CS 377 – Operating Systems

Discussion Session 4 Questions

Name: _____

Write your answers individually, but feel free to consult your notes/slides/book this week. Be succinct (complete sentences not necessary). **Remember to turn your paper over.**

1. **CPU Scheduling**. Several types of scheduling policies were discussed in class – first-come-first-served (FCFS), round robin (RR), shortest job first (SJF, including multilevel feedback queues), and lottery scheduling (LS).

   (a) Suppose you want to optimize your scheduler for certain types of workloads. For each type, state and briefly justify which type of scheduler you would use: (i) multiuser workloads in which no individual user should be favored, (ii) workloads with many mixed CPU and I/O jobs, and (iii) workloads with frequent I/O bound jobs and some very long-running, CPU-heavy jobs.

   (b) Suppose you have 2 jobs: job A has length 10 and job B has length 20. Job A has 1 second of I/O every other second of work (starting after 1 second of work), while job B has 1 second of I/O every 5 seconds of work. Using multilevel feedback queues and assuming three queues and no context switch time, sketch the scheduling of the jobs below. Remember the notation $Job_{time}^{workDone}$; for example, B_6^2 means that job B has completed 2 seconds of work at time $t = 6$. The first two entries are filled in for you.

Queue	Time Slice	Job
1	1	A^1_1 B^1_2
2	2	
3	4	

2. **Threads**. Two primary types of threads were discussed in class – user-level threads and kernel-level threads. Threads complement processes as basic components used to execute jobs on the CPU.

   (a) True or false: user-level and kernel-level threads are exclusive (that is, processes use one or the other). Briefly explain.

   (b) Suppose you have a multithreaded process that can be configured to use either kernel or user-level threads. Under each of the following situations about the process, which type of threads would you prefer (and why): (i) running on a quad-core machine, (ii) executing long I/O requests, and (iii) running an extremely large number of threads.

   (c) Why would we want to use (kernel-level) threads at all instead of just using multiple processes?