CPU Scheduling

Practice Exercises

5.1 A CPU-scheduling algorithm determines an order for the execution of its scheduled processes. Given n processes to be scheduled on one processor, how many different schedules are possible? Give a formula in terms of n.

Answer:
$n!$ (n factorial $= n \times n - 1 \times n - 2 \times \ldots \times 2 \times 1$).

5.2 Explain the difference between preemptive and nonpreemptive scheduling.

Answer:
Preemptive scheduling allows a process to be interrupted in the midst of its execution, taking the CPU away and allocating it to another process. Nonpreemptive scheduling ensures that a process relinquishes control of the CPU only when it finishes with its current CPU burst.

5.3 Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use nonpreemptive scheduling, and base all decisions on the information you have at the time the decision must be made.

Process	Arrival Time	Burst Time
P_1	0.0	8
P_2	0.4	4
P_3	1.0	1

a. What is the average turnaround time for these processes with the FCFS scheduling algorithm?

b. What is the average turnaround time for these processes with the SJF scheduling algorithm?

c. The SJF algorithm is supposed to improve performance, but notice that we chose to run process P_1 at time 0 because we did not know

that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first 1 unit and then SJF scheduling is used. Remember that processes P_1 and P_2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as *future-knowledge scheduling*.

Answer:

a. 10.53

b. 9.53

c. 6.86

Remember that turnaround time is finishing time minus arrival time, so you have to subtract the arrival times to compute the turnaround times. FCFS is 11 if you forget to subtract arrival time.

5.4 Consider the following set of processes, with the length of the CPU burst time given in milliseconds:
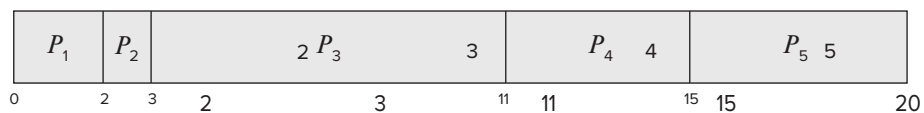
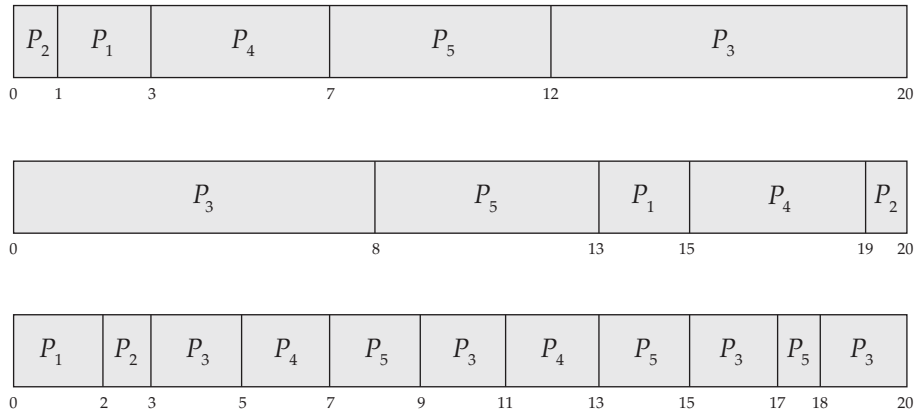| Process	Burst Time	Priority
P_1	2	2
P_2	1	1
P_3	8	4
P_4	4	2
P_5	5	3

The processes are assumed to have arrived in the order P_1, P_2, P_3, P_4, P_5, all at time 0.

a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, non-preemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2).

b. What is the turnaround time of each process for each of the scheduling algorithms in part a?

c. What is the waiting time of each process for each of these scheduling algorithms?

d. Which of the algorithms results in the minimum average waiting time (over all processes)?

Answer:

a. The four Gantt charts:

P₂	P₁	P₄	P₅	P₃
0 1	3	7	12	20

P₃	P₅	P₁	P₄	P₂
0	8	13 15	19 20

P₁	P₂	P₃	P₄	P₅	P₃	P₄	P₅	P₃	P₅	P₃
0 2 3 5 7 9 11 13 15 17 18 20

b. Turnaround time:

	FCFS	SJF	Priority	RR
P_1	2	3	15	2
P_2	3	1	20	3
P_3	11	20	8	20
P_4	15	7	19	13
P_5	20	12	13	18

c. Waiting time (turnaround time minus burst time):

	FCFS	SJF	Priority	RR
P_1	0	1	13	0
P_2	2	0	19	2
P_3	3	12	0	12
P_4	11	3	15	9
P_5	15	7	8	13

d. SJF has the shortest wait time.

