) Process State
4) Priority
5) General purpose registers
6) List of open files
7) List of open devices
8) Protection

PCB's
- ↳ for each process
- ↳ connected as linked lists
- ↳ Holds all information ~~to~~ related to one process
- ↳ ~~It~~ It is also called context
- ↳ Everything about a process is a context

- A program resides in secondary memory generally. But subdivisions of a program is sent to main memory and get executed. called This subdivisions are process.
- Program is divided into several processes
- Static and Global variables running until the process is destroyed.
- Stack and heap are also there in memory.
  - ↓ Allocated to each process (function stack/recursion stack)
  - ↓ used by processes if needed

Ex.c → compiler → a.out



02

States of a process

1) New (secondary memory)
2) Ready (primary ")
3) Run (")
4) Block or wait (primary ")
5) Termination or completion
6) Suspend ready (secondary ")
7) Suspend wait or ~~suspend~~ suspend block. (secondary ")

Operations on process

1) Creation
2) Scheduling
3) Execution
4) Killing / Delete.

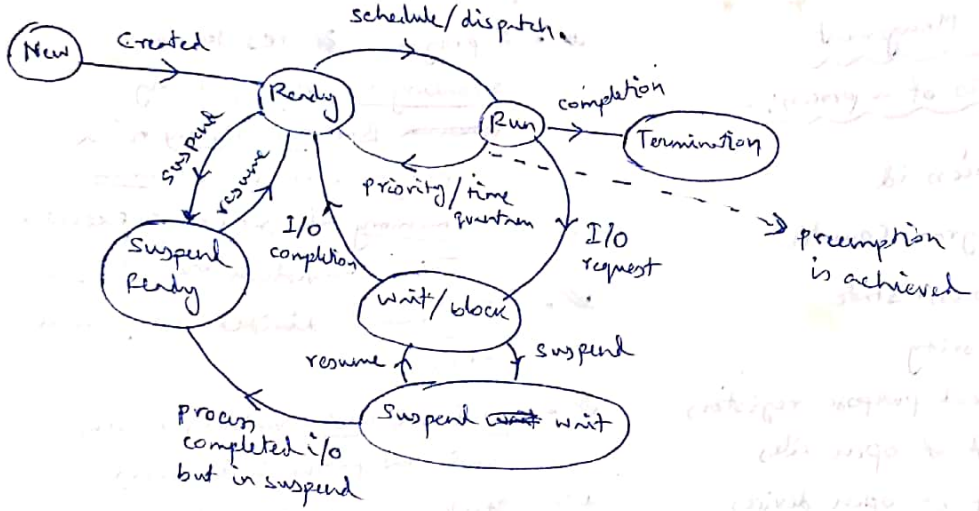
Multiprogramming

- With preemption → Also called → multitasking / time sharing
- Without preemption

- Context switching → switch between processes for a ~~quantum~~ time.
- A process's PCB is deleted after its termination.

03

Process State ~~Diag~~ Diagram



1) Long Term scheduler → No of process ~~created~~ creation — take decisions about
2) Short " " → Select processes from ready queue constantly and give it to processor
3) Medium " " → Take suspension decision of processes

Call Dispatcher → check priority or time quantum

- Degree of multiprogramming is no of processes that can be present in ready state at maximum.
- Long term scheduler have important impact on the performance of a system.
  - It should choose a ~~mix~~ mix of CPU & I/O bound process.
- Context switching → CS time can be very ~~less~~ less by ~~reduce~~ reducing the size of the PCB or context, so moving time will be less.

• Swapping ⇒ moving process from main memory to secondary memory and then secondary memory to main memory.

↳ Less swapping is good for performance, so good medium term scheduler should be chosen.

04

Consider a system with 'N' CPU processors and 'M' processes, then

	min	max
ready	0	M
running	0	N
block	0	M

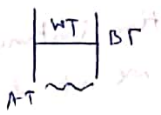
} → max no. of processes can present

↳ If all I/O

05

Important parameters of processes

1) Arrival time
2) Burst time
3) Completion time
4) Turn around time → (CT − AT) / (BT + WT)
5) Waiting time → (TAT − BT)
6) Response time →



• In real time, it is not possible to know the burst time of a process.

