

Homework 2

Submit your *typed* answers at the start of class on Friday, June 6th. Figures and equations can be done by hand. Remember to submit your homeworks with a cover page that includes your name, netid and student number.

You are permitted to discuss homework questions with other students, but you must write out the answers yourself and give the names of the students you discussed the homework with. We will not look favorably on answers that are copied from online sources.

1. List the actions taken to context-switch between (i) two user-level threads; (ii) two kernel-level threads. Point out why context switching between two kernel threads might be more expensive than user threads.
2. In Linux, threads are implemented on top of processes that share a single address space. One of the reasons cited for this decision is the already low overhead for context-switching between processes. Suppose you take an operating system which doesn't provide threads, and add threads by writing a `thread_fork` operation which does the following: create a new process, but make it share the same address space as the calling process. You make no other changes, and are disappointed to see that despite the process context-switching time being fast, when you switch between "threads" which share the same address space, there is no improvement on the original performance. What could you do to improve the performance? (Hint: List all the actions the operating system has to do when context-switching between processes first).
3. Give pseudocode for implementing a semaphore using a monitor and condition variables.
4. Silberschatz 7.8 (the Sleeping-Barber problem); give a solution using semaphores.
5. Silberschatz 7.9 (the Cigarette-Smokers problem); give a solution using semaphores.
6. An old bridge has only one lane and can only hold at most 3 cars at a time without risking collapse. Suppose we want to write a computer simulation for the bridge, in which each car is represented by a separate thread. Write two functions, `ArriveBridge(int direction)` and `ExitBridge()` that control traffic, so that at any given time, there are at most 3 cars on the bridge, and all of them are going the same direction. A car calls `ArriveBridge` when it arrives at the bridge and wants to go in the specified direction (0 or 1); `ArriveBridge` should not return until the car is allowed to get on the bridge. A car calls `ExitBridge` when it gets off the bridge, potentially allowing other cars to get on. Give an implementation of the functions using semaphores. Don't worry about starving cars trying to go in one direction; just make sure cars are always on the bridge when they can be.
7. Section 7.4.2 in Silberschatz describes spinlocks. In many operating systems for multiprocessor computers, locks are implemented as a hybrid of spinlocks and blocking. Explain what the trade-off is between the two techniques and suggest a reasonable way of combining them efficiently.