University of California
College of Engineering
Computer Science Division- EECS

Spring 2003

Anthony D. Joseph

Midterm Exam

March 13, 2003

CS162 Operating Systems

Your Name:	
SID AND 162 Login:	
TA:	
Discussion Section:	

General Information:

This is a **closed book and notes** examination. You have two hours to answer as many questions as possible. The number in parentheses at the beginning of each question indicates the number of points given to the question; there are 100 points in all. You should read **all** of the questions before starting the exam, as some of the questions are substantially more time consuming.

Write all of your answers directly on this paper. Make your answers as concise as possible. If there is something in a question that you believe is open to interpretation, then please ask us about it!

Good Luck!!

Problem	**Possible**	**Score**
1	28	
2	21	
3	12	
4	27	
5	12	
Total	**100**	

1. (28 points total) Short answer questions:

   a. (9 points) List any THREE major components of most modern operating systems (e.g., Unix, Solaris, WindowsNT, Windows2000, or WindowsXP) and briefly describe their role of each.

   i)

   ii)

   iii)

   b. (9 points) Give a definition of a counting semaphore, and list and describe the valid operations.

   c. (4 points) List the conditions for deadlock.

d. (6 points) Provide definitions for internal and external fragmentation. Draw a picture show internal fragmentation in a pure paging system (not paged segmentation or segmented paging, but only paging.) Draw a picture showing external fragmentation in a pure segmentation system.

2. (21 points total) Processor Scheduling. Here is a table of processes and their associated running times. All of the processes arrive in numerical order at time 0.

Process ID	**CPU Running Time**
Process 1	6
Process 2	1
Process 3	2
Process 4	4
Process 5	3

a. (9 points) Show the scheduling order for these processes under First-In-First-Out (FIFO), Shortest-Job First (SIF), and Round-Robin (RR) scheduling with a timeslice quantum = 1 time unit.

Time	**FIFO**	**SJF**	**RR**
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

b. (12 points) For each process in each schedule above, indicate the queue wait time and turnaround time (TRT).

Scheduler	Process 1	Process 2	Process 3	Process 4	Process 5
FIFO queue wait					
FIFO TRT					
SJF queue wait					
SJF TRT					
RR queue wait					
RR TRT					

The queue wait time is the total time a thread spends in the wait queue.

3. (12 points total) Two-level Virtual Memory. For each of the following two-level virtual memory addressing schemes, explain, in one or two sentences how the scheme works.

a. Virtual address format:

Paging Level 1	Paging Level 2	Offset

b. Virtual address format:

Paging Level 1	Segment Level 2	Offset

c. Virtual address format:

Segment Level 1	Paging Level 2	Offset

d. Virtual address format:

Segment Level 1	Segment Level 2	Offset

4. (27 points total) Two-Level Page-based virtual Addressing. Consider a 32-bit machine with a multi-level virtual memory system with 32-bit pointers and 4096-byte pages that supports two-levels of page tables. All Page Table Entries (PTEs) are 4 bytes.

   a. (6 points) Show the complete format of a virtual address.

   b. (6 points) Explain the steps the hardware takes in translating a virtual address to a physical address for this scheme (do not worry about supporting paging to disk).

   c. (3 points) How many memory operations are required to read or write a single 32-bit word?

d. (4 points) List the fields of a Page Table Entry (PTE).

e. (6 points) How much physical memory is needed for a process with one page of virtual memory?

f. (2 points) What happens in the virtual memory subsystem on a context switch?

The Overwhelming Might of the US Military

The flowing is the transcript of an actual radio conversation that took place in October 1995, off the coast pf England. The British Ministry of Defense recently released the transcript:

British: Calling unknown radar constant at position *****, please divert your course 15^0 to the south to avoid a collision.

Americans: Recommend you divert your course 150 to the north to avoid a collision.

British: Negative. Yu will have to divert your course 150 to avoid a collision.

Americans: This is the Captain of a US Navy ship. I say again, divert your course north.

British: Negative, I say again. You will have to divert your course.

Americans: This is the captain of the aircraft carrier USS Lincoln, the second largest ship in the United States' Atlantic Fleet. Wee are accompanied by numerous support vessels. I demand that you change your course 150 north. That's 150 north, or countermeasures will be undertaken to ensure the safety of this ship.

British: This is a lighthouse. Your move …

We can only assume that the US ships adjusted their course south.

5. (12 points total) Concurrency problem: In parallel programs (one multi-threaded process), a common design methodology is to perform processing in sequential stages. All of the threads work independently during each stage, but they must synchronize at the end of each stage at a synchronization point called a *barrier*. If a thread reaches the barrier before all other threads have arrived, ii waits. When all threads reach the barrier, they are notified and can begin execution on the next phase of the computation.

   There are three complications to barriers. First there is no master thread that controls the threads, waits for each of them to reach the barrier, and then tells them to re-start. Instead the threads must monitor themselves and determine when they should wait or proceed. Second, for many dynamics programs, the number of threads that will be created during the lifetime of the parallel program is unknown in advance, since a thread can spawn another thread, which will start in the same program stage as the thread that created it. Third, a thread may end before the barrier. In all cases, all threads must synchronize at the barrier before the processing is allowed to proceed to the next phase.

   a. (10 points) Provide the pseudo-code for a monitor class called *Barrier* that enables this style of barrier synchronization. Your solution must support creation of a new thread (an additional thread that needs to synchronize), termination of a thread (one less thread that needs to synchronize), waiting when a thread reaches the barrier early, and releasing waiting threads when the last thread reaches the barrier. *Implement your solution using monitoring* (e.g., *wait( ), signal( ), and signalAll( )*).

   Your class must implement the following three methods: *threadCreated(), threadEnd(), barrierReached()*.
   Hint: this concept is very similar to Java *synchronized* objects.

b. (2 points) In your *barrierReached* method, which conditional statement (i.e., if, or while) did you use and why?