06

CPU Scheduling

- Who → Short term Scheduler
- Where → Ready state to running
- When → when a process moves from
  - i) Run → Termination
  - Run → Wait
  - Run → Ready
  - ii) New → Ready i.e when a process is just created
  - iii) Wait → Ready

07

FCFS :-

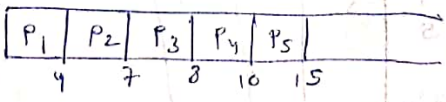
Criteria : Arrival time

mode : Non-preemptive

WT	TAT
0	4
3	6
5	6
5	7
6	11

PNO	AT	BT	CT
1	0	4	4
2	1	3	7
3	2	1	8
4	3	2	10
5	4	5	15

GANTT Chart



Avg TAT = $\frac{4+6+6+7+11}{5} = \frac{34}{5} = 6.8$

Avg WT = $\frac{0+3+5+5+6}{5} = \frac{19}{5} = 3.8$

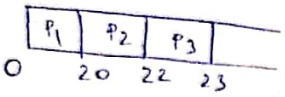
In non-preemptive scheduling, response time = waiting time

- FCFS has convoy effect.

08

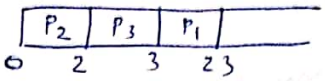
Convoy effect of FCFS

PNo	AT	BT	CT	TAT	WT
1	0	20	20	20	0
2	1	2	22	21	19
3	1	1	23	22	21



Avg WT = $\frac{0+19+21}{3} = \frac{40}{3}$

PNO	AT	BT	CT	TAT	WT
1	1	20	23	22	2
2	0	2	2	2	0
3	0	1	3	3	2



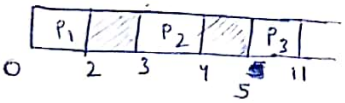
Avg WT = $\frac{2+0+2}{3} = \frac{4}{3}$

09

FCFS example

PNO	AT	BT
1	0	2
2	3	1
3	5	6

- Gaps are created because at that time instance, remaining processes are not at ready queue, so have to wait until their AT comes.



CT	TAT	WT
2	2	0
4	1	0
11	6	0

Avg TAT = $\frac{2+1+6}{3} = 3$

Avg WT = $\frac{0+0+0}{3} = 0$

- Implement by queue data structure
- Complexity → O(n)

10 → Context switch time $\delta = 1$ unit FCFS with overhead

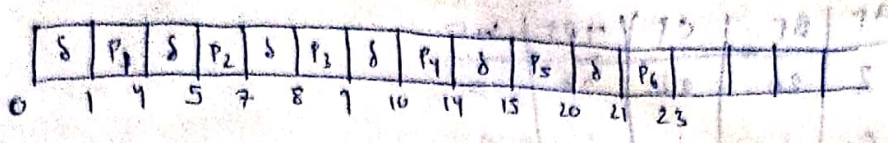
PNO	AT	BT	CT	TAT	WT
1	0	3			
2	1	2			
3	2	1			
4	3	4			
5	4	5			
6	5	2			

