Operating Systems, Spring 2020, Exercise 4

1. Five batch jobs A through E, arrive at a computer center at almost the same time. They have estimated running times of 10, 7, 3, 6, and 12 minutes. Their (externally determined) priorities are 6, 2, 7, 9, and 4 respectively, with a lower value corresponding to a higher priority.

   For each of the following scheduling algorithms, determine the turn-around time for each process and the average turnaround time for all jobs. Ignore process switching overhead.

   a. Round robin
   b. Priority scheduling
   c. First-come-first-served (run in the order listed above)
   d. Shorted-job-first

   For (a) assume that the system is multiprogrammed, and that each job gets its fair share of the CPU, for example 2 minute-time-slot at a time. For (b) through (d) assume that only one job at a time runs, until it finishes. All jobs are completely CPU bound.

2. Traditional Unix scheduling. See Fig 9.17 in Ch 9 (slide 49 in Ch 9).
   a. Is this good for real time system?
   b. Can any (user) process starve? How or why not?
   c. When would a user use the *nice* factor for his/her process?
   d. Can nice factor be negative? Can it hamper OS services?
   e. In Fig 9.17, which process would run next at time 5 sec? Assume that it terminates at time 5.81 sec. Which process will run at time 6 sec?

3. (Problem 9.3 [Stal 12])
   Prove, that among nonpreemptive scheduling algorithms, shortest-job-first provides the minimum average waiting time for a batch of jobs that arrive at the same time. Assume that the scheduler must always execute a task if one is available. You may assume that the jobs arrive at the same time.

4. (Problem 2.43 [Tan08] modified)
   A real-time system needs to handle two (voice) phone calls that each run every 5 msec and consume 1 msec of CPU time per burst, plus one video at 25 frames/sec, with each frame requiring 15 msec of CPU time.
   a. Is this system schedulable?
   b. How would you guarantee that both calls and video will execute properly (i.e., which priorities/scheduling protocol would you use)?

5. Aperiodic tasks, priority inversion
   a. (Probl. 10.2 [Sta18]) Aperiodic tasks. Consider a set of five aperiodic tasks with execution profiles of Table 10.8 (10.7 in [Sta12]) (http://www.cs.helsinki.fi/i/kerola/kj/stal_9th_tbl-10-8.jpg)
      Develope scheduling diagrams similar to those of Figure 10.6 (slide 51 in Ch 10) for this set of tasks.
   b. (Probl. 10.8 [Stal12]) Priority inversion. Draw a diagram similar to Fig 10.9b (slide 58 in Ch 10) that shows the sequence of events for this same example using priority ceiling.
      What is the major difference between priority inheritance and priority ceiling?
      What is the real problem that these methods solve?

6. Unix SVR4 scheduling (Chapter 10.4)
   a. How many dispatch queues are there?
   b. How do you quickly find the next process to run without going through all queues?

c. Why do you need preemption points in kernel code? When are they needed?
d. Why is time quantum longer for priority 0 processes than for priority 59 processes? Why is small time quantum ok for priority 59 processes?