CHAPTER 5

Exercises

5.3 One technique for implementing **lottery scheduling** works by assigning processes lottery tickets, which are used for allocating CPU time. Whenever a scheduling decision has to be made, a lottery ticket is chosen at random, and the process holding that ticket gets the CPU. The BTV operating system implements lottery scheduling by holding a lottery 50 times each second, with each lottery winner getting 20 milliseconds of CPU time (20 milliseconds × 50 = 1 second). Describe how the BTV scheduler can ensure that higher-priority threads receive more attention from the CPU than lower-priority threads.

Answer:

By assigning more lottery tickets to higher-priority processes.

5.5 Consider the exponential average formula used to predict the length of the next CPU burst. What are the implications of assigning the following values to the parameters used by the algorithm?

a. $\alpha = 0$ and $\tau_0 = 100$ milliseconds
b. $\alpha = 0.99$ and $\tau_0 = 10$ milliseconds

Answer:

When $\alpha = 0$ and $\tau_0 = 100$ milliseconds, the formula always makes a prediction of 100 milliseconds for the next CPU burst. When $\alpha = 0.99$ and $\tau_0 = 10$ milliseconds, the most recent behavior of the process is given much higher weight than the past history associated with the process. Consequently, the scheduling algorithm is almost memoryless, and simply predicts the length of the previous burst for the next quantum of CPU execution.

5.6 A variation of the round-robin scheduler is the **regressive round-robin scheduler**. This scheduler assigns each process a time quantum and a priority. The initial value of a time quantum is 50 milliseconds. However, every time a process has been allocated the CPU and uses its entire time quantum (does not block for I/O), 10 milliseconds is added to its time quantum, and its priority level is boosted. (The time quantum for a process can be increased to a maximum of 100 milliseconds.) When a process blocks before using its entire time quantum, its time quantum is reduced by 5 milliseconds, but its priority remains the same. What type of process (CPU-bound or I/O-bound) does the regressive round-robin scheduler favor? Explain.

Answer:

This scheduler would favor CPU-bound processes as they are rewarded with a longer time quantum as well as priority boost whenever they consume an entire time quantum. This scheduler does not penalize I/O-bound processes as they are likely to block for I/O before consuming their entire time quantum, but their priority remains the same.

5.7 Consider the following set of processes, with the length of the CPU burst time given in milliseconds:

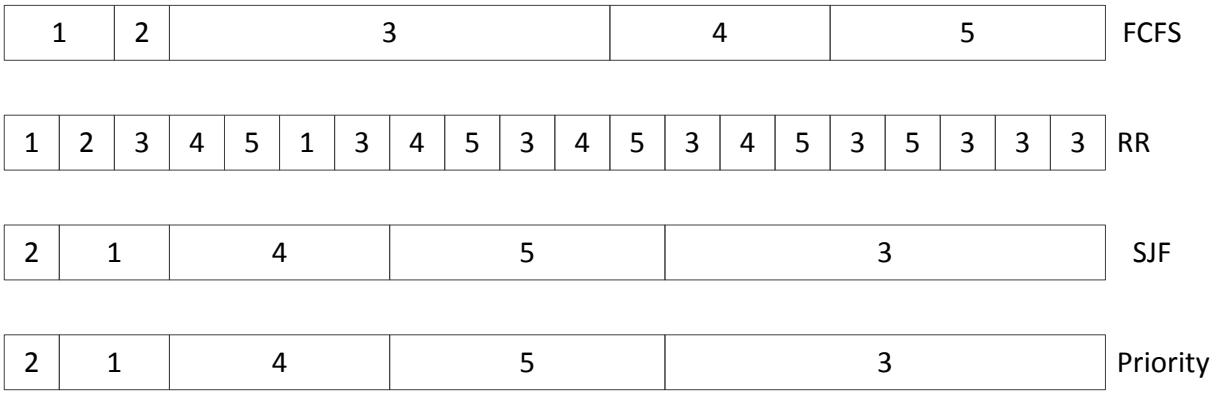
Process	Burst Time	Priority
P_1	2	2
P_2	1	1
P_3	8	4
P_4	4	2
P_5	5	3

