CSCI 3753: Operating Systems
Fall 2016
Problem Set Three

Please write your answers in the space provided.
Due date: Tuesday, October 25 in class. No extensions will be given except at the instructor's discretion in documented cases of extreme hardship or emergencies. Please submit a hardcopy of your solutions.

Problem 1. [40 Points] Consider the following set of processes, with the length of the CPU execution time given in seconds:

Process	Execution Time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

Draw four Gantt charts that illustrate the timeline of execution of the processes when scheduled according to FCFS, SJF, nonpreemptive priority (priority 1 is most important), and round robin (time slice=1). Which one has the lowest average wait time? Which one has the lowest average turnaround time?

Problem 2. Suppose the Completely Fair Scheduler (CFS) algorithm is applied to the following scenario. There exist two processes P1 and P2 that need to be scheduled. P1 has two threads T1 and T2 that are implemented as kernel threads. P2 has three threads T3, T4, and T5 that are implemented as user space threads. We apply round robin scheduling to all schedulable tasks, where each task schedulable by the kernel receives a time slice of size T seconds.

(a) **[20 Points]** Draw a table of wait times (times owed on an ideal CPU) for one round robin of this scenario, i.e. what are the wait time balances as each time slice is allocated to a kernel-schedulable task, and what is the final wait time for each such task?

Since the Kernel doesn't see user level threads, the scheduling entities will be T1, T2 and P2, and each of these will be allocated T seconds of CPU time in each round. This means the average wait time for T1 and T2 will be twice their execution times assuming sufficiently large execution times. T3, T4 and T5 will each get T/3 seconds of CPU in each round. So, the average wait times for these three threads will be eight times their respective execution times, again assuming sufficiently large execution times.

(b) **[10 Points]** Is this system fair at the level of schedulable tasks? Is this system fair at the level of processes? Justify your answers.

The system is fair at the level of tasks, i.e. kernel threads, but not applications. Every schedulable thread gets exactly 1/3 of CPU. But the system is not fair at the level of applications. Application 1 (Process 1) gets 2/3 of CPU because it has two threads, while Application 2 (Process 2) gets 1/3 of CPU. This is why Linux introduced the idea of "groups" into scheduling.

Problem 3. [35 Points] (Based on exercise 6.12, page 307 in the textbook)
Lottery scheduling works by assigning processes lottery tickets, which are used for assigning CPU time. Whenever a scheduling decision has to be made, a lottery ticket is chosen at random, and the process that holds that ticket gets the CPU.

(a) **[10 Points]** Is starvation possible in lottery scheduling? Explain your answer.

Yes. Since a ticket is chosen at random, there exists a scenario where a ticket belonging to a process never gets chosen.

(b) **[10 Points]** Describe how a lottery scheduler can favor I/O bound processes over CPU bound processes.

Assign more tickets to I/O bound processes.

(c) **[10 Points]** Discuss the advantages and disadvantages of lottery scheduling compared to multi-level feedback queue.

Advantages: Lottery scheduling is much simpler to implement. There is no need to maintain any complex data structures or track execution times.

Disadvantages: Starvation is possible in lottery scheduling.