CS 33211 **Homework #2** OS

Due in class at 12:30pm on Wednesday 24 October 2007
typed answers preferred

1. **The two-level thread model provides the benefits of both user-level and kernel-level threads. Explain.**

   It allows user-level threads to be scheduled onto lightweight processes, which provides many of the benefits of user-level threads — simple representation within the process's memory space, minimal OS involvement, and flexible scheduling under the user's control.

   By allowing multiple lightweight processes, each bound to a kernel-level thread, it provides many of the benefits of kernel-level threads — the possibility of the kernel giving that process multiple time slides, and the ability to switch to a different lightweight process if the one currently running blocks.

2. **In CPU scheduling, it is desirable to avoid a process' starvation. Explain why starvation is, or is not, possible in Round-Robin and Shortest-Remaining-Time scheduling.**

   In Round-Robin scheduling, starvation is not possible because each process runs for at most a fixed amount of time and all the ready processes run in rotation.

   In Shortest-Remaining-Time scheduling starvation is possible for long processes, since short processes may require less time to run than the long process's remaining time, and if enough such short processes keep arriving, the long process will never get to run.

3. **Non-preemptive CPU scheduling selects a new process to dispatch onto the CPU only when a running process terminates or blocks. Preemptive CPU scheduling may also select a new process to dispatch at other times. For each of those other times, explain the rationale for selecting a new process at that time.**

   Preemptive scheduling may set a timer, and choose a new process when the time expires, as Round-Robin does, with the rationale of preventing CPU-bound processes from keeping the CPU for prolonged periods.

   Preemptive scheduling may choose a new process when a waiting process unblocks or is created, as Shortest-Remaining-Time does, with the rationale of always letting the shortest process run, whether that be a process that has just entered the ready list or the one that is currently running.

4. **What are the deficiencies with mutual exclusion Algorithm 1?**

(Note that the algorithm does "work" correctly in that it does enforce mutual exclusion and does prevent deadlock). Algorithm 1's primary deficiency is that it is overly restrictive — it explicitly forces the two processes to take turns, and will not allow one to run more frequently during a certain time interval. It also has the deficiency that if a process crashes while in the critical section it will never update the turn variable so the other process will never get a chance to enter the critical section. Finally, the algorithm uses busy-waiting, which is a waste of CPU resources.

5. In the bounded-buffer producer-consumer problem solution using semaphores, semaphores are used for two distinctly different purposes. Explain.

The binary semaphore mutex is used to enforce mutual exclusion, so that only one of the two processes is allowed to manipulate the shared resource (the Coke slots inside the machine) at a given time.

The counting semaphores fullSlot and emptySlot are used for synchronization, so that if a given condition is not met (e.g., no empty slots for the DeliveryPerson to fill, or no full slots (Cokes) for the ThirstyPerson to drink) the process blocks until the condition is met.