

## Midterm Review Questions

The questions following cover some many of the topics that will be on the midterm. Please look over these questions and try to solve them before class on Friday.

## 1. Concurrency

```

Thread A
for (i=0; i<5; i++) {
    x = x + 1;
}

Thread B
for (j=0; j<5; j++) {
    x = x + 2;
}

```

Assume a single-processor system, that load and store are atomic, that  $x$  is initialized to 0 *before either thread starts*, and that  $x$  must be loaded into a register before being incremented (and stored back to memory afterwards). The following questions consider the final value of  $x$  after both threads have completed.

- Give a *concise* proof why  $x \leq 15$  when both threads have completed.
  - Give a *concise* proof why  $x \neq 1$  when both threads have completed.
  - Suppose we replace  $x=x+2$  in Thread B with an atomic double increment operation `atomicIncr2(x)` that cannot be preempted while being executed. What are all the possible final values of  $x$ ? Explain.
  - What needs to be saved and restored on a context switch between two threads in the same process? What if the two threads are in different processes? Be explicit.
  - Under what circumstances can a multithreaded program complete more quickly than a non-multithreaded program? Keep in mind that multithreading has context-switch overhead associated with it.
2. Consider the following bounded-buffer solutions for a Coke machine using three semaphores (`mutex`, `emptyBuffers`, and `fullBuffers`). Given each of the following variations, say whether it is correct or incorrect. If you say correct, explain any of the advantages and disadvantages of the new code. If you say incorrect, explain what could go wrong (i.e., trace through an example where it does not behave properly).

a.

```

Producer () {
    mutex.P();
    emptyBuffers.P();
    put 1 coke in machine;
    fullBuffers.V();
    mutex.V();
}

Consumer () {
    mutex.P();
    fullBuffers.P();
    take 1 coke from machine;
    emptyBuffers.V();
    mutex.V();
}

```

b.

```

Producer () {
    mutex.P();
    emptyBuffers.P();
    put 1 coke in machine;
    fullBuffers.V();
    mutex.V();
}

Consumer () {
    fullBuffers.P();
    mutex.P();
    take 1 coke from machine;
    mutex.V();
    emptyBuffers.V();
}

```

```

c.
Producer () {
    emptyBuffers.P();
    mutex.P();
    put 1 coke in machine;
    fullBuffers.V();
    mutex.V();
}

Consumer() {
    fullBuffers.P();
    mutex.P();
    take 1 coke from machine;
    emptyBuffers.V();
    mutex.V();
}

```

3. Suppose there are two types of dining philosophers. One type always picks up his left fork first and the other type always picks up his right fork first – call these a lefty and a righty. Each type executes consecutive “wait”s on their forks (left followed by right for lefties, and the other way around for righties), eats, then does “signal”s on the forks in reverse order of the waits (right followed by left for lefties, and the other way around for the righties).
  - a. Does a seating arrangement of lefties and righties with at least one of each avoid deadlock? Why?
  - b. Does it prevent starvation? Why?
4. Some Linux atomic operations do not involve two accesses to a variable, such as `atomic_read(atomic_t *v)`. A simple read operation is obviously atomic in any architecture. Therefore, why is this operation added to the repertoire of atomic operations?
5. You are designing a data structure for efficient dictionary lookup in a multithreaded application. The design uses a hash table that consists of an array of pointers each corresponding to a hash bin. The array has 1001 elements, and a hash function takes an item to be searched and computes an entry between 0 and 1000. The pointer at the computed entry is either null, in which case the item is not found, or it points to a doubly linked list of items that you would search sequentially to see if any of them matches the item you are searching for. There are three functions defined on the hash table: Insertion (if an item is not there already), Lookup (to see if an item is there), and deletion (to remove an item from the table). Considering the need for synchronization, would you:
  1. Use a mutex over the entire table?
  2. Use a mutex over each hash bin?
  3. Use a mutex over each hash bin and a mutex over each element in the doubly linked list?

Justify your answers.

6. You are asked to implement a one-shot barrier for N threads. Each thread executes code of the following manner:

```

Barrier *b;
...
Arriveatbarrier(b);

```

Basically each of the first N-1 thread to call **Arriveatbarrier(b)** will block until the last (Nth) thread has arrived there. Once that happens, all threads will be released simultaneously to proceed.

One-shot means this code will be executed exactly once by each thread.

- a. Write an implementation using semaphores and mutexes.
- b. Does your solution work if the barrier is reused multiple times by threads (i.e., it is not a one-shot barrier)?

7. Most round-robin schedulers use a fixed size quantum. Give an argument in favor of a small quantum, and another in favor of a larger one. Compare the contrast the types of systems and jobs to which the arguments apply.
8. Five batch jobs A through E, arrive at a computer center at essentially the same time (externally defined) priorities are 6, 3, 7, 9, and 4 respectively. Their running times are 15, 9, 3, 6, and 12 minutes respectively. A lower value corresponds to a higher priority. For each of the following scheduling algorithms, determine the turnaround time for each process and the average turnaround for all processes. Ignore context switch overhead. Assume only one job runs at a time until it finishes and that all jobs are completely processor bound.
  - a. Round Robin with a quantum of 1 minute.
  - b. Priority
  - c. FCFS (run in order 15, 9, 3, 6, 12)
  - d. SJF.
9. Classify the relation between each of the following pairs into either “if one increases, the other always increases” or “if one decreases, the other always increases” or “no such relation exists always.”
  - a. Throughput and Waiting Time.
  - b. Waiting Time and Turnaround Time.
  - c. Response time and Turnaround Time.
  - d. Throughput and Turnaround Time.
10. (No solution will be given – do this at home) You have been hired to coordinate people trying to cross a river. There is only a single boat, capable of holding at most three people. The boat will sink if more than three people board it at a time. Each person is modeled as a separate thread, executing the function below:

```

Person(int location)
/* location is either 0 or 1; */
/* 0 = left bank, 1 = right bank of the river */
{
    ArriveAtBoat(location);
    BoardBoatAndCrossRiver(location);
    GetOffOfBoat(location);
}

```

Write code for the three functions above.

11. (No solution will be given – do this at home) Implement semaphore wait and semaphore signal functions using the test\_and\_set lock primitive. Note: you cannot use the “atomic” magic word.
12. (No solution will be given – do this at home) You are given an XCHG (exchange) hardware instruction that atomically exchanges two memory addresses. Implement a test\_and\_set lock using the XCHG primitive. Note: you cannot use the “atomic” magic word.