Name: _____

CS 33211 **Exam #2** OS

**Monday 5 November 2007**

1. **All of the threads in a process share the same program code, yet each requires storage for its own Program Counter value in its Thread Control Block. Explain. (8 points)**

   All the threads in a process share the same code but may be executing that code at different points in the code, so it is necessary to keep track of what instruction in the code each thread is executing. When the thread is actively running, the CPU's Program Counter points to the next instruction to be executed, and when there is a context switch and the thread is moved off the CPU, that Program Counter value is stored in the thread's Thread Control Block. When the thread is dispatched to run again, the value from the Thread Control Block is loaded into the CPU's Program Counter.

2. **Which would be better suited for implementing a server process that is expected to handle multiple users simultaneously — user-level or kernel-level threads — and why would that type of threads be better suited? (12 points)**

   Kernel-level threads would be better suited because if the kernel knows the process has multiple threads it can choose to give it more execution time than if it has only a single thread. Also, with kernel-level threads, if one thread blocks the CPU can simply schedule another thread in the process, whereas if a user-level thread blocks, the entire process blocks.

   *However, I gave credit for a good argument in favor of user-level threads on the basis that handling multiple users and switching between them quickly would work better with user-level threads since they involve much less OS overhead and are have faster context switches.*

3. **Shortest-Job-First (SJF) and Shortest-Remaining-Time (SRT) are two related CPU scheduling algorithms.**

   a. **How do these two algorithms differ with respect to picking the next process to run on the CPU? (10 points)**

      They choose from a different pool of processes: SJF chooses the process that has the smallest next CPU burst among those in the ready list, whereas SRT chooses the process that has the smallest expected CPU burst among those in the ready list as well as the remaining expected CPU burst of the currently running process. Further, SJF is non-preemptive so it only picks a new process when the current one terminates or blocks, whereas SRT is preemptive so it also picks when a new process enters the ready list.

Name: _____

**b. How do these two algorithms differ with respect to their treatment of processes with a long CPU burst? (10 points)**

In SJF, a long process could have to wait a long time to execute (possibly forever; it could theoretically starve) but once it starts executing it gets to finish. In SRT, a long process could also have to wait a long time to execute, but once it gets started it may be prempted if a short process enters the ready list, though at least in that case it will be shorter than before so possibly will get to run again sooner.

**4. Why do we care so much about processes with a short CPU burst and want to develop CPU scheduling algorithms that give preferential treatment to those processes? (5 points)**

Because interactive processes usually have short CPU bursts to process their I/O, and users generally want good response for those interactive processes. Further, by getting short processes out of the ready list quickly there are less processes to keep track of and as a result, less overhead for the OS.

**5. Consider a semaphore S that is initialized to 1, and a critical region protected by that semaphore.**

**a. What happens when process A calls the P operation on that semaphore (answer in terms of the value of the semaphore S and the effect on process A or any other process)? (5 points)**

S is decremented from 1 to 0
Since S is not < 0, A continues into the critical region
no other processes are affected

**b While process A is in the critical region, what happens when process B calls the P operation on that semaphore (answer in terms of the value of the semaphore S and the effect on process B or any other process)? (5 points)**

S is decremented from 0 to -1
Since S < 0, B blocks and is added to a queue of semaphores waiting on S
A is already in the critical region and there is no affect on A

**c. While process A is still in the critical region, what happens when process C calls the P operation on that semaphore (answer in terms of the value of the semaphore S and the effect on process C or any other process)? (5 points)**

S is decremented from -1 to -2
Since S < 0, C blocks and is added to a queue of semaphores waiting on S (along with B)
A is already in the critical region and there is no affect on A

Name: _____

**d. What happens when process A leaves critical region and calls the V operation on that semaphore (answer in terms of the value of the semaphore S and the effect on process A or any other process)? (5 points)**

S is incremented from -2 to -1
Since S <= 0, B is removed from the queue of semaphores waiting on S (B is removed before C since it has been waiting the longest), woken up, and added to the ready list
C remains on the queue of semaphores waiting on S

**6. In the Coke machine bounded-buffer produce-consumer problem using semaphores, what operations are performed on counting semaphore emptySlot, which was initialized to 100? (10 points)**

DeliveryPerson does a P on emptySlot before putting a Coke in the machine, which decrements the value of emptySlot, forcing the Delivery Person to sleep if there is no empty slot until one becomes available.

Thirsty person does a V on emptySlot after putting a Coke in the machine, which increments the value of emptySlot, waking up a sleeping Delivery Person if there is one waiting for an empty slot.

**7. Deadlock can be prevented if the "mutual exclusion" criterion can be avoided. Explain how printer spooling can avoid mutual exclusion and thus prevent deadlock from occurring. (10 points)**

A process wanting to print gives the print job to the print spooler, which sends jobs to the printer sequentially and thus preserving mutually exclusive access to the printer. However, the process wanting to print immediately continues its execution as soon as it hands the print job to the print spooler. As a result, multiple processes may "think" they are printing simultaneously — avoiding the mutual exclusion restriction — though in reality their print jobs are printed sequentially.

**8. When deadlock is detected in a system, the deadlock can be detected by terminating a process. How is the process to be terminated selected? (15 points)**

There are many options. The OS can terminate all deadlocked processes though that's unnecessary. The OS can terminate one of the deadlocked processes, considering the number of resources needed by each, the "cost" of each according to some measure, the amount of execution time the process has already used, whether or not the process has interacted with files or done I/O, etc.