CSC 369H1 F L0101

Question 1. [7 MARKS]

Each of the following statements is **false**. In one sentence, explain why. (If you disagree that the statement is false, provide your reasoning.)

Part (a) [1 MARK] A system call is invoked in the same manner as a user function.

Part (b) [1 MARK] Malloc and free must maintain data structures that track all free and allocated blocks of memory. Otherwise, memory could be leaked (lost).

Part (c) [1 MARK] If two threads access a shared variable at the same time, there will be a concurrency error.

Part (d) [1 MARK] As seen in Exercise 6, hand-over-hand locking will always be slower than a single lock for the entire data structure.

Part (e) [1 MARK] Interactive systems should use non-preemptive scheduling algorithms.

Part (f) [1 MARK] If we can reasonably predict how long a job will run, then Shortest Job First (or Shortest Time to Completion First) is a good policy for minimizing response time.

Part (g) [1 MARK] A trap instruction is a privileged operation.

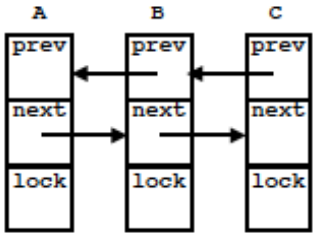
CSC369H1 F L0101

Question 2. [3 MARKS]

Suppose you have implemented a doubly-linked list data structure. Threads begin traversing the list either from the head node or the tail node. Is it possible to enforce mutual exclusion by using one lock for the head and one lock for the tail? Explain.

Question 3. [4 MARKS]

Consider an implementation of a doubly-linked list insert function where synchronization is implemented using a lock for each node. To change any pointer value, the lock must be held for the node containing the pointer, and the node the pointer points to. For example, to change `A->next` in the example below, we would need to hold `A->lock` and `B->lock`.



A `remove` function takes a valid pointer to a node in the doubly linked list, and removes that node from the list. To remove node B in the above example we would need to hold the locks for A, B, and C.

Part (a) [2 MARKS]

Does this implementation of `remove` satisfy mutual exclusion? In other words, could another thread modify any `next` or `prev` pointers so that the list is not correctly linked together? Explain your thinking.

Part (b) [2 MARKS]

Explain what can go wrong with this approach and why.

CSC 369H1 F L0101

Question 4. [7 MARKS]

Consider the following multi-level queue algorithm:

- Processes in queue 0 are scheduled using round robin and a time quantum of 2 time units.
- Processes in queue 1 are scheduling using round robin and a time quantum of 4 time units.
- Processes that use their full time quantum move to (or stay in) queue 1.
- Processes that do not use their full time quantum move to (or stay in) queue 0.
- The scheduler only chooses processes from queue 1 when queue 0 is empty.

Part (a) [2 MARKS] When processes arrive, should they begin in queue 0 or queue 1? Explain your answer with an example.

Part (b) [2 MARKS]

Explain what benefit, if any, is gained by giving queue 1 a longer time quantum than queue 0.

Part (c) [1 MARK] Explain how starvation could occur with an example.

Part (d) [2 MARKS] Propose a modification to this algorithm that would prevent starvation.