CPU Scheduling

Arrival time :- The time at which a process enters into the ready state.

Burst :- the amount of CPU time required by a process to finish. Predicting this time is tricky.

Completion time :- The time at which the process finishes.

Turn around time :- Difference b/w arrival time and completion time is turnaround time.

waiting time :- time for which a process waits in the ready queue.

Response time - diff. b/w the arrival time and time at which the process is scheduled for 1st time.

Page No.:
Date: / /

CPU Scheduling

Picking a process from ready state and giving it to CPU is CPU scheduling.

Short term scheduler does CPU scheduling. (STS)

Sometime, STS just picks the process but taking it to the CPU (i.e., content switch) is done by dispatcher.

When scheduling is done

(1) Whenever a CPU is empty, the STS will bring a process to CPU, i.e., running state.

A CPU will become empty when
- → The running process terminated
- → The running process gone to waiting state
- → The running process move to ready state because of time-slice expired or preempted

(2) When a process comes from new to ready state and CPU is empty or its priority is high

(3) When a high priority process comes from waiting to ready state, it must have to immediately scheduled.

Page No.:
Date: / /

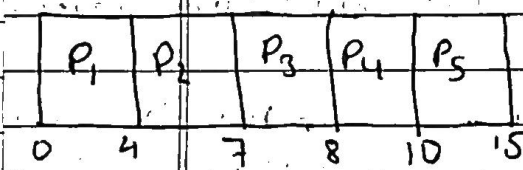
FCFS (First come First Served)

→ The process which arrived first in the ready state will get the CPU first.

→ The algorithm is non-Preemptive i.e. once a process gets CPU, it cannot be removed until it not finishes its execution.

→ Assuming that only one CPU and no I/O state.

P NO.	AT	BT	CT	TAT	WT (TAT = WT + BT)
1	0	4	4	4	0
2	1	3	7	6	3
3	2	1	8	6	5
4	3	2	10	7	5
5	4	5	15	11	6

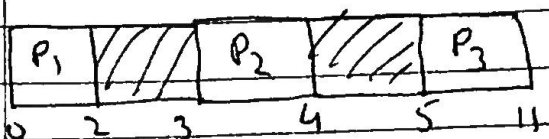


Response time and waiting time is same in case of non-preemptive scheduling.

Av. TAT = (4+6+6+7+11)/5

Av. WT = (0+3+5+5+6)/5

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

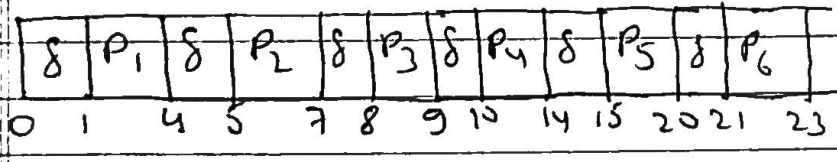


CT	TAT	WT
2	2	0
4	1	0
11	6	0

=)

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

Context switch time = 1



$$\text{efficiency} = \frac{17}{23} \times 100$$

Page No.:
Date: / /

Shortest Job first

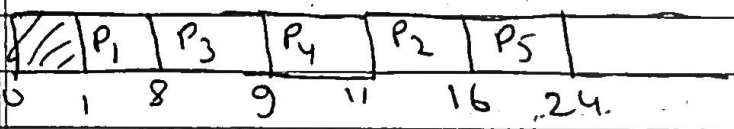
→ The process which have least burst time will be scheduled first.

→ It has two mode: Preemptive and non-Preemptive.

→ Here we consider non-preemptive.

→ Assuming only one CPU and no waiting state.

PID	AT	BT
1	1	7
2	2	5
3	3	1
4	4	2
5	5	8



CT	TAT	WT
8	7	0
16	14	9
9	6	5
11	7	5
24	19	11

Page No.:
Date: / /

SJF

Advantages: (i) maximum throughput
(ii) minimum av. WT and TAT

Disadvantages:
(i) Starvation to longer jobs
(ii) SJF is not implementable bcoz BT of processes can not be known initially.

Solⁿ:- SJF with predicted BT.

Prediction techniques

(A) Static – (i) Process size
if P_{old} = 200 KB and it taken 20 units of time then it is possible that if the new process P_{new} = 201 KB then its execution time = 20 units.

(ii) Process type: Each type of processes may have different CPU burst times.
e.g. OS processes, interactive processes, foreground processes, background processes.

