CPS 210 Midterm

Hallowe'en 1996

Instructions. This exam has two parts. Answer all questions according to the instructions for each part. This is a closed book examination. You have 70 minutes. No tricks, no treats. Total points: 150.

Part 1: True or False

Indicate whether each of the following statements is true or false. Each question in this part is worth 5 points (50 points total). An effort has been made to avoid ambiguity, but you may elaborate on your answer if you find a question unclear or ambiguous in some way. Otherwise, you need not explain your answers.

1. Spinlocks and atomic sequences (e.g., as in Bershad/Redell/Ellis) are unnecessary in Unix kernels for uniprocessor systems.
2. User-level threads are often called "lightweight" (i.e., less expensive than kernel-supported threads) in part because multiple user threads can share the same kernel stack.
3. To avoid deadlock, kernel code may need to disable interrupts before acquiring a spinlock.
4. A CPU executing in user mode returns to kernel mode only if it executes a system call trap instruction.
5. Most Unix kernels are nonpreemptive in order to simplify synchronization of processes executing in kernel mode.
6. A Unix interrupt handler should never wake up a sleeping process, because it might fail to return from the interrupt.
7. A group of threads or processes can never deadlock if they acquire any needed resources according to some fixed ordering they all agree on.
8. Reader/writer locks can reduce lock contention, but they are prone to writer starvation, which can be fixed only at the cost of reducing the concurrency available for readers.
9. One purpose of memory management hardware (the MMU and/or TLB) is to allow the kernel to determine if a given virtual memory page has been referenced or modified.
10. The *fork* system call takes much longer to execute when called from a large program, because it must copy the program's code segments.

Part 2: Paragraphs

Select five of the six questions below and answer them. Your answer for each of the five questions you select should be ten sentences or less; diagrams and/or code examples may be useful as well. Longer answers *may* be truncated during grading, and besides that you'll run out of time, so think before you write. Each answer is worth 20 points (100 points total).

11. Compare and contrast the condition variable *wait/signal* interface with direct use of *sleep* and *wakeup* as implemented in a typical Unix kernel. Is the condition variable interface easier to use or more efficient? Why?

12. Discuss the relative merits of user-level thread packages vs. kernel-supported thread facilities. Why might using a user-level thread package result in a faster program? Why might it result in a slower program?

13. Argue for and against the following statement: *the primary concern of kernel interface design should be to provide an easy-to-use programming interface at the system call level.* Substantiate your answer with examples.

14. How does a file system buffer cache improve performance for file *reads*? Outline three scenarios to illustrate three ways in which the buffer cache reduces disk accesses or seek overhead in typical Unix systems.

15. Why are *fork* and *exec* different system calls in Unix? What flexibility or functionality would be lost if they were combined? What are the costs of leaving them separate?

16. How does a Unix process differ from a kernel-supported thread (e.g., "lightweight processes" or LWPs in Solaris)? Write down as many differences as you can in the space allowed, in order of decreasing importance.