Computer Science and Engineering 3221.03

Midterm Test
Oct. 22 2008

Answer all questions in the space provided

Make sure that you have 8 pages

Student Last Name: _____

Student Given Name: _____

Student Id. No: _____

Question	Value	Score
A	50	
B	20	
C	45	

Question 1. [50 points]

1. [5 points] What will happen if you turn off the DMA on your computer.

2. [5 points] Name two solutions to well known problems that we discussed in class that do not satisfy the bounded waiting requirement.

3. [5 points] Why would someone prefer a solution to the two problems above that does not satisfy the bounded waiting requirement?

4. [5 points] Why does a server (like a web server) keep a pool of threads?

5. [5 points] What is the main advantage of user level threads?

6. [5 points] Name the technique that solves the problem of starvation in priority scheduling.

7. [5 points] What is the name of the pre-emptive version of SJF?

8. [5 points] Name the three threading models for the use of kernel and user threads.

9. [5 points] What is the issue with system calls like `read()` and user level threads?

10. [5 points] What is the main advantage of named pipes (FIFOs) vs the regular pipes?

Question 2.
[20 points]

1. [10 points] At various times the following 4 processes arrive at the ready queue.

Process	Burst	Arrival
P1	6	0
P2	1	1
P3	5	3
P4	2	4

What is the average turnaround time for SJF and SRTF scheduling policies.

2. [10 points] Which of the following schedule serializable? What is the equivalent serial schedule if there is one?

Schedule 1

Trans. 0	Trans. 1
read(A)	
	read(A)
write(A)	
	write(A)
read(B)	
write(B)	
	read(B)
	write(B)
and	

Schedule 2

Trans. 0	Trans. 1
write(A)	
	read(A)
read(A)	
read(B)	
write(B)	
	read(B)
	write(B)

Question 3.
[45 points]

1. [15 points] We implemented a monitor in class using semaphores. Now do the opposite: implement a semaphore using monitors. Write a monitor named `semon` that has two operations `wait()` and `signal()` that implement a semaphore.

2. [15 points] A variant of the readers-writers problem is the onion eaters and garlic eaters problem. There is a “room” where several onion eaters can be at the same time, and several garlic eaters can be at the same time but the garlic eaters do not enter when there are onion eaters and onion eaters do not enter when there are garlic eaters so the room has only one kind of processes. Write a program that solves the synchronization problem without starvation. You will use two semaphores `G` and `O` and when an eater enters, makes sure that he notifies any other eaters of his own kind that are waiting. You can write the code for garlic eaters only, since it is symmetric.

3. [15 points] Using **monitors**, write a solution to the five dining philosophers problem that allows at most four philosophers to the table. Provide methods `pickup()` and `putdown()`.