CS140 Operating Systems and Systems Programming
Midterm Exam

October 31st, 2003

(Total time = 50 minutes, Total Points = 50)

Name: (please print)_____

In recognition of and in the spirit of the Stanford University Honor Code, I certify that I will neither give nor receive unpermitted aid on this exam.

Signature:_____

This examination is close notes and close book. You may not collaborate in any manner on this exam. You have 50 minutes to complete the exam. Before starting, please check to make sure that you have all 10 pages.

1	
2	
3	
Total	

Name:_____

1. CPU Scheduling (14 points total)

(2 points) What advantages does a preemptive CPU scheduling algorithm have over a non-preemptive one?

(3 points) Why do different levels of a multi-level feedback queue CPU scheduler have different time quantum?

(3 points) Why do we say that round robin CPU scheduling does poorly when faced with jobs of equal length?

(3 points) What is the difference between time-sharing and space-sharing a resource? Give examples.

(3 points) Programs such as just-in-time compilers for Java increase performance by using runtime code generation to generate better code to run. Explain how runtime code generation can actually hurt performance.

2. Memory (17 points total)

(4 points) At one time, virtual memory system designers were advised to bias page replacement algorithms against modified pages in favor of those that have not been modified. The result of this suggestion was the unfortunate behavior of program code pages (which tend to be some of the only pages that are not modified) being kicked out of memory before other pages. Describe the rationale for the original suggestion of paging unmodified pages first.

(3 points) The N-bit clock page replacement algorithm described in the lecture notes computed the page use_count as:

use_count = (use_bit << N-1) | (use_count >> 2)

where use_bit is the reference/use bit for the page.
Describe the motivation for computing the use_count like this.

(2 points) Explain how adding segmentation to a pure paging system can reduce the size of the page tables.

(4 points) The Intel x86 architecture started using pure segmentation and was extended to use segmentation and paging. Given an architecture with a base-and-bound system, would there be any benefit to adding paging (i.e.. base-and-bound and paging)? Explain your answer.

(2 points) Would it ever make sense for two processes to share the same page table?

(2 points) Can a first fit memory allocator ever have less fragmentation than a best fit one?

3. Concurrency and synchronization (19 points)

(3 points) Your partner says he knows how to recover from a deadlock condition in Nachos. He simply keeps track of the locks acquired by each process and when a deadlock is detected he kills the processes involved in the deadlock and releases the locks they are holding. Will this work? Why or why not?

(4 points) (a) Explain how a compare-and-swap instruction can be used to increment a shared integer variable without using other synchronization operations such as locks or semaphores. (b) Will the scheme you describe work if an interrupt handler also updates the shared integer variable? (i.e. will it work if the variable is incremented both in normal code and in an interrupt handler).

(2 points) Does a deadlock prevention algorithm necessarily prevent starvation as well? Justify your answer.

(2 points) What is the optimal spin before-blocking time for a spin lock on a uniprocessor? How about a multiprocessor?

(3 points) Most operating systems today use a scheme that supports multiple address spaces, each with multiple threads of control. Assume you are given two operating systems: OS-A which supports multiple addresses spaces each with one thread of control and OS-B which is a single address space OS with multiple threads of control. Which OS (OS-A or OS-B) would it be easier to modify to support multiple address spaces each with multiple threads of control? Justify your answer.

(2 points) Can a system that uses only spin locks (i.e. no blocking locks or semaphores) deadlock? Justify your answer.

(3 points) Although both procedure calls and thread switches effectively switch the execution context of the CPU, procedure calls are normally much faster than thread switches. Explain why this is the case.