5.5 The following processes are being scheduled using a preemptive, round-robin scheduling algorithm.

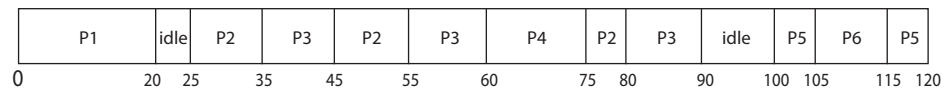| Process	Priority	Burst	Arrival
P_1	40	20	0
P_2	30	25	25
P_3	30	25	30
P_4	35	15	60
P_5	5	10	100
P_6	10	10	105

Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. In addition to the processes listed above, the system also has an **idle task** (which consumes no CPU resources and

is identified as P_{idle}). This task has priority 0 and is scheduled when-ever the system has no other available processes to run. The length of a time quantum is 10 units. If a process is preempted by a higher-priority process, the preempted process is placed at the end of the queue.

a.   Show the scheduling order of the processes using a Gantt chart.

b.   What is the turnaround time for each process?

c.   What is the waiting time for each process?

d.   What is the CPU utilization rate?

Answer:

a.   The Gantt chart:

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

0   20   25   35   45   55   60   75   80   90   100 105   115 120

b.   P1: 20-0 - 20, P2: 80-25 = 55, P3: 90 - 30 = 60, P4: 75-60 = 15, P5: 120-100 = 20, P6: 115-105 = 10

c.   P1: 0, p2: 40, P3: 35, P4: 0, P5: 10, P6: 0

d.   105/120 = 87.5 percent.

5.6   What advantage is there in having different time-quantum sizes at dif-ferent levels of a multilevel queueing system?

Answer:
Processes that need more frequent servicing—for instance, interactive processes such as editors—can be in a queue with a small time quantum. Processes with no need for frequent servicing can be in a queue with a larger quantum, requiring fewer context switches to complete the processing and thus making more efficient use of the computer.

5.7   Many CPU-scheduling algorithms are parameterized. For example, the RR algorithm requires a parameter to indicate the time slice. Multilevel feedback queues require parameters to define the number of queues, the scheduling algorithms for each queue, the criteria used to move processes between queues, and so on.
    These algorithms are thus really sets of algorithms (for example, the set of RR algorithms for all time slices, and so on). One set of algorithms may include another (for example, the FCFS algorithm is the RR algorithm with an infinite time quantum). What (if any) relation holds between the following pairs of algorithm sets?

a.   Priority and SJF

b.   Multilevel feedback queues and FCFS

c.   Priority and FCFS

d.   RR and SJF

Answer:

a. The shortest job has the highest priority.

b. The lowest level of MLFQ is FCFS.

c. FCFS gives the highest priority to the job that has been in existence the longest.

d. None.

5.8 Suppose that a CPU scheduling algorithm favors those processes that have used the least processor time in the recent past. Why will this algorithm favor I/O-bound programs and yet not permanently starve CPU-bound programs?

Answer:
It will favor the I/O-bound programs because of the relatively short CPU bursts requested by them; however, the CPU-bound programs will not starve, because the I/O-bound programs will relinquish the CPU relatively often to do their I/O.

5.9 Distinguish between PCS and SCS scheduling.

Answer:
PCS scheduling is local to the process. It is how the thread library schedules threads onto available LWPs. SCS scheduling is used when the operating system schedules kernel threads. On systems using either the many-to-one or the many-to-many model, the two scheduling models are fundamentally different. On systems using the one-to-one model, PCS and SCS are the same.

5.10 The traditional UNIX scheduler enforces an inverse relationship between priority numbers and priorities: the higher the number, the lower the priority. The scheduler recalculates process priorities once per second using the following function:

   Priority = (recent CPU usage / 2) + base

where base = 60 and *recent CPU usage* refers to a value indicating how often a process has used the CPU since priorities were last recalculated.

   Assume that recent CPU usage for process P_1 is 40, for process P_2 is 18, and for process P_3 is 10. What will be the new priorities for these three processes when priorities are recalculated? Based on this information, does the traditional UNIX scheduler raise or lower the relative priority of a CPU-bound process?

Answer:
The priorities assigned to the processes will be 80, 69, and 65, respectively. The scheduler lowers the relative priority of CPU-bound processes.