Ready → Scheduler → Dispatcher → process

Running

TAT

WT



Schedule time → 23

Overhead → 6

✓ Inefficiency → $\left(\frac{6}{23}\right) \times 100$

$= 0.26 \times 100 = 26\%$

✓ Efficiency → $\eta = \left(1 - \frac{6}{23}\right) \times 100$

$= 0.74 \times 100 = 74\%$

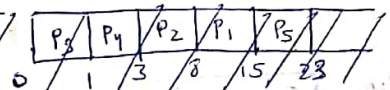
✓ 11

SJF (Short Job First) :-

Criteria – Burst time

mode – Non preemptive

CT	PNO	AT	BT	TAT	BT
15	1	1	7	14	
8	2	2	5	6	
4	3	3	1		
3	4	4	2		
23	5	5	8		



✓

PNO	AT	BT	CT	TAT	WT
1	1	7	8	7	0
2	2	5	16	14	9
3	3	1	9	6	5
4	4	2	11	7	5
5	5	8	24	19	11

✓ Avg TAT

$= \frac{7+14+6+7+19}{5}$

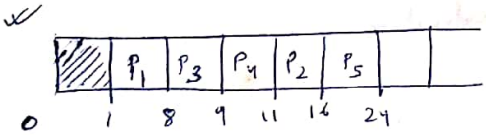
$= \frac{53}{5} = 10.6$

Avg WT

$= \frac{0+9+5+5+11}{5}$

$= 6$

✓

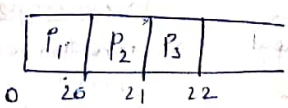


✓ 12

Analysis of SJF :-

✓ • SJF convoy effect

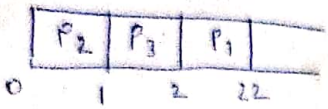
PNO	AT	BT	CT	TAT	WT
1	0	20	20	20	0
2	1	1	21	20	19
3	2	1	22	20	19



Avg WT – $\frac{0+19+19}{3} = \frac{38}{3}$

$= 12.66$

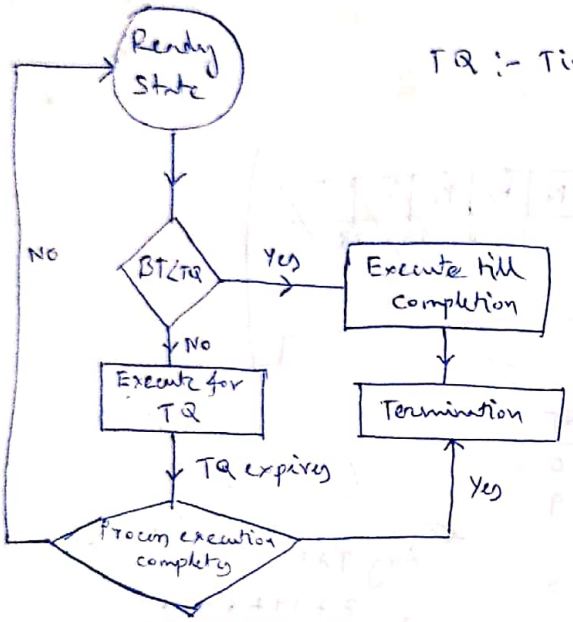
PNo	AT	BT	CT	TAT	WT
1	2	20	22	20	0
2	0	1	1	1	0
3	1	1	2	1	0



Throughput = $\frac{3}{22}$

13

Round Robin Algorithm



TQ :- Time Quantum

- Practically implementable, because not dependent on burst time.
- It only does required a queue data structure.
- No starvation.

14

SJF with prediction of BT

SJF

Advantages
- Maximum throughput
- Minimum Avg WT and TAT

Disadvantages
- Starvation to longer jobs
- It is not implementable because BT of processes cannot be known ahead.

Solution :-
SJF with a predicted BT.

Prediction Techniques

- Static
  1) Proam size
  2) Proam Type
- Dynamic
  1) Simple averaging
  2) Exponential averaging / ~~Aging~~ Ageing.

Proam Size (Bytes)

P_{old} = 200 KB ⇒ 20 units

P_{new} = 201 KB ⇒ 20 units

Proam Type

Proam
- OS ⇒ (3-5 units)
- User
  - interactive (5-8)
  - fore ground (10-15)
  - Back ground (15-20)

simple average

→ Given n processes $(P_1, \ldots P_n)$

→ Let t_i be the actual BT.