(b) Dynamic

(i) Simple averaging:

Page No.:
Date: / /

Let there are n processes.
Actual BT of the ith process is t_i

Let T_i denotes the predicted BT of the ith process then

$$T_{K+1} = \frac{1}{K} \sum_{i=1}^{K} t_i$$

(ii) exponential averaging:-

$$T_{K+1} = \alpha t_K + (1-\alpha) T_K \quad 0 < \alpha \leq 1 \quad -(1)$$

$$T_K = \alpha t_{K-1} + (1-\alpha) T_{K-1} \quad -(2)$$

$$T_{K+1} = \alpha t_K + (1-\alpha)\alpha t_{K-1} + (1-\alpha)^2 T_{K-1}$$
$$= \alpha t_K + (1-\alpha)\alpha t_{K-1} + (1-\alpha)^2 \alpha t_{K-2} + (1-\alpha)^3 T_{K-2}$$
$$= - - - - -$$

By substituting the values, finally T_{K+1} will depend on t_1 to t_K values and T_1.

α is called smoothening factor.
The value of α decides the weightage given to previous process (t_K) and remaining history of prediction (T_K).

if α is high means we want to give high weightage to last process. i.e. next process prediction time is similar to last process actual running time.

If α is low, means more weightage is given to ~~next~~ prediction time of 1st process.

e.g. $\alpha = 0.5$, $T_1 = 10$

actual BT for (t_1, t_2, t_3, t_4) is $(4, 8, 6, 7)$

then $T_5 = ?$

$T_2 = \alpha t_1 + (1-\alpha) T_1 = 7$

$T_3 = \alpha t_2 + (1-\alpha) T_2 = 7.5$

$T_4 = \alpha t_3 + (1-\alpha) T_3 = 6.75$

$T_5 = \alpha t_4 + (1-\alpha) T_4 = 6.875$.

if α is high means we want to give high weightage to last process i.e. next process prediction time is similar to last process actual running time.

If α is low, means more weightage is given to ~~next~~ prediction time of 1st process.

e.g. $\alpha = 0.5$, $T_1 = 10$

actual BT for (t_1, t_2, t_3, t_4) is $(4, 8, 6, 7)$

then $T_5 = ?$

$T_2 = \alpha t_1 + (1-\alpha) T_1 = 7$

$T_3 = \alpha t_2 + (1-\alpha) T_2 = 7.5$

$T_4 = \alpha t_3 + (1-\alpha) T_3 = 6.75$

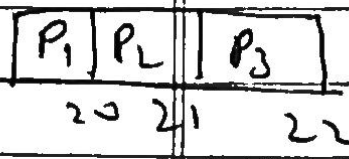
$T_5 = \alpha t_4 + (1-\alpha) T_4 = 6.875$

$\alpha = 0.5$ $T_1 = 10$

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

Page No. ____
Date __/__/__

PNO	AT	BT	CT	TAT	WT
1	0	20	20	20	0
2	1	1	~~21~~	~~20~~	~~19~~
3	2	1	23	21	20

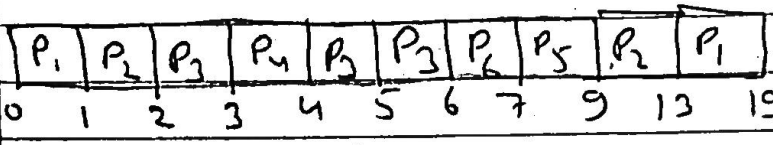


If AT of P_1 becomes 3 then av. waiting time is 0.

Page No.:
Date: / /

Shortest Remaining time first

PNo	AT	BT	CT	TAT (CT-AT)	WT (TAT-BT)
1	0	7	19	19	12
2	1	5	13	12	7
3	2	3	6	4	1
4	3	1	4	1	0
5	4	2	9	5	3
6	5	1	7	2	1



→ After each unit of time, it need to be checked that whether a new process came with lower BT.

→ In case of same BT, lower AT process get scheduled.

