

## 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 ( $20 \text{ milliseconds} \times 50 = 1 \text{ second}$ ). Describe how the BTV scheduler can ensure that higher-priority threads receive more attention from the CPU than lower-priority threads.

**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?

- $\alpha = 0$  and  $\tau_0 = 100$  milliseconds
- $\alpha = 0.99$  and  $\tau_0 = 10$  milliseconds

**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.

**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.

- 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).
- What is the turnaround time of each process for each of the scheduling algorithms in part a?
- What is the waiting time of each process for each of these scheduling algorithms?
- Which of the algorithms results in the minimum average waiting time (over all processes)?

**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

- Show the scheduling order of the processes using a Gantt chart.
- What is the turnaround time for each process?
- What is the waiting time for each process?
- What is the CPU utilization rate?

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

- First-come, first-served
- Shortest job first
- Round robin
- Priority

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

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

**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 the CPU utilization for a round-robin scheduler when:

- The time quantum is 1 millisecond
- The time quantum is 10 milliseconds

**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?

**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  $\alpha$ ; when it is running, its priority changes at a rate  $\beta$ . All processes are given a priority of 0 when they enter the ready queue. The parameters  $\alpha$  and  $\beta$  can be set to give many different scheduling algorithms.

- What is the algorithm that results from  $\beta > \alpha > 0$ ?
- What is the algorithm that results from  $\alpha < \beta < 0$ ?

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

- FCFS
- RR
- Multilevel feedback queues

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

- 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?

**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.

**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.