→ Let τ_i denote the predicted BT

$$\tau_{n+1} = \frac{1}{n}\sum_{i=1}^{n} t_i$$

Exponential averaging / Ageing

→ previous n processes

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

α : smoothening factor

$0 \le \alpha \le 1$

$$\tau_n = \alpha t_{n-1} + (1-\alpha)\tau_{n-1} \quad \text{--- ii)}$$

ii) into i)

$$\tau_{n+1} = \alpha t_n + (1-\alpha)\alpha t_{n-1} + (1-\alpha)^2 \tau_{n-1}$$

↳ ... keep expanding

$$\tau_5 = \alpha t_4 + (1-\alpha)\tau_4$$

$$= (0.5)(7) + (0.5)(6.75)$$

$$= 6.875$$

$$\alpha t_n + (1-\alpha)\alpha t_{n-1} + (1-\alpha)^2 \alpha t_{n-2} + (1-\alpha)^3 \tau_{n-2}$$

Exercise –

$\alpha = 0.5$, $\tau_1 = 10$,

actual BT (t_1, t_2, t_3, t_4) ⇓ (4, 8, 6, 7)

then τ_5 = ?

$$\tau_2 = \alpha t_1 + (1-\alpha)\tau_1$$

$$= 0.5(4) + 0.5(10) = 7$$

$$\tau_3 = \alpha t_2 + (1-\alpha)\tau_2$$

$$= 0.5(8) + 0.5(7)$$

$$= 7.5$$

$$\tau_4 = \alpha t_3 + (1-\alpha)\tau_3$$

$$= 0.5(6) + (0.5)\,7.5$$

$$= 6.75$$

is

Round Robin

TQ = 2

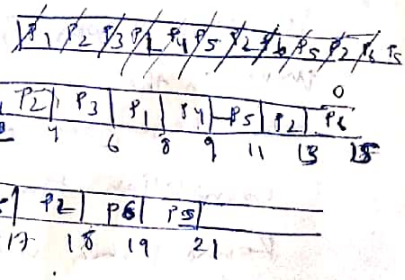
- TQ + AT
- preemptive

PNo	AT	BT	
1	0	~~4~~	✗ 0
2	1	~~8~~	✗ ✗ 0
3	2	~~2~~	0
4	3	~~1~~	0
5	4	~~6~~	4 ✗ 0
6	6	~~3~~	1 0

CT	TAT	WT
8	8	4
18	17	12
6	4	2
9	6	3
21	17	4
19	13	10

Avg TAT –

Avg WT –



• If TQ increases, then the context switching is decreased.
  ↳ problem → starvation
  ↳ Decreases → more context switching

• TQ should be neither too big nor too small.

✓ TQ = 4

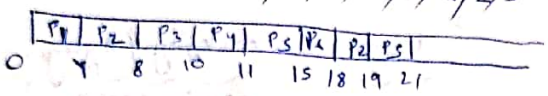
PNO	AT	BT
1	0	4 0
2	1	8 4 0
3	2	2 0
4	3	1 0
5	4	6 2 0
6	5	3 0

CT	TAT	WT
4	4	0
19	18	13
10	8	4
11	17	7
21	13	11
18	13	10

Context switching are less than previous Because of TQ.

← • Starvation occurs (P6)

P1 P2 P3 P4 P5 P1 P2 P5



✓ 16

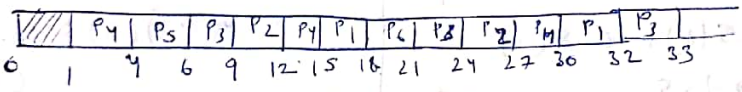
✓ Round Robin example:- TQ = 3

PNO	AT	BT
1	5	5 2 0
2	4	6 3 0
3	3	7 4 1 0
4	1	9 6 3 0
5	2	2 0
6	6	3 0

CT	TAT	WT	RT	
32	27	22	10	15-5
27	23	17	5	9-4
33	30	23	3	6-3
30	29	20	0	1-1
6	4	2	2	4-2
21	15	12	12	18-6

↗ First occurrence - AT

P4 P5 P3 P2 P4 P1 P6 P3 P2 P4 P1 P3



✓ a. TQ ↑ CS ↓ RT ↑ (Better CS)

✓ • TQ ↓ CS ↑ RT ↓ (Better RT)

✓ • First add processes from list by checking the arrival time, then add the current process if its execution is not over (queue insertion).