→ once all processes arrived, it works as SJF

After 1 unit, P_1 6, P_2 5

2 P_1 6 P_2 4 P_3 3

3 P_1 6 P_2 4 P_3 2 P_4 1

4 P_1 6 P_2 4 P_3 2 P_5 2

5 P_1 6 P_2 4 P_3 1 P_5 2 P_6 1

6 P_1 6 P_2 4 P_5 2 P_6 1

~~P_1 6 P_2~~

Now it became SJF

PNo	AT	BT
1	0	20
2	15	25
3	30	10
4	45	15

After 15 unit, P_1 5 P_2 25

20 P_2 25

30 P_2 15 P_3 10

40 P_2 15

45 P_2 10 P_4 15

Now SJF

=)

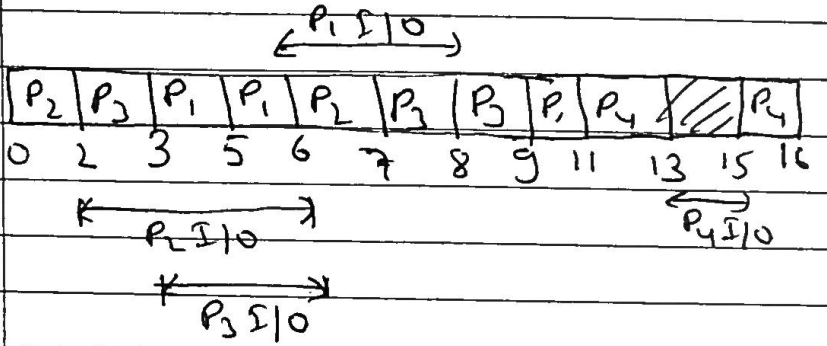
PNO	AT	BT
1	0	9
2	1	4
3	2	9

Page No.:
Date: / /

Average WT using SRTF ?

=)

PNO	AT	BT	I∘BT	BT	CT	TAT	WT
1	0	3	2	2	11	11	6
2	0	2	4	1	7	7	4
3	2	1	3	3	9	7	4
4	5	2	2	1	16	11	8



After 2 unit, P_2 went for I/O, P_1 5, P_3 3
After 3 unit, P_2 is in I/O, P_3 went for I/O, P_1 5
5 , P_2 & P_3 in I/O, P_1 3, P_4 3
6 , P_1 went for I/O, ~~P_1 3~~, P_2 1, P_3 2, P_4 3
7 ~~P_1 3~~ P_3 2 P_4 3
P_1 in I/O,
8 P_1 2, P_3 1, P_4 3
9 P_1 2 P_4 3
11 P_4 3
13 P_4 went for I/O
15 P_4 1

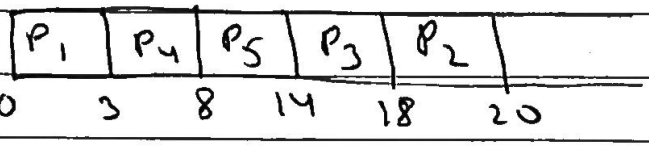
Page No.:
Date: / /

Longest Job First algorithm

Process having longest BT gets scheduled first.

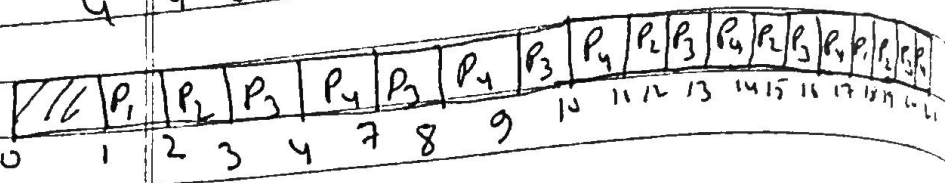
It is non-preemptive

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



Longest Remaining time first

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



at t=1, P_1 2

at t=2, P_1 1 P_2 4

at t=3, P_1 1 P_2 3 P_3 6

at t=4, P_1 1 P_2 3 P_3 5 P_4 8

at t=7, P_1 1 P_2 3 P_3 5 P_4 5

at t=8, P_3 4 P_4 5

t=9 P_3 4 P_4 4

t=10 P_3 3 P_4 4

t=11 P_3 3 P_4 3

t=12 P_1 1 P_2 2 P_3 3 P_4 3

t=13 P_1 1 P_2 2 P_3 2 P_4 3

t=14 P_1 1 P_2 2 P_3 2 P_4 2

t=15 P_1 1 P_2 1 P_3 2 P_4 2

t=16 P_1 1 P_2 1 P_3 1 P_4 2

t=17 P_1 1 P_2 1 P_3 1 P_4 1

t=18 P_2 1 P_3 1 P_4 1 P_1 finished

t=19 P_3 1 P_4 1 P_2 finished

t=20 P_4 1 P_3 finished

t=21 All Process finished

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

Page No.:____
Date:__/__/__

Here, no process is going to complete until longest process going to complete. Almost all processes finishes at same time.

