

Name:

CS318, Midterm

Fall 2000

Pledge:

Directions:

- Please answer each question in the space provided. The amount of space should be sufficient for a correct answer. If you need more space, please use the backs of pages, and make a note to that effect. If you run out of space, exam books are provided at the front of the room.
- This exam is closed-book, closed-notes, and is covered by the Honor Code. Please write and sign the pledge after you finish your exam.
- There are a total of five sections, with the number of points for each shown by the question number.
- To be fair, I will try to avoid answering content-related questions during the exam, unless it's to correct a mistake on my part.
- If you feel that a question **requires** additional assumptions or information to answer, please state them. Your guiding principle should be *Occam's razor*, which loosely translated states that you should allow as few assumptions as necessary to explain the situation.
- Unless otherwise stated/implied, assume a C-like language running on a Unix-like operating system.
- Please write legibly

1. (8 points) True or False – under each statement, write true or false. If you believe that there is some confusion, feel free to defend your choice with a one-sentence or two-sentence explanation.

- User-level threads are faster on multiprocessors than kernel threads
- The first operation in Lamport’s fast mutual exclusion is a read
- Scheduling ready processes at random eliminates priority inversion
- In a virtual machine system, traps/interrupts to invoke system calls are delivered to the hypervisor
- Test-And-Set can be used to develop **efficient** Acquire/Release operations
- The join operation terminates a thread
- Scheduler information is stored on the kernel’s stack
- User processes use signals to inform the kernel about state changes

2. (12 points) Short answer – for each question, provide a brief answer, no more than four sentences or so.

- What is the fundamental difference between a mutex and a semaphore? In what circumstances is each appropriate?

- What are the fundamental differences between monolithic and microkernel operating systems? What are the benefits/drawbacks of each approach?

- Describe the per-thread information needed for user-level threads. What are the benefits and drawbacks of using user-level threads versus kernel-managed threads?

- Rate the following four scenarios from highest to lowest with regard to useful work done per unit time, and explain why: blocking on a contended lock, spinning on a contended lock, blocking on an uncontended lock, or spinning on an uncontended lock?

3a. (15 points) Your friend has just told you about a brand new processor with 256 general-purpose registers, and wants you to develop the OS for it. Using the mini-kernel from your projects as a model, please answer the following:

- what parts of your system **will** be affected, how will they be affected, and why will they be affected this way
- what parts of your system **may** be affected, under what circumstances might they be affected, and what effect this will have

Please be concise yet complete. For example, the following paragraph presents a reasonable analysis for non-OS issues: “Instruction encoding must change, either by eliminating three-operand instructions, or increasing the size of each instruction. This change stems from the fact that adding registers requires using more bits to specify each register. The code size is likely to increase due to either requiring more instructions or more memory to contain the same number of instructions. Instruction fetch bandwidth increases as a result.”

3b. (5 points) Can you speed up any aspect of a system call using those registers? If so, state what can be changed and considerations in doing so.

4. (20 points) Homer Simpson likes to sleep and eat doughnuts. When he's not doing one, he's doing the other, and he's so adept that the smell of a doughnut on his plate will wake him. Marge Simpson, being a kind-hearted soul, will place a new doughnut on Homer's plate if she senses that it is empty. Otherwise, she naps as well.

Write safe, efficient code to model the behavior for Homer and Marge using appropriate minimal synchronization. State your assumptions. Full credit will only be given for minimal synchronization. Minimal here can be taken to reflect the order presented in the book. Hint: spinning isn't efficient, disabling interrupts isn't safe.

5. (20 points) After a hard night of programming, you have a dream wherein a mid-level manager at your upcoming summer job suggests the following scheduling policy for a pre-emptive system:

A primitive system has a hard disk that can only be used by one process at a time, so it is protected by a lock. Whenever a process blocks on a lock, schedule the process holding the lock if it is ready to run. Whenever a process releases a lock, schedule a process waiting for the lock, if any exist. Otherwise, randomly pick a ready process and run it.

Discuss the benefits and drawbacks of this system - characterize its behavior under various relevant load scenarios.