✓ If TQ = ∞, then RR → FCFS

✓ 17

Consider 4 jobs P_1, P_2, P_3 and P_4 arriving in ready queue in the same order at time = 0. If BT requirements of these jobs are 4, 1, 8, 1 respectively, what is completion time of P_1, assuming Round Robin with TQ = 1

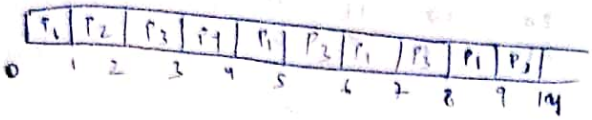
P_1 ~~X X X~~ 10

P_2 1 6

P_3 8 ~~X X~~ 5

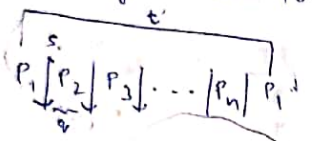
P_4 1 0 CT of P_1 = 7

P_1 P_2 P_3 P_4 P_1 P_3 P_1 P_3 P_1 P_3



18

Consider 'n' processes sharing the CPU in RR fashion. If the context switch time is 's' units. What must be the time quantum 'q' such that the no of context switches are reduced, but at the same time each process is guaranteed to get the turn of the CPU for every 't' seconds.



Let, $P_1 | P_2 | P_3 | P_4 | P_1$

4 context switches for 4 processes

$n.(s) + (n-1).q \leq t$

$$q \leq \frac{t - ns}{(n-1)}$$

19

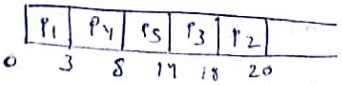
Longest Job First Algorithm

Process having longest BT gets scheduled first.

Criteria :- BT

Mode :- Non-preemptive

PNO	AT	BT	CT	TAT	WT	RT
1	0	3	3	3	0	0
2	1	2	20	19	17	17
3	2	4	18	16	12	12
4	3	5	8	9	0	0
5	4	6	14	10	4	4

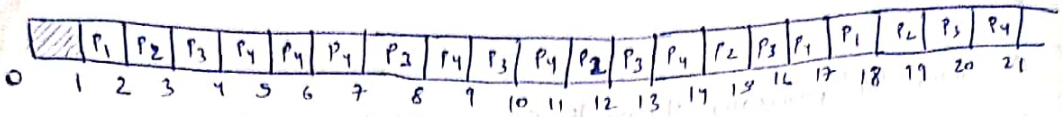


20

Longest Remaining Time First

TQ = 1

PNo	AT	BT	CT	TAT	WT	RT
1	1	2 1 0	18	17	15	0
2	2	4 3 2 1 0	19	17	13	0
3	3	6 5 4 3 2 1 0	20	17	11	0
4	4	8 7 6 5 4 3 2 1 0	21	17	9	0

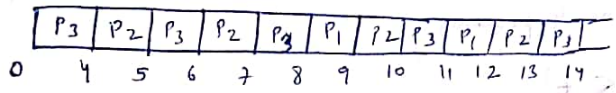


21

Gate 2006:-

avg TAT using LRTF = ?

PNo	AT	BT	CT	TAT	WT	RT
1	0	2 1 0	12	12	10	8
2	0	4 3 2 1 0	13	13	9	4
3	0	8 7 6 5 4 3 2 1 0	14	14	6	0



Avg TAT

$= \frac{12+13+14}{3}$

$= 13$

22

Highest Response Ratio Next (HRRN)

Criteria :- Response ratio (RR) = $\frac{W+S}{S}$

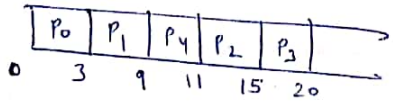
- W : waiting time for a process so far
- S : service time of a process or BT.

