Department of Computer Engineering
University of Peradeniya

CO327: Operating Systems

Assignment 3

May 31, 2020

1. Why is it important for the scheduler to distinguish I/O-bound programs from CPU-bound programs?
2. Explain the difference between preemptive and non-preemptive scheduling.
3. Discuss how the following pairs of scheduling criteria conflict in certain settings.
   (a) CPU utilization and response time
   (b) Average turnaround time and maximum waiting time
   (c) I/O device utilization and CPU utilization
4. 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.
5. 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 are 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 favour? Explain.
6. 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 non-preemptive scheduling, and base all decisions on the information you have at the time the decision must be made.

Process	Arrival Time	Burst Time
P1	0.0	8
P2	0.4	4
P3	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 P1 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 P1 and P2 are waiting during this idle time, so their waiting time may increase. This algorithm could be called *future-knowledge Scheduling*.

7. What advantage is there in having different time-quantum sizes at different levels of a multilevel queueing system?
8. Most scheduling algorithms maintain a run queue, which lists processes eligible to run on a processor. On multicore systems, there are two general options: (1) each processing core has its own run queue, or (2) a single run queue is shared by all processing cores. What are the advantages and disadvantages of each of these approaches?
9. 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 $\alpha < \beta < 0$?
10. Explain why interrupt and dispatch latency times must be bounded in a hard real-time system.
11. Write a short note on the current status and the historical evolution of the Linux Scheduler.

DEADLINE: 06:00 p.m. on Tuesday, January 22, 2019