CS140 Operating Systems and Systems Programming
Midterm Exam

October 27, 2006

(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 8 pages.

1	
2	
3	
4	
5	
6	
7	
Total	

Name:_______________

1. (8 points) At a recent conference keynote talk the speaker said that modern operating systems use “page flipping” to avoid copying data. Describe what page flipping is. Hint: Assume you have an operating system that supports system calls that allow processes (each with its own address space) to pass messages between each other (e.g. send/receive message on a pipe-like abstraction.) Consider how an operating system could implement this service without physically touching the bytes of the message. In other words, the message bytes are moved from one process’s address space to another without the bytes being read or written as part of the operation.

2. (7 points) Assume you have a system with four threads of control, a shared variable, and a lock shared between the threads. The code was written in such a way that any accesses by a thread to the shared variable would always have the lock held by that thread. In other words, a thread always holds the lock when it accesses the shared variable. Can this system have a race condition involving the shared variable? Justify your answer.

3. (6 points) Consider the following segments that make up a process's address space: (a) **code**, (b) **data**, and (c) **stack**. State what parts of the segments (if any) are derived from the program's object file.

4. (8 points) When programming scientific code with multiple threads, sometimes it is useful to have all threads rendezvous at a place in the code. This is normally done with an operation called a "barrier". It works so that when threads call the Barrier() function, none return until all threads have called the function. For example you might find code that looks like:

```
    {
        ThreadsRunInParallel();   // Threads may finish at different times
        Barrier(ThreadID());
        ThreadsDoSomethingElseInParallel(); //Threads should start here together
    }
```

Your job is to write the Barrier() function using only semaphores. You are not permitted to use shared variables other than the semaphores. Be sure to show your initial values for the semaphore. You can assume there is a global constant NUM_THREADS that indicates the number of threads in the system. Thread IDs are integers starting at 0 and going to NUM_THREADS-1. NUM_THREADS is less than 100. To simplify things you can assume that Barrier() is only called once and all threads participate in the barrier.

Semaphore declaration and initialization:

```
void Barrier(int threadID) {
```

5. (7 points) Assume you given a uniprocessor system with one gigabyte of memory and a 300 gigabyte disk. The OS on the machine has a demand paged virtual memory system with a local replacement policy and a multi-level feedback queue (MLFQ) CPU scheduler. On the system there are two compute-intensive jobs running: Job-A and Job-B. Job-A has working set of 50 gigabytes while Job-B has a working set of 100 megabytes. Assume you left the system to run for a while until it reached a steady state with both jobs running.
(a) Which job would you expect to have a higher CPU scheduling priority from the MLFQ scheduler?
(b) Assume you add a second CPU to system, how would this affect the priorities of the jobs?
(c) Assume you switch from a local to a global replacement policy, how does this change affect the priorities of the jobs?
Be sure to justify your answer and state any assumptions you make.

6. (8 points) In this question we are asking you to compare static and dynamic linking.
   (a) Describe the advantages of dynamic linking over static linking.
   (b) Describe the problems in dynamic linking that are not present in static linking.

7. (6 points) Assume you have a system in which there is circularity in the wait-for graph of the system. Is this a definitive sign of a deadlock? Justify your answer.