-> HRRN not only favours shorter jobs but also limits the waiting time of longer jobs.

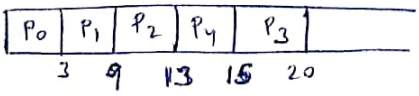
-> mode :- non-preemptive

PNO	AT	BT
0	0	3
1	2	6
2	4	4
3	6	5
4	8	2

SJF



HRRN



$RR_2 = \frac{(9-4)+4}{4} = \frac{5+4}{4} = \frac{9}{4} = 2.25$

$RR_3 = \frac{(9-6)+5}{5} = \frac{3+5}{5} = \frac{8}{5} = 1.6$

$RR_4 = \frac{(9-8)+2}{2} = \frac{1+2}{2} = \frac{3}{2} = 1.5$

$RR_3 = \frac{(13-6)+5}{5} = \frac{12}{5} = 2.4$

$RR_4 = \frac{(13-8)+2}{2} = \frac{7}{2} = 3.5$

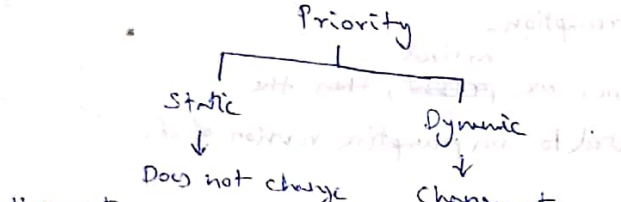
HRRN

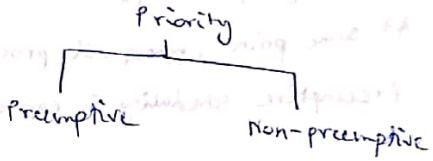
CT	TAT	WT	RT
3	3	0	0
7	7	1	1
13	9	5	5
20	14	9	9
15	7	5	5

- Disadvantage → Cannot be implemented practically because BT can be is not known.
- Round Robin is most favourable
- HRRN is better than SJF, theoretically.

23

Priority Scheduling





- If two processes are having the same priority, AT is going to break the tie. If AT is also same then the process with the lower id gets the priority.

24

Priority Scheduling (Non-preemptive)

PNo	Priority	AT	BT	CT	TAT	NT	RT
1	(L) 2	0	4	4	4	0	0
2	4	1	2	25	24	22	22
3	6	2	3	23	21	18	18
4	10	3	5	9	6	1	1
5	8	4	1	20	16	15	15
6	12 (H)	5	4	13	8	4	4
7	9	6	6	19	13	7	7

P_1	P_4	P_6	P_7	P_5	P_3	P_2		

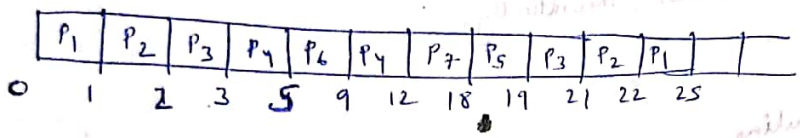
0 4 9 13 19 20 23 25

25

Priority Scheduling (Preemptive)

Throughput = $\frac{7}{29}$ = $\frac{\text{(No of processes)}}{\text{(Total scheduling time)}}$

PNo	Priority	AT	BT	CT	TAT	WT	RT
1	2	0	4	25	25	21	0
2	4	1	2	22	21	19	0
3	6	2	3	21	19	16	0
4	10	3	5	12	9	4	0
5	8	4	1	19	15	14	14
6	12	5	4	9	4	0	0
7	9	6	6	18	12	6	6

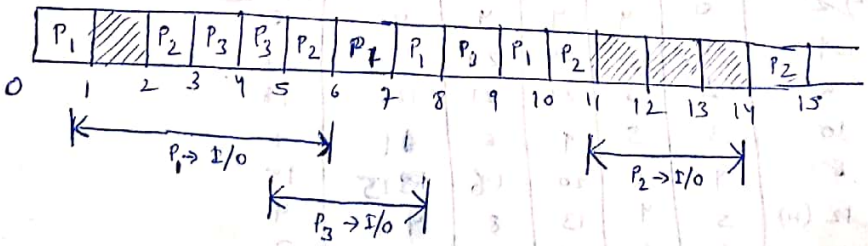


- A process can run until ~~another~~ the AT of another process with high priority comes, i.e in preemption.
- At some point, when all process are ~~ready~~ arrived, then the preemptive scheduling is converted to non preemptive version of it.

