

Quiz 2

Course: cs241 - System Programming, CS Department

Date: February 10, 2006

Netid:

Name:

Note: Completion of quiz is an individual effort. The quiz takes 10 minutes. The student gets additional 5 points for taking the quiz. *Each question has ONLY ONE ANSWER!!!*

1. **(1 Point) Threads of the same task have the following characteristics: (I) Share the same address space, (II) Reduce context switching overhead; (III) Are protected from each other the same way as processes. Which thread characteristics are correct?**
 - a. I, II, III
 - b. I, III
 - c. I, II

2. **(1 Point) If a user process P runs two user-threads, T1(read-thread), and T2 (compute-thread) on top of a multi-threaded kernel, and thread T1 calls 'read' instruction, the following happens:**
 - a. user process P will block and another process Q will run
 - b. thread T1 will block and T2 will run
 - c. kernel threads will block

3. **(1 Point) Priority inversion between two processes, one with high priority and the other with low priority, that share a critical section, will cause the following problem:**
 - a. high priority process executes before low priority process and finishes faster than it should
 - b. low priority process executes before high priority process and finishes faster than it should
 - c. high priority process waits for low priority process (that holds the semaphore) to finish, but the low priority process never gets scheduled
 - d. low priority process changes priority temporary to the priority of the high priority process

4. **(1 Point) What is the minimum number of parameters to a 'message send' operation?**
 - a. 1
 - b. 2
 - c. 3

5. **(1 Point) Rendezvous message passing synchronization approach requires**
 - a. Double buffering
 - b. No buffering
 - c. A buffer queue
 - d. A place to meet

6. (1 Point) Consider the 2-process solution to the critical section problem ('*i*' refers to the current process and '*j*' is the other one):

```
repeat
  while turn <> i do no-op;
    <critical section>
  turn:=j;
  <remainder section>
until false;
```

This solution does NOT satisfy

- Mutual Exclusion
 - Progress
 - Bounded Waiting
 - Both (b) and (c)
 - None of the above
7. (1 Point) Consider the 2-process solution to the critical section ('*i*' refers to the current process and '*j*' is the other one):

```
repeat
  flag[i]:=true;
  turn := j;
  while (flag[j] and turn ==j) do no-op;
    <critical section>
  flag[i] := false;
  <remainder section>
until false;
```

This solution does NOT satisfy:

- Mutual Exclusion
 - Progress
 - Bounded Wait
 - Both (b) and (c)
 - None of the above
8. (1 Point) A process *P* tries to acquire a lock using the `Test_and_Set` instruction:

```
while (Test_and_Set(lock)) do no-op;
```

If the lock is busy (some other process has acquired it) and all interrupts are disabled, then

- process *P* blocks, waiting for the lock to be available
- process *P* wastes processor and memory cycles
- none of the above

9. (2 Points) Complete the pseudo-code solution to the bounded buffer problem, assuming names $Down()=P()$, $Up()=V()$, and given:

```
binary semaphore m;
semaphore f; /* counting semaphore*/
semaphore d; /* counting semaphore*/
m=1; f=n; d=0;

Producer(i):
Loop:           ...../* (1)
                ...../* (2)
                Deposit new data x into buffer
                V(m);
                ...../* (3)

EndLoop

Consumer(i):
Loop:           P(d);
                P(m);
                Consume next data y from buffer
                ...../* (4)
                ...../* (5)

EndLoop
```

- $P(f) \Rightarrow (1), P(m) \Rightarrow (2), V(f) \Rightarrow (3), V(m) \Rightarrow (4), V(d) \Rightarrow (5)$
- $P(f) \Rightarrow (1), P(m) \Rightarrow (2), V(d) \Rightarrow (3), V(m) \Rightarrow (4), V(f) \Rightarrow (5)$
- $P(d) \Rightarrow (1), P(m) \Rightarrow (2), V(d) \Rightarrow (3), V(m) \Rightarrow (4), V(d) \Rightarrow (5)$
- $P(d) \Rightarrow (1), P(m) \Rightarrow (2), V(f) \Rightarrow (3), V(m) \Rightarrow (4), V(d) \Rightarrow (5)$