Unit 9 Sample Problems - CPU Scheduling

In this exercise, we will review the concepts of process scheduling.

- Length: 1:15 minutes with discussion.
- Questions: Q1, Q3, Q5-Q8, Q10, Q12.

1 Basic Concepts

1. [Lisonbee] You are editing a video that uses very high resolution, uncompressed video files. When you go to render the video, how will the CPU bursts for this process look relative to the I/O bursts? **Explain.** [2 points]

   Ans: [Lisonbee]

   Due to the size of the video files, the CPU bursts will be short relative to the I/O bursts. Most of the time that this process spends rendering the video will be dealing with the large amounts of data rather than performing computations.

2. [Alvaran] You run a console-based calculator program, which waits for user input. You submit an additional problem that the program calculates and outputs to the screen. Lastly you terminate the program. **Explain** which parts of the scenario correspond to scheduling events? For each event, what type of scheduling is used (preemptive vs non-preemptive)? [2 points]

   Ans: [Alvaran] The program prompting the user for a math problem involves switching from the ready state to the waiting state, which is non-preemptive scheduling. Receiving then calculating the input involves a switch from the waiting state to the ready state, which is preemptive scheduling. The termination of the program involves non-preemptive scheduling.

2 Scheduling Criteria

3. [Lisonbee] Consider a scenario where all queued jobs run for a random amount of time, but all are equally important in terms of the work they complete. Of the five criteria for scheduling jobs, which one would be the most important to optimize in this situation? **Explain.** [2 points]

   Ans: [Lisonbee]

Throughput would be the most crucial criteria for this situation as each job is equally weighted; they should be scheduled so that the most amount of jobs can be completed within a fixed amount of time.

4. [Lisonbee] If you are trying to optimize for waiting time, roughly how will the processes be ordered in terms of the time they take to complete? **Explain.** [2 points]

   Ans: [Lisonbee]

   To optimize for waiting time, generally shorter jobs will be scheduled first, and then longer ones to ensure that the majority of jobs are not waiting a long time in the queue.

3 Scheduling Algorithms

5. [Lisonbee] Consider a set of ordered pairs of process IDs and CPU time: (P0, 12), (P1, 6), (P2, 3), (P3, 4). **Calculate** both the average turnaround time and average waiting time of this set using FCFS scheduling. (You may leave the answer as a fraction.) [2 points]

   Ans: [Lisonbee]

   Average completion time = (12+18+21+25)/4=19

   Average waiting time = (0+12+18+21)/4=12.75

6. [Lisonbee] Consider a set of ordered pairs of process IDs and CPU time: (P0, 12), (P1, 6), (P2, 3), (P3, 4). **Calculate** both the average turnaround time and average waiting time of this set using SJF scheduling. (You may leave the answer as a fraction.) [2 points]

   Ans: [Lisonbee]

   Average completion time = (3+7+13+25)/4=12

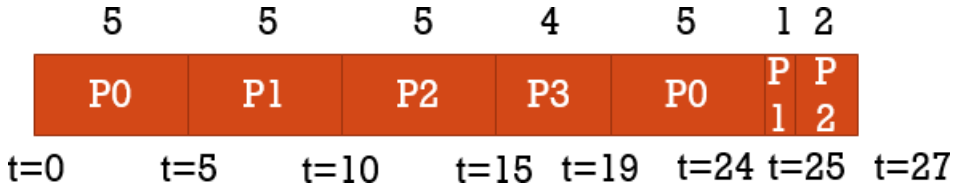
   Average waiting time = (0+3+7+13)/4=5.75

7. [Acuña] Consider a set of ordered pairs of process IDs and CPU time: (P0, 10), (P1, 6), (P2, 7), (P3, 4). **Calculate** both the average turnaround time and average waiting time of this set using RR scheduling with a time quantum of 5. (You may leave the answer as a fraction.) [4 points]

Ans: [Acuña]

Average Completion Time: (24+25+27+19)/4=23.75

Average Waiting Time: (14+19+20+15)/4=17



8. [Silberschatz 6.4] What advantage is there in having different time-quantum sizes at different levels of a multilevel feedback queuing system? **Explain.** [2 points]

   Ans: [Acuña]

   It will improve the throughput of the system by reducing the amount of time small processes spend waiting. In the initial queue, small jobs will be able to complete immediately from the low quantum. Larger jobs will drop down into the next queue with a higher quantum. Since a multilevel queue goes from top to bottom, the jobs that have consumed the least amount of time will be processed first. Jobs that need more time will be given a lower priority since they will filter down to have a higher quantum.

4 Thread Scheduling

9. [Acuña] Would we ever want a program using 1:1 thread mapping to use PCS? Would we ever want a program using n:m or n:1 thread mapping to use SCS? **Explain.**

   Ans: [Acuña]

   A program using n:m or n:1 mapping can't use SCS because there aren't enough kernel-level threads to do 1:1 mapping.

   A program using 1:1 mapping could use if we wanted scheduling to occur at two levels.

5 Multiple-Processor Scheduling

10. [Acuña] Consider the problem of running a large algorithm on a image (such as in a previous homework), using threads to split work across multiple CPU cores. There there are two possible approaches: one is to use global shared memory for the threads, and the second is to use geometric deposition to split the data into a number of chunks equal to the number of threads. **Explain** which approach has better potential to lead to better performance and why. If your answer uses any assumptions about the scheduler, state them.

    Ans: [Acuña]

The second approach should better. The issue with the first way is that there is only one memory allocation that all threads use, meaning it must be moving around different physical CPUs for the threads to get data. This forms a bottleneck. In the second, each thread has the ability to operate independently with it's data. However, this requires the assumption that the scheduler will try to place threads on CPUs where their independent piece of data already exists. If the thread is scheduled on a new CPU, then data will have to be moved between cores and performance will be degraded.

6 Real-Time CPU Scheduling

11. [Acuña] Why would it be inappropriate to use something like a SJF or Priority scheduler to handle polling sensors at regular intervals in a self-driving car and then making adjustments to the car? **Explain.**

    Ans: [Acuña]

    In a hard real-time environment, we need to have some guarantee of the latency between starting some event (to poll) and some action (adjusting a car part). This simply isn't captured by the simpler schedulers or their process models. We need to know about process deadlines to make judgment if a process: 1) should run. 2) can run. Although we can technical put a bunch of instances of the process into a priority scheduler, we won't be able to determine if the processes can run successfully in a period.

12. [Acuña] Consider the processes: (P0, 30, 15), (P1, 25, 15). The triples have the form (process, period, time). Assume that deadline and period are identical. **Draw** the scheduling that would occur for these processes using Rate-Monotonic Scheduling over 60 time units.

    Ans: [Acuña]