26

Preemptive Priority with processes containing CPU and I/O time

PNO	AT	Priority	CPU	I/O	CPU	CT	TAT	WT	RT
P_1	0	2	1	5	4	10	10	6	0
P_2	2	3 (L)	1	3	1	15	13	9	0
P_3	3	1 (H)	2	3	1	9	6	3	0



Inefficiency = $\frac{4}{15} \times 100 = 26.67\%$

Efficiency = $\frac{11}{15} \times 100 = 73.33\%$

27

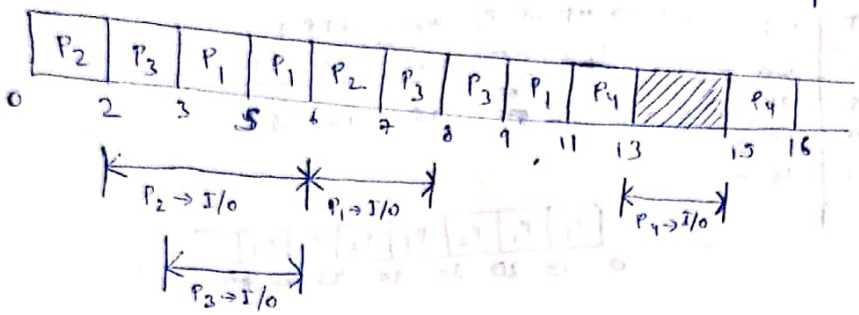
SRTF with process containing CPU and I/O time example :

PNO	AT	BT	IOBT	RT	CT	TAT	WT	RT
1	0	(3 1 0	2	2 0)	11	11	6	3
2	0	(2 0	4	2 0)	7	7	4	0
3	2	(1 0	3	1 0)	9	7	4	0
4	5	(2 0	2	1 0)	16	11	8	6

Actual

BT	IOBT	BT
3	2	2
2	4	1
1	3	2
2	2	1

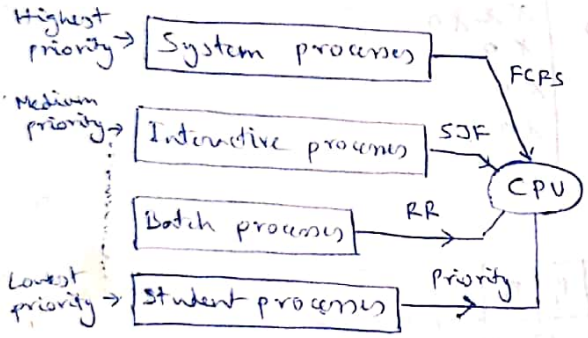
↓ ignore this in computation and add 1st and 2nd BT.



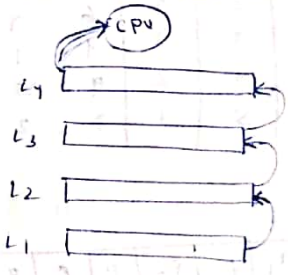
28

Multilevel queues and Multilevel feedback queues

Multilevel queue scheduling



ML Feedback ~~queue~~ scheduling



↓

- Avoid starvation completely by the concept of aging.

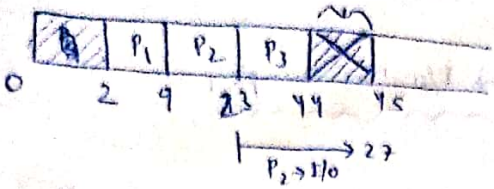
29

example 2 (27) - SRTF

- Three processes arriving at time zero with total ~~exact~~ execution time of 10, 20, and 30 units respectively. Each process spends the first 20% of execution time doing I/O, 70% time doing computation and last 10% time doing again I/O. Compute the % of idle time for SRTF.