=>

PID	AT	BT	CT	TAT
1	0	2	12	12
2	0	4	13	13
3	0	8	14	14

Page No
Date: /

Priority Scheduling

Static - Priority doesn't change throughout the execution of the process.

Dynamic:- changes at regular intervals of time.

Preemptive and non-Preemptive ~~Priority~~ Priority algorithm.

In priority scheduling, processes are scheduled based on their priority assigned by OS.

Non-Preemptive Priority Scheduling

Pno	Priority	AT	BT	CT	TAT	WT
1	2	0	4	4	4	0
2	4	1	2	25	24	22
3	6	2	3	23	21	18
4	10	3	5	9	6	1
5	8	4	1	20	16	15
6	12	5	4	13	8	4
7	9	6	6	19	13	7

2 is highest (lowest) Priority & 12 is lowest (highest) Priority

P_1	P_4	P_6	P_7	P_5	P_3	P_2

0 4 9 13 19 20 23 25

In case of non-Preemptive scheduling, response time is equal to waiting time.

Preemptive Priority Scheduling

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

P_1	P_2	P_3	P_4	P_6	P_4	P_7	P_5	P_3	P_2	P_1

0 1 2 3 5 9 12 18 19 21 22 25

at t=1 P_1 3 P_2 2 P_2 has highest Priority.

t=2 P_1 3 P_2 1 P_3 3 P_3 - - -

t=3 P_1 3 P_2 1 P_3 2 P_4 5 P_4 - - -

t=5 P_1 3 P_2 1 P_3 2 P_4 3 P_6 4 P_6 - - -

t=9 P_1 3 P_2 1 P_3 2 P_4 3 P_5 1 P_7 6 P_4 - - -

t=12 P_1 3 P_2 1 P_3 2 P_5 1 P_7 6 P_7 - - -

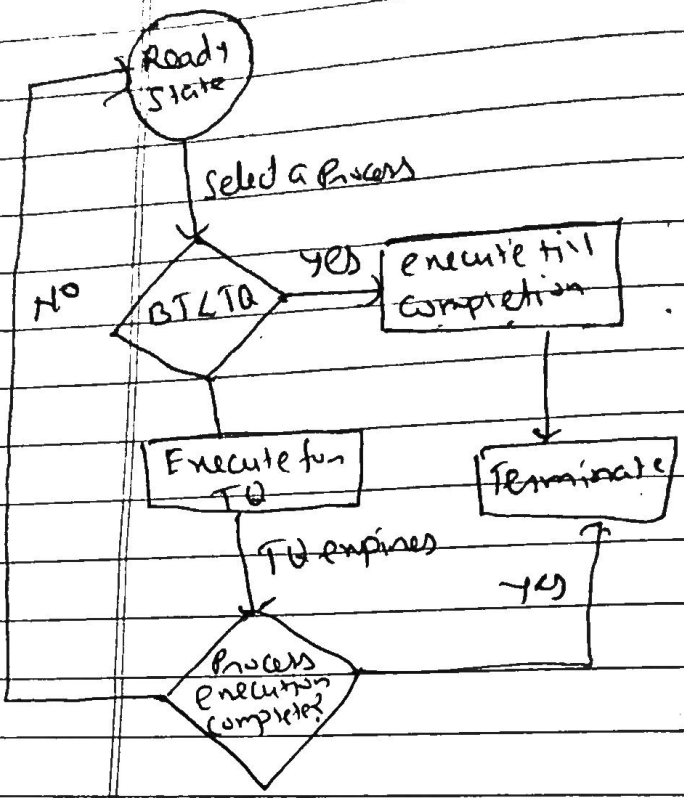
t=18 P_1 3 P_2 1 P_3 2 P_5 1 P_5 - - -

t=19 P_1 3 P_2 1 P_3 2 P_3 - - -

t=21 P_1 3 P_2 1 P_2 - - -

t=22 P_1 3 P_1 - - -

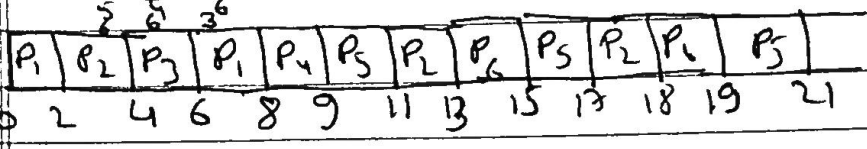
Round Robin Algo.



