// Anisha Aggarwal
// CIS415
// Assignment 2

1. OSC 4.14

    a) I would create a single thread for input and a single thread for output because blocking will occur, so there is no benefit to creating more than a single thread for each. They are also reading and writing into a single file.

    b) I would create 4 threads because there should be as many threads as there are processors available to get the most concurrency.

2. OSC 4.18
    a) If the number of kernel threads is less than the number of processors, then some of processors will be unused and the concurrency will be reduced. This would also cause other user threads to not be able to run while the available kernel threads are waiting. Therefore performance will be suboptimal.

    b) If the number of kernel threads is equal to the number of processors, then all of the processing cores will be used when needed, thus providing the most concurrency and best performance.

    c) If the number of kernel threads is greater than the number of processors but less than the number of user-level threads, then all of the processor cores will be in use when needed. There is an overhead in creating extra kernel threads, however it's performance is just as good as (b).

3. This algorithm will favor I/O bound processes because I/O bound processes typically run for a short duration before they give-up CPU and wait for I/O to complete. Given that they are short CPU bursts, there wouldn't be significant impact on waiting time for CPU bound processes, therefore will not permanently starve CPU bound processes.

4. OSC 6.11

    a) To improve response time, the scheduler would use smaller time quantum which in turn causes high context-switch overhead and lower CPU utilization.

    b) To improve the average turnaround time, by doing SJF (shortest job first), you will increase the waiting time for the long running jobs and decrease it for short running jobs. Therefore the maximum waiting time could be increased.

    c) To improve I/O device utilization, scheduler would want to prioritize short running I/O jobs. This would result in higher context-switch overhead and hence reduce CPU utilization.

5. Real-time Scheduling

a) Rate-monotonic scheduling decides priority of a job based on its period. The shorter the period, the higher the priority will be. A higher priority job will preempt the lower priority job. It is classified as a fixed-priority algorithm because the priority will be decided when the job is submitted and the priority of a job will not change.

b) Periodic tasks: T1 = (0, 4, 1), T2 = (0, 8, 2), T3 = (0, 20, 2)

i. CPU utilization = (1/4) + (2/8) + (2/20) = 0.25 + 0.25 + 0.1 = 0.6

CPU Utilization bound = 3(2^(1/3) - 1) = 0.7798, which is >0.6 so it is feasible to schedule.

Time	Ready to Run	Scheduled
0	T1, T2, T3	T1
1	T2, T3	T2
3	T3	T3
4	T1, T3	T1
5	T3	T3
6	-	-
8	T1, T2	T1
9	T2	T2
11	-	-
12	T1	T1
13	-	-
16	T1, T2	T1
17	T2	T2
19	-	-
20	T1, T2, T3	T1
…	…	…

ii. With π =5,
Total CPU utilization = (1/4) + (2/8) + (2/20) + (1/5) = 0.25 + 0.25 + 0.1 + 0.2 = 0.8

CPU Utilization bound = 4(2^(1/4) - 1) = 0.757

Since CPU utilization of 0.8 is more than CPU utilization bound of 0.757, it is not feasible to schedule.