↓ CPU

	AT	BT	IO	CPU	IO
P_1	0	10	2	7	1
P_2	0	20	4	14	2
P_3	0	30	6	21	3



Inefficiency → $\frac{2}{44} \times 100 = 4.54$

Efficiency → $\frac{42}{44} \times 100 = 95.46$

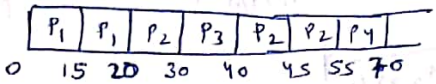
30

SRTF example - Code 2007

PNo	AT	BT
1	0	~~20~~ ~~8~~ 0
2	15	~~25~~ ~~15~~ ~~10~~ 0
3	20	~~10~~ 0
4	45	15 0

→ WT of P_2 using SRTF ?

P_2 →	CT	TAT	WT	RT
	~~55~~	40	15	5

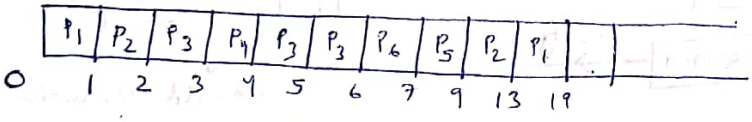


31

SRTF

~~Note : Preemptive~~ ∴ If TQ is not given, take it by default as 1 until all processes are arrived.

PNo	AT	BT	CT	TAT	WT	RT	
1	0	7	19	19	12	0	~~7~~ ~~6~~ 0
2	1	5	13	12	7	0	~~5~~ ~~4~~ 0
3	2	3	6	4	1	0	~~3~~ ~~2~~ ~~1~~ 0
4	3	1	4	1	0	0	~~1~~ 0
5	4	2	9	5	3	3	~~2~~ 0
6	5	1	7	2	1	1	~~1~~ 0



~~P1~~, ~~P2~~, ~~P3~~, ~~P4~~, ~~P5~~, ~~P6~~

∴ If at any moment, BT of two processes are same, then choose the one with lower AT.

preemptive → non-preemptive after all processes are arrived.

Cannot be implemented practically because of dependency on burst time.

32

SRTF example – Gate 2014

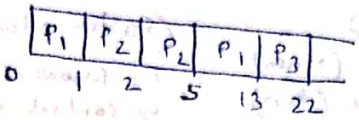
PNo	AT	BT
1	0	9 8 0
2	1	4 3 0
3.	2	9 0

→ Avg WT using SRTF :

CT	TAT	WT
13	13	4
5	4	0
22	20	11

Avg WT = $\frac{4+0+11}{3}$

= $\frac{15}{3}$ = 5



Important short notes

1) Next CPU burst is predicted is given by :-

$\tau_n = \alpha \times t_n + (1-\alpha) \times \tau_{n-1}$; $0 \le \alpha \le 1$

t_n is current CPU time,

τ_{n-1} is the past predicted value, and

τ_n is the new predicted value.

2) For n fork() calls, 2^{n-1} child processes will be created.

3) Round Robin Algorithm :

- n processes in ready queue, q is time quantum, then each process gets 1/n of the CPU time in chunks of at most q time units.
- Each process must wait no longer than (n-1) q time units until next time quantum.
- TAT also depends on the size of the time quantum.

4) If waiting time or fraction of each process is P and n is the no. of processes, then

- CPU utilization = $1 - p^n$
- And probability that N processes will wait at the same time = p^N

5) Unix Inode :

- Suppose there are 12 direct blocks and 1 indirect block and 1 double indirect block and 1 triple indirect block then the maximum size of process supported by inode is :

$$= \left[12 + \frac{BS}{ES} + \left(\frac{BS}{ES}\right)^2 + \left(\frac{BS}{ES}\right)^3\right] \times BS \text{ ; where}$$

BS = Block size; ES = entry size (or block pointer size).