CSc 452: Principles of Operating Systems
Fall 24 (Lewis)

Test 1

Thu 10 Oct 2024

Name: _____ NetID: _____

Question	Points	Score
Page 1	20	
Page 2	20	
Page 3	20	
Page 4	20	
All False	20	
Total:	100	

NetID: _____

1. (a) (10 points) What is a "context switch?"

(b) (6 points) If we didn't have a timer interrupt, what major problem would we face in our operating systems?

(c) (4 points) What is a non-blocking operation, on a file, mailbox, or other communication mechanism?

2. (a) (6 points) When you open, read, write, or delete a file, you must perform a syscall into the kernel. Explain why this is - why can't you just access the file yourself, from inside user mode?

(b) (6 points) Explain the difference between a mutex, a critical section, and a lock.

(c) (8 points) What is an atomic instruction? What new capability does it provide to our programs, more than ordinary load, store, and arithmetic instructions?

3. (a) (10 points) What does the word "spin" in spinlock refer to? What does it tell you about how the lock works?

(b) (5 points) In the context of concurrency control systems such as locking, what is the difference between "starvation" and "deadlock?"

(c) (5 points) In class, we discuseed the (very surpising!) fact that the dispatcher must hold interrupts disabled even while it performs the context switch. Explain what sort of problem might arise if we restored interrupts before the switch.

4. (a) (8 points) Suppose that you have two concurrent processes, each incrementing the same variable, many times - but they don't use locks or atomic instructions to prevent races:

```
for (int i=0; i<10*1000*1000; i++)
    x++;
```

Now, suppose that a third process is reading the value of `x` over and over, just watching to see how it might change. Explain how it might be possible for `x` to actually go **down** in value (perhaps by a lot), from time to time. (Ignore the fact that `x` might overflow, that doesn't count.)

(b) (8 points) In our project, we disabled interrupts in our functions, in order to solve race conditions. Explain why this was "as good as" grabbing a lock - provided that we are only running on a single CPU.

(c) (4 points) Why would disabling interrupts be insufficient, as a replacement for a lock, if we had more than one CPU?

NetID: _____

5. Each of the statements below is **False.** Explain why.

(a) (5 points) Because user mode code cannot disable interrupts, the most critical calculations must be done quickly, in order to prevent race conditions with other proceses.

(b) (5 points) If a process doesn't access any devices (such as reading from the disk), we can accurately predict how fast it will run, because it won't be interrupted.

(c) (5 points) In our Phase 1 scheduler, we had absolute priorities - meaning that the highest priority process always ran. If one process created a child that was higher priority, the parent would not get any time on the CPU until the child dies.

(d) (5 points) When a process calls `quit()`, the kernel will record its exit status and then context-switch to its parent.