Name:_____ Student ID:_____

CMPSCI 377: Operating Systems
Spring 2019, Midterm 1: Processes, Threads, Scheduling, and Synchronization

• Do not forget to put down your name and student number
• The exam is closed book and closed notes.
• Explain your answers clearly and be concise. Do not write long essays.

1. Introduction (25 pts total)

(a) What illusion does an operating system present to user programs and why? (Hint: it has the word “interface” in it) (5pts)

(b) What is the API that a user program uses to access Operating System resources? (5pts)

(c) Describe how the contents of a single memory address A behave when accessed by two processes on a machine and two threads of a single process. (5pts)

(d) Describe the OS’s involvement in how does a user program reads from disk vs how it reads from memory. (5pts)

(e) Describe the difference between how the scheduler reacts to a process reading from disk vs. reading from memory. (5pts)

2. Process API (15 pts total)

(a) When forking, what (if any) are the differences between the memory of the original and forked process? (5 pts)

(b) Consider the following fragment of C code:

```
int x = 1;
while (x != 0) {
  printf("%d\n", x);
  x--;
}
printf("done!\n");
```

Which of the following is the correct process state transition for this code fragment? Assume that printf is I/O and will block. (5 pts)

A. READY, RUNNING, BLOCKED, RUNNING, BLOCKED, READY
B. RUNNING, BLOCKED, READY, RUNNING, READY, BLOCKED, RUNNING
C. READY, RUNNING, BLOCKED, READY, RUNNING, BLOCKED, READY, RUNNING
D. READY, BLOCKED, RUNNING, BLOCKED, READY, RUNNING, BLOCKED, RUNNING
E. BLOCKED, READY, RUNNING, BLOCKED, READY, RUNNING, BLOCKED, RUNNING

(c) What are the possible outputs from the program? (5 pts)

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

void forkexample()
{
    int x = 1;

    if (fork() == 0)
        printf("%d", ++x);
    else
        printf("%d", --x);
}
int main()
{
    forkexample();
    return 0;
}
```

Fill in as many as needed:

3. Processes (10 pts total)

(a) When a program invokes a system call, what happens to the contents of the registers. Be specific about who does what. (5pts)

(b) Where and when does a switch between two user processes occur? What is it called? (5pts)

4. Scheduling (20 pts total)

Assume that new processes enter the run queue at the back of the queue (so processes already in the system will take precedence). Since A and B arrive at the same time, assume A arrives first.

(a) FIFO scheduler (5pts)

Task A: Arrival time = 0 seconds, Duration = 5 seconds
Task B: Arrival time = 0 seconds, Duration = 10 seconds
Task C: Arrival time = 10 seconds, Duration = 10 seconds

Average turnaround time:

/ 3

Average response time:

/ 3

(B) RR Scheduler with a time slice of 1 second. (5pts)

Task A: Arrival time = 0 seconds, Duration = 5 seconds
Task B: Arrival time = 0 seconds, Duration = 5 seconds
Task C: Arrival time = 5 seconds, Duration = 5 seconds

Average turnaround time:

/ 3

Average response time:

/ 3

(c) In general how to FIFO and RR compare in terms of average response time and average turn around time? (5pts)

(d) MLFQ Scheduler. (5pts)

Two queues, one with higher priority and a time slice of 2 and one with lower priority and a time slice of 4. (5pts).

Task A: Arrival time = 0 seconds, 1 second of CPU, followed by 1 second of I/O. Total CPU Duration = 4 seconds
Task B: Arrival time = 0 seconds, All CPU, Duration = 4 seconds

Average turnaround time:

/ 2

How much total time does A have to wait while it is ready to run:

5. Locks and Semaphores (20 pts total)

(a) In class we described a lock implemented using test and set: (10 pts)

```
void lock(){
  while (TestAndSet(&lock, 1));
}
```

What if we add an extra while loop to our code as shown below. Will our lock still operate correctly? Why or why not?

```
void lock(){
  while (lock == 1)
    sleep(1); // sleeps for one second
  while (TestAndSet(&lock, 1));
}
```

(b) There is another implementation of locks called “Peterson’s Algorithm”. We have given you most of the solution below. Complete the following code to make this work. There are only two threads. (10 pts)

```
int interested[2] = [FALSE, FALSE];
Int turn = 0;

void lock (int thread_num) {
        interested[thread_num] = TRUE;
        int j = 1 - thread_num;
        turn = j;
        while (                &&                );
}

void release (int thread_num) {
        interested[thread_num] = FALSE;
}
```

Thread 0
lock(0);
<Critical section>
release(0);
	Thread 1
	lock(1);
	<Critical section>
	release(1);

6. Things you should know (10 pts total)

(a) An MLFQ is using what to predict the future? (5 pts)

(b) What does this output? (5 pts)

```
#include <stdio.h>
int main()
{
    float arr[5] = {12.5, 10.0, 13.5, 90.5, 0.5};
    float *ptr1 = &arr[0];
    float *ptr2 = ptr1 + 3;
    printf("%f ", *ptr2);
    return 0;
}
```