CS140 Operating Systems and Systems Programming
Midterm Exam

October 28th, 2002

(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 9 pages.

1	
2	
3	
4	
5	
6	
Total	

Name:________________

1. CPU Scheduling (6 points – 2 points each)

(a) Explain how the effects of *priority donation* in a priority-based CPU scheduler are approximated in a system that uses *lottery scheduling*.

(b) In a multi-level feedback queue (MLFQ) CPU scheduler, the length of a process's timeslice depends on the process's priority level. Explain the reasoning behind this property of a MLFQ scheduler.

(c) You and your Nachos partner are arguing over how to implement a fair CPU scheduler. Your partner claims that round-robin with a reasonable timeslice is the fairest possible algorithm. Is your partner correct? Justify your answer.

2. Synchronization and deadlock (10 points – 2 points each)

(a) Explain why an OS designer needs to worry about deadlock involving physical memory but not the CPU.

(b) Is it possible to have a deadlock involving virtual memory? Justify your answer.

(c) Many operating systems designed to run only on uniproccessors use disabling of interrupts to create critical sections in the code. Explain how this interrupt disabling prevents multiple threads from entering the critical section.

Name____________________

(d) Explain why disabling of interrupts to implement critical sections will not work on a multiprocessor.

(e) When running on a multiprocessor, would it ever make sense to have a spin lock also disable interrupts for the duration of a critical? Explain your answer.

3. Linking & code generation (6 points – 3 points each)

(a) Explain why a system that uses dynamically linked libraries (DLLs) might have the first call to a library function take much long than subsequent calls.

(b) Explain how dynamically generated code can be faster than more traditional statically compiled code even thought it has the additional overhead of runtime code generation.

4. Operating System Structure (8 points)

It is common for operating systems on modern machines to map the operating system kernel into the top of all the address spaces. This approach allows the OS kernel to directly access data structures such as system call arguments that are in the application's portion of the address space. Although these addresses can be used in the kernel, care must be taken when an OS uses this approach. Describe two of the problems that can occur if the OS is not careful with these application-provided addresses.

5. Memory management (12 points)

Assume that you are working on the operating system for a computer system that uses a segmentation scheme that works as follows:

The machine has a 32bit address where the top 2 bits are the segment number and the lower 30 bits are the offset into the segment. The hardware segment table has 4 entries each with a valid bit, protection bits (read/write, user/supervisor), 32 bit base and 32 bit length. To save having to store all the bits, the lower 12 bits of the base and the length must be zero.

The architects of the machine are thinking of replacing the segmentation with a pure paging scheme that uses a 4 kilobyte page size and a flat page table per process. The page table has 2^{20} entries each with a 20bit physical page number, a valid bit, protection bits (read/write, user/supervisor), and the normal reference and dirty bits.

You are asked if you could add a layer to the operating system that would emulate the old segmentation approach on top of this new paging hardware.

(a) (8 points) Describe how you do this mapping of segmentation on to paging. Show the page table you would build for the following segmentation table:

Seg #	Valid	Protection	Base	Length
0	Yes	Read only	0x0000000	0x800000
1	Yes	Read/Write	0x0800000	0x800000
2	No	No Access	0	0
3	Yes	Read/Write Sup Only	0x1000000	0x1000000

(Hint: 0x800000 is 8MB and 0x1000000 is 16MB)

(b) (4 points) Let’s say you were required to do a “native” port of the operating system to take advantage of this paging system. What would be your biggest complaint about the paging support in the memory management unit.

6. Paging (8 points)

(a) (5 points) Given a modern computer system such as the Sweet Hall machines, would you expect a disk used only as the backing store for the virtual memory system to have:

1. More page read than page write requests.
2. About the same number of read and write requests.
3. More write than read requests.

Justify your answer.

(b) (3 points) Your Nachos partner claims that any thrashing problem of a system can be fixed by adding more memory to the computer. Assuming that you can add infinite memory to a computer, is this a valid claim? If so, justify your answer. If not, explain why not.