5 4 6
6 6 3

Time quantum (TQ) = 2

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



$P_1 P_2 P_3 P_1 P_4 P_5 P_2 P_6 P_5 P_2 P_1 P_5$

CT	TAT	WT
8	8	4
18	17	12
6	4	2
9	6	5
21	17	11
19	13	10

t=0, P_1 ● 4

t=2 P_1 ● 2 P_2 5 P_3 2 P_2 turn ($P_2 P_3 P_1$)

t=4 P_1 2 P_2 3 P_3 2 P_4 1 ~~P_3~~ P_5 6 P_3 turn ~~($P_2 P_3 P_1 P_4 P_5$)~~ ($P_3 P_1 P_4 P_5 P_2$)

~~t=6 P_1 2 P_2 3 P_4 1 P_5 6 P_6 3~~

t=6 P_1 2 P_2 3 P_4 1 P_5 6 P_6 3 P_1 turn ($P_1 P_4 P_5 P_2 P_6$)

t=8 P_2 3 P_4 1 P_5 6 P_6 3 P_4 turn ($P_4 P_5 P_2 P_6$)

t=9 P_2 3 P_5 6 P_6 3 P_5 turn ($P_5 P_2 P_6$)

t=11 P_2 3 P_5 4 P_6 3 P_2 turn ($P_2 P_6 P_5$)

t=13 P_2 1 P_5 4 P_6 3 P_6 turn ($P_6 P_5 P_2$)

t=15 P5 P2 1 P5 4 P6 1 (P5 turn (P5 R P6)

t=17 ~~P5 4 P6~~ P2 1 P5 2 P6 1 P2 turn (P2 P6 P5)

t=18 P5 2 P6 1 P1 turn P1 P5

t=19 P5 2 P5 turn P5

t=21

Page No.:____
Date:__/__/__

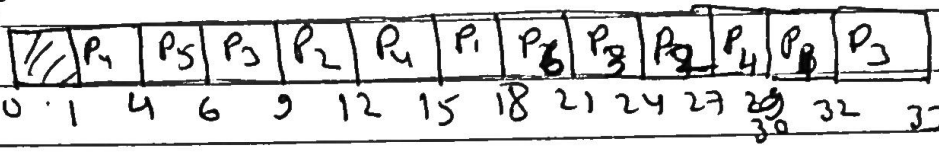
If TQ is less than Content switching will increase and TQ is high then starvation

Date: / /

If TQ is less than Content switching will increase and TQ is high then starvati and high RT.

⇒ TQ = 3

Pno	AT	BT
1	5	5
2	4	6
3	3	7
4	1	9
5	2	2
6	6	3



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

CT	WT	RT
32	22	10
27	17	5
33	23	3
30	20	0
6	2	2
21	12	12

Pno	AT	BT
1	0	4
2	0	1
3	0	8
4	0	1

TQ = 1

Page No.
Date:

Highest Response Ratio Next

Criteria: response ratio (RR) $= \frac{w+s}{s}$

w - waiting time for a process so far

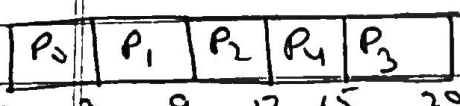
s - Service time of a process or BT

→ So, it depends on both waiting time as well as service time. So, if BT is less then RR will increase but in case of higher WT also RR will increase.

So, HRRN not only favors shorter jobs but also limits the WT of longer jobs.

→ Non-Preemptive

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



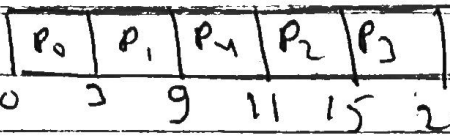
$RR_{2,9} = \frac{5+4}{4} = 2.25$ $RR_{3,9} = \frac{3+5}{5} = 1.6$

$RR_{4,9} = \frac{1+2}{2} = 1.5$ $RR_{3,13} = \frac{7+5}{5} = 2.4$

$RR_{4,13} = \frac{5+2}{2} = 3.5$

Page No.:
Date: / /

SJF

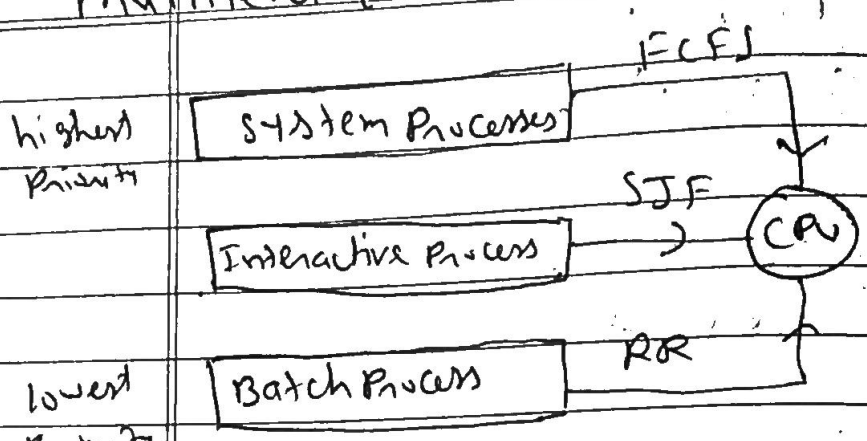


In SJF, P_2 (longest process) has to wait more than P_4 (shortest process)

HRRN is also not implementable bcoz it also needs BT in advanced.

CT	TAT	WT
3	0	0
7	1	1
9	5	5
14	9	9
7	5	5

Multilevel queue scheduling



Here, all the processes of highest priority queue will be served. So in this case processes in the low priority queue will starve.

Here, separate scheduling algo. can be applied on different queue.

As a soln, multilevel feedback queue algo. is impl. proposed in which a process in low priority queue can move to next high priority queue.

Process Scheduling

Linux Complete fair share scheduler (CFS) -
This is implemented in Linux kernel version 2.6.23 to the latest kernel version. It is based on rotating staircase deadline scheduler.

In today's kernel scheduler, there are 5 scheduling classes viz. Stop, Deadline, Realtime, CFS and Idle.
The scheduler iterates over each class in priority order starting with highest priority class. If a class has runnable process then it will run otherwise turn of a process from lower priority class comes. Stop is the highest priority class & Idle is the lowest priority class.

(i) Process of stop class can preempt everything & preempted by nothing. It is used by task migration, clock events etc.

(ii) Scheduling policy used in Deadline class is SCHED_DEADLINE. This class is used for periodic realtime task.

(iii) Realtime class :- It is used for POSIX realtime task. The scheduling policies used in this class are SCHED_FIFO and SCHED_RR
It runs the processes having priority values 0-99

(iv) CFS :- In this class, mostly user processes run. It runs process having priority values 100 to 139.
Scheduling policy used :
SCHED_NORMAL :- default for all user processes
SCHED_BATCH :-
SCHED_ISO - unused or reserved.
SCHED_IDLEPRIO :- low priority process

(v) Idle class:- It is lowest priority class.
Kernel idle threads are run when nothing else is runnable through the CPU.

$ ps -l $$
Check the priority & nice value

$ renice 7 $$
Check the priority & nice value.

$ sudo renice -20 $$
Check the priority & nice value

$ ~~renice 0 $$~~

$ watch free &

% Runs a command in every 2 second

$ ps -f pid_watch

Check the Priority & nice value

Since, watch is child of the shell, so, its priority & nice value will be same as parent, i.e. shell

$ renice 0 $$

$ kill -9 pid_watch

$ nice -7 sleep 100 &

$ ps -l pid_sleep

-7 means 7
- -7 = -7

$ kill -9 pid_sleep

System calls related to scheduling.

nice

sched.c int nice(int inc);

$ sudo nice -n -9 ./a.out

It will run the a.out with nice value -9

Hence o/p will be shown accordingly.

(i) nice system call changes the base priority of the calling process by adding a value to the nice value of the calling process

(ii) It affects only the process that invokes it.
It has been replaced with setpriority() system call.

=) int getpriority (int which, int who);

which :- PRIO_PROCESS :- Operates on the process whose PID equals who. If who is 0, use the caller's PID

PRIO_PGRP :- Operate on all the members of the process group whose PGID equal who. If who is 0, then use the caller's process group.

PRIO_USER :- Operate on all processes whose RUID equals who. If who is 0, use the caller's RUID.
RUID - real userid.