CMSC 421/Section 0101: Operating Systems
Fall 2003

Krishna Sivalingam
14 October 2003
75 Minutes

Name:______________________________

Last 5 Digits of UMBC ID:______________________________

"I certify that my answers on this test are entirely from my efforts."

____

(Signature) (Printed Name)

This is a closed-book, closed-notes, closed-neighbor exam. Answer all questions in the space provided – you can write on both sides of the paper. If you use additional papers, please STAPLE them to this answerbook and write your name on each additional paper.

1. ______ (10 points)

2. ______ (15 points)

3. ______ (25 points)

4. ______ (25 points)

Total: ______ (75 points)

1. (10 points) **Basics:**

   List the various Operating System funtionalities that can classified under the category *Process Management* and *Memory Management*.

2. (15 points) **Processes and Threads:**

   Describe the sequence of actions, with the aid of a diagram if needed, that is carried by the kernel when the **fork()** and **pthread_create()** calls are invoked.

3. **Scheduling:**

   (a) (6) Explain the differences in the degree to which the following scheduling algorithms discriminate in favor of short processes: (a) FCFS; (b) Round Robin, and (c) Multilevel feedback queues

(b) (19) Consider the following set of processes, with the length of the CPU-Burst time given in milliseconds:

Process	Arrival Time	Burst Time
P1	0.0	8
P2	0.4	4
P3	1.0	3
P4	1.2	2

Determine the average turnaround time for each of the following algorithms:

(i) SJF with preemptive scheduling

(ii) Three level feedback queue scheduling with three queues as follows: Queue 1 has quantum of 2; Queue 2 has quantum of 4; and Queue 3 is FCFS. The scheduler first executes process in queue 1; only when queue 1 is empty does it schedule from queue 2.

4. The following algorithm has been proposed for an entry and exit protocol for critical sections. Assume there are two processes P_i and P_j. The shared variables are *turn* initialized to i, *flags* and a boolean array *wait* with one entry for each process. Assume initially all elements of this array are FALSE. The algorithm for P_i is:

```
ENTRY CODE:

flags[i] = BUSY;

if (flags[j] == BUSY) {

    wait[i] = TRUE; // I am waiting!

    while (( (flags[j] == BUSY) &&
           ( (turn == j)  || wait[j] == FALSE ))) {}; // Busy Wait
    wait[i] = FALSE;

}

EXECUTE CRITICAL SECTION CODE

EXIT CODE:

turn = j;
flags[i] = FREE;
```

(a) [12 points] Prove or disprove that the progress condition is satisfied by the above solution.

(b) [12 points] Prove or disprove that the mutual exclusion condition is satisfied by the above solution.

YOU CAN WRITE IN THIS PAGE TOO.