CS 4431 Operating Systems

Midterm Examination

1. Assume that t_1, t_2 and t_3 are three kernel threads. t_1 and t_2 execute in one address space and t_3 executes in a different address space. Compare the context switch cost when a processor is switched from t_1 to t_2, with the cost of switching it from t_1 to t_3. For full credit, enumerate the actions that are performed due to context switches in the two cases and point out the actions which explain the differences in the costs. (15 pts.)

2. In the second programming assignment, to implement synchronization across threads, semaphore operations P() and V() were implemented. Instead of semaphore operations, assume that the following two calls have to be implemented:

   - `Mycondition_wait(cond: condition)`: This operation is executed by a thread when it wants to block itself. The thread is moved to the queue corresponding to the condition variable.
   - `Mycondition_signal(cond: condition)`: This operation is executed by a thread to unblock another thread that is on the condition queue identified by `cond`. The semantics of this operation is identical to the `condition_signal` operation of the threads library that you used to program the second assignment.

   Provide pseudo code for the implementation of these two operations. It should be possible to integrate this implementation with the code that you have already developed in the second programming assignment. Explain what new data structures need to be added and when and how they are accessed. (20 pts.)

3. In the spin lock implementations, we argued that there should be minimal lock transfer latency. Give a precise definition of lock transfer latency. (5 pts.)

   Discuss the two delay based implementations of spin locks on a shared bus multiprocessor machine with caches. Which implementation is expected to have a lower lock transfer latency? You can assume that an invalidation protocol is used to keep the caches coherent. (20 pts.)

4. Consider the following two statements that can appear in the code of a serializer.

   S1: `enqueue`(q) `until` B

   S2: `while not` B do `enqueue`(q) `until true`

If we replace statement `S1` in a serializer by `S2`, will the modified serializer still grant the resource to requesting threads in the same order as before? If not, explain how it will change. (15 pts.)

5. You have been hired to implement a highly simplified database for a bank. The bank has n customer accounts and a customer can request the following three operation: `balance, deposit` and `withdraw`. `balance` returns the current balance in the customer's account and the other two operations also have the obvious semantics. Customer requests can be received concurrently and the same customer can have multiple pending requests.

   Can we implement this database inside a monitor? Explain what level of concurrency is desirable in this application and can this level of concurrency be achieved if the database is encapsulated inside a monitor? (hint: can this be viewed as an instance of the readers-writers problem?) (10 pts.)

   Give a monitor based implementation of the bank database. You should allow the maximum level of concurrency that is possible. Discuss your implementation approach and provide pseudo-code fragments for the various operations implemented by the monitor. (15 pts.)