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, nonpreemptive priority (a smaller priority number implies a higher priority), and RR (quantum = 1).
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 are



b. Turnaround time

	FCFS	RR	SJF	Priority
P_1	2	6	3	3
P_2	3	2	1	1
P_3	11	20	20	20
P_4	15	14	7	7
P_5	20	17	12	12

c. Waiting time (turnaround time minus burst time)

	FCFS	RR	SJF	Priority
P_1	0	4	1	1
P_2	2	1	0	0
P_3	3	12	12	12
P_4	11	10	3	3

P_5	15	12	7	7

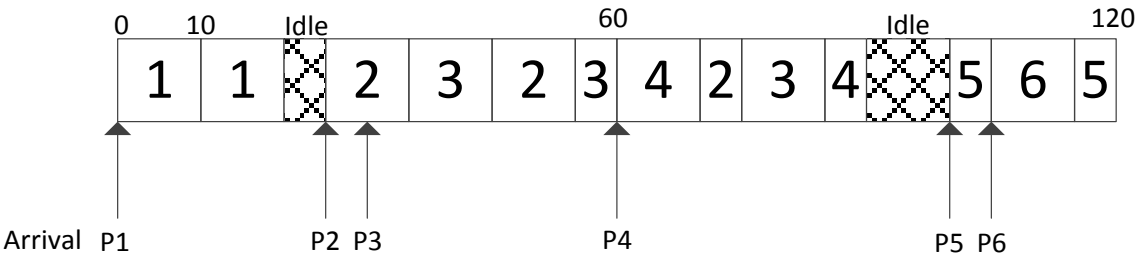
d. Shortest Job First and Priority (since for this example, shortest jobs have highest priority)

5.8 The following processes are being scheduled using a preemptive, round-robin scheduling algorithm. Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. In addition to the processes listed below, 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 whenever 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.

Thread	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

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. Gantt chart above.
b. p1: 20-0 - 20, p2: 75-25 = 50, p3: 85 - 30 = 55, p4: 90-60 = 30, p5: 120-100 = 20, p6: 115-105 = 10
c. p1: 0, p2: 25, p3: 30, p4: 15, p5: 10, p6: 0
d. 105/120 = 87.5 percent.

5.10 Which of the following scheduling algorithms could result in starvation?

a. First-come, first-served
b. Shortest job first
c. Round robin
d. Priority

Answer:

Shortest job first and priority-based scheduling algorithms could result in starvation.

5.11 Consider a variant of the RR scheduling algorithm where the entries in the ready queue are pointers to the PCBs.

a. What would be the effect of putting two pointers to the same process in the ready queue?
b. What would be two major advantages and disadvantages of this scheme?
c. How would you modify the basic RR algorithm to achieve the same effect without the duplicate pointers?

Answer:

a. In effect, that process will have increased its priority since by getting time more often it is receiving preferential treatment.
b. The advantage is that more important jobs could be given more time, in other words, higher priority in treatment. The consequence, of course, is that shorter jobs will suffer.
c. Allot a longer amount of time to processes deserving higher priority. In other words, have two or more quantums possible in the Round-Robin scheme.

5.12 Consider a system running ten I/O-bound tasks and one CPU-bound task. Assume that the I/O-bound tasks issue an I/O operation once for every millisecond of CPU computing and that each I/O operation takes 10 milliseconds to complete. Also assume that the context-switching overhead is 0.1 millisecond and that all processes are long-running tasks. Describe is the CPU utilization for a round-robin scheduler when:

a. The time quantum is 1 millisecond
b. The time quantum is 10 milliseconds

Answer:

a. The time quantum is 1millisecond: Irrespective of which process is scheduled, the scheduler incurs a 0.1 millisecond context-switching cost for every context-switch. This results in a CPU utilization of 1/1.1 * 100 = 91%.

b. The time quantum is 10 milliseconds: The I/O-bound tasks incur a context switch after using up only 1 millisecond of the time quantum. The time required to cycle through all the processes is therefore 10*1.1 + 10.1 (as each I/O-bound task executes for 1 millisecond and then incur the context switch task, whereas the CPU-bound task executes for 10 milliseconds before incurring a context switch). The CPU utilization is therefore 20/21.1 * 100 = 94%.

