

Solutions of Selected Problems

P. 2: What is the difference between Non-preemptive and Preemptive scheduling..

Nonpreemptive: A process is in the Running state, it continues to execute until (a) it terminates or (b) blocks itself to wait for I/O or to request some operating system service.

Preemptive: The currently running process may be interrupted and moved to the Ready state by the operating system. The decision to preempt may be performed when a new process arrives, when an interrupt occurs that places a blocked process in the Ready state, or periodically based on a clock interrupt.

P. 3: Explain how a preemptive priority scheduling system would work.

In such a system, the highest priority process that is ready is run is always the one that is currently running. If a process becomes ready and has higher priority than the process currently running, then the current process is preempted and the higher priority process is allowed to run.

P. 6. Suppose a new process in a system arrives at an average of four processes per minute and each such process requires an average of 12 seconds of service time. Estimate the fraction of time the CPU is busy in a single processor system.

The fraction of time that a CPU is expected to be busy can be estimated if you know the arrival rate and service rate for all the processes. In this case, the arrival rate is 4 processes per minute (one process every 15 seconds), and the service rate is 12 seconds per process. Thus the fraction of time the CPU is expected to be busy is $(1/15)/(1/12) = 12/15 = 80\%$.

P. 7. Consider the following processes are to be scheduled using, FCFS, Round Robin with time quantum 1 and 4.

	A	B	C	D	E
T_a	0	1	3	9	12
T_s	3	5	2	5	5

Each square represents one time unit; the number in the square refers to the currently-running process.

FCFS

RR, $q = 1$

RR, $q = 4$

A	A	A	B	B	B	B	B	C	C	D	D	D	D	D	E	E	E	E	E
A	B	A	B	C	A	B	C	B	D	B	D	E	D	E	D	E	D	E	E
A	A	A	B	B	B	B	C	C	B	D	D	D	D	E	E	E	E	D	E

P. 8. Assume you have the following processes to execute with one CPU.

Process	Arrival Time	Execution Time
0	0	75
1	10	40
2	10	25
3	80	20
4	85	45

Suppose a system uses RR scheduling with a time quantum of 15 and context switch time is five time units with RR scheduling.

Create a Gantt chart illustrating the execution of these processes.

What is the turn around time for process 3.

```

Time    Process
0-15    p_0
15-20   context switch
20-35   p_1
35-40   context switch
40-55   p_2
55-60   context switch
60-75   p_0
75-80   context switch
80-95   p_1
95-100  context switch
100-110 p_2
110-115 context switch
115-130 p_3
130-135 context switch
135-150 p_4
150-155 context switch
155-170 p_0
170-175 context switch
175-185 p_1
185-190 context switch
190-195 p_3
195-200 context switch
200-215 p_4
215-220 context switch

220-235 p_0
235-240 context switch
240-255 p_4
255-260 context switch
260-275 p_0

```

p_3 turn around time = 115

P. 10. Consider a set of 5 aperiodic tasks with their execution profiles given below. Develop the scheduling diagram of these processes employing EDF and FCFS.

Process	Arrival Time	Execution Time	Starting Deadline
A	10	20	100
B	20	20	30
C	40	20	60
D	50	20	80
E	60	20	70

Each square represents 10 time units.

Earliest deadline

Earliest deadline with unforced idle times

FCFS

	A	A		C	C	E	E	D	D		
		B	B	C	C	E	E	D	D	A	A
	A	A		C	C	D	D				

P. 12. Consider three processes P, Q and S. P has a period of 100msec in which it requires 30msecs of processing. The corresponding values for Q and S are (6, 1) and (25, 5) respectively. Assume that P is the most important process in the system, followed by Q and then S.

- (1) What is the behavior of the scheduler if priority is based on importance?
- (2) What is the process utilization of P, Q and S.
- (3) How should the process be scheduled so that all the deadlines are met.
- (4) Illustrate one of the schemes that allows these processes to be scheduled.

(1) As P has the highest priority it will run first for 30 ms. Then Q will run for 1 ms; unfortunately it has missed its first deadline at 6ms, 12ms, 18, 24ms and 30ms. S will run last (after 31ms) but have missed its deadline at 25ms.

(2) Utility of P is 30%. Utility of Q is 16.67%. Utility of S is 20%. Total utility is 66.67%.

(3) Two approaches could be used. If scheduling is based on earliest deadline then the test is that total utilization is less than 100% priority model is used then the rate monotonic test could be applied. It will not be assumed that the general test can be remembered by the student, although the lower bound value of 69% should be. As total utilization is less than 69% the process set is scheduable. The rate monotonic scheme assigns priorities in an inverse relation to period length.

(4) For rate monotonic Q will have highest static priority, then S and then P. The execution sequence will be:

Process Execution-Time Total-Time

Q	1	1
S	5	6
Q	1	7
P	5	12
Q	1	13
P	5	18

Q	1	19
P	5	24
Q	1	25
S	5	30
Q	1	31
P	5	36
Q	1	37
P	5	42
Q	1	43
P	5	48
Q	1	49
idle	1	50

For earliest deadline the execution sequence will be the same up to the _rst idle time.

P. 13. Add a fourth process R, to the set of processes given in P 12. Failure of this process will not lead to safety being undermined. R has a period of 50msecs, but has a processing requirement that is data dependent and varies from 5 to 25 msecs. Discuss how this process should be integrated with P, Q and S.

At the minimum execution of R, its utility is 10% (Total now 76.67%). At the maximum execution of R utility is 50%, (Total now 116.67%). As R is not safety critical then it must miss its deadline (if any process must).

The earliest deadline scheme will not ensure this. The approach that should be taken is to use rate monotonic scheme and to transform P so that its period is less than R; i.e P becomes a process with a period of 10 and a requirement of 3ms (per period). With the new scheme Q will still have the highest static priority, then P, then S and lowest priority will go to R. The execution sequence will be:

Q	1	1
P	3	4
S	2	6
Q	1	7
S	3	10
P	2	12
Q	1	13
P	1	14
R	4	18
Q	1	19
R	1	20
P	3	23
R	1	24
Q	1	25
S	5	30
Q	1	31
P	3	34
R	2	36
Q	1	37
R	3	40
P	2	42

Q	1	43
P	1	44
R	4	48
Q	1	49
R	1	50

With this scheme S gets 16ms in its _rst period.

P. 14. Is the process set of Table below schedulable using the simple utilization based test. Is the process set schedulable using the response time analysis.

Process	Period	Execution Time
a	50	10
b	40	10
c	30	9

The task set is schedulable as

$$U(P1) = 0.2$$

$$U(P2) = 0.25$$

$$U(P3) = 0.3$$

Hence $U = 0.75$ which is below the threshold of 0.780 for three processes.

P. 15. The process set shown in Table 13.8 is not schedulable using the utilization criterion because a must be given the top priority due to its criticality. How can the process set be transformed so that it is schedulable. The computations represented by a must still be given top priority.

Process	Period	Execution Time	Criticality
a	60	10	HIGH
b	10	3	LOW
c	8	2	LOW

The task set is un-schedulable because priorities have been assigned that are not optimal. It must have period transformation applied to it. P1 is transformed to a task that has period 6 and computation time 1. RMS now gives P1 the highest priority (i.e. P1 is highest and the allocation is optimal).

The utilization of the task set is $.1666 + .333 + .25$, which is below the bound for schedulability. Hence system is OK.

P. 18. Real-time system designers wish to run a mixture of safety-critical, mission-critical and non-critical periodic and sporadic tasks on the same processor. They are using preemptive priority-based scheduling and have used the response-time analysis equation to predict that all tasks meet their deadlines.

Give reasons why the system might nevertheless fail to meet its deadlines at run-time.

What enhancement could be provided to the runtime support system to help eliminate the problem?

Basically, WCET and blocking time can be wrong, sporadic tasks may get invoked more often than anticipated, and the application periodic processes could compute the wrong delay value.

The RTS tools may be wrong.

RTS could handle the timing events for periodic tasks, and check that the sporadic tasks don't go off more often than anticipated.