5.13 Consider a system implementing multilevel queue scheduling. What strategy can a computer user employ to maximize the amount of CPU time allocated to the user's process?

Answer:

The program could maximize the CPU time allocated to it by not fully utilizing its time quantums. It could use a large fraction of its assigned quantum, but relinquish the CPU before the end of the quantum, thereby increasing the priority associated with the process.

5.14 Consider a preemptive priority scheduling algorithm based on dynamically changing priorities. Larger priority numbers imply higher priority. When a process is waiting for the CPU (in the ready queue, but not running), its priority changes at a rate α; when it is running, its priority changes at a rate β. All processes are given a priority of 0 when they enter the ready queue. The parameters α and β can be set to give many different scheduling algorithms.

a. What is the algorithm that results from $\beta > \alpha > 0$?
b. What is the algorithm that results from $\alpha < \beta < 0$?

Answer:

a. FCFS
b. LIFO

5.15 Explain the differences in how much the following scheduling algorithms discriminate in favor of short processes:

a. FCFS
b. RR
c. Multilevel feedback queues

Answer:

a. FCFS—discriminates against short jobs since any short jobs arriving after long jobs will have a longer waiting time.
b. RR—treats all jobs equally (giving them equal bursts of CPU time) so short jobs will be able to leave the system faster since they will finish first.
c. Multilevel feedback queues work similar to the RR algorithm—they discriminate favorably toward short jobs.

5.18 Consider the scheduling algorithm in the Solaris operating system for time-sharing threads:

a. What is the time quantum (in milliseconds) for a thread with priority 10? With priority 55?
b. Assume a thread with priority 35 has used its entire time quantum without blocking. What new priority will the scheduler assign this thread?
c. Assume a thread with priority 35 blocks for I/O before its time quantum has expired. What new priority will the scheduler assign this thread?

Answer:

a. 160 and 40
b. 35
c. 54

5.19 Assume that two tasks *A* and *B* are running on a Linux system. The nice values of *A* and *B* are−5 and+5, respectively. Using the CFS scheduler as a guide, describe how the respective values of `vruntime` vary between the two processes given each of the following scenarios:

- Both *A* and *B* are CPU-bound.
- *A* is I/O-bound, and *B* is CPU-bound.
- *A* is CPU-bound, and *B* is I/O-bound.

Answer:

- Since A has a higher priority than *B*, `vruntime` will move more slowly for *A* than *B*. If both *A* and *B* are CPU-bound (that is they both use the CPU for as long as it is allocated to them), `vruntime` will generally be smaller for *A* than *B*, and hence *A* will have a greater priority to run over *B*.
- In this situation, `vruntime` will be much smaller for *A* than *B* as (1) `vruntime` will move more slowly for *A* than *B* due to priority differences, and (2) Awill require less CPU-time as it is I/O-bound.
- This situation is not as clear, and it is possible that *B* may end up running in favor of *A* as it will be using the processor less than *A* and in fact its value of `vruntime` may in fact be less than the value of `vruntime` for *B*.

5.22 Consider two processes, P_1 and P_2, where $p_1 = 50$, $t_1 = 25$, $p_2 = 75$, and $t_2 = 30$.

a. Can these two processes be scheduled using rate-monotonic scheduling? Illustrate your answer using a Gantt chart such as the ones in Figure 5.16–Figure 5.19.
b. Illustrate the scheduling of these two processes using earliest-deadline-first (EDF) scheduling.

Answer:

a. Rate monotonic: Consider when P_1 is assigned a higher priority than P_2 with the rate monotonic scheduler. P_1 is scheduled at t= 0, P_2 is scheduled at t = 25, P_1 is scheduled at t = 50, and P_2 is scheduled at t = 75. P_2 is not scheduled early
enough to meet its deadline. When P_1 is assigned a lower priority than P_2, then P_1 does not meet its deadline since it will not be scheduled in time.
b. EDF: It would be possible to schedule the given periodic processes using EDF as shown in the gantt chart below: