Drew Smith
COP4610
March 30th, 2016
Assignment 2

COP4610 / CGS5765 Operating Systems Homework 2 CPU Scheduling and Process Synchronization (50 pts)

Process Synchronization (25 pts)

1. (Exercise 5.8) The first known correct software solution to...(5pts)

a. Mutual exclusion is satisfied through the use of the flag and turn variables. If both processes set their flag to true, only one will succeed, which would be the process whose turn it is. The waiting process can only enter its critical section when the other process updates the value of turn.

b. Progress is also satisfied through the use of the flag and turn variables. This algorithm does not provide strict alternation. If a process accesses their critical section, it can set their flag variable to true and enter their critical section. The algorithm sets turn to the value of the other process when exiting its critical section. If this process enters its critical section before the other process, it will repeat the process of entering its critical section and setting turn to the other process.

c. Bounded waiting is preserved through the use of the turn variable. If two processes enter their own critical section, they both must set the value of their flag to true. Only the thread whose turn it is can go on while the other thread waits. If bounded waiting were not preserved, it would therefore be possible that the waiting process would have to wait indefinitely while the first process repeatedly enters and exits its own critical section.

2. (Exercise 5.15) Consider how to implement a mutex lock... (5pts)

```
do{
        flag[i] = true;
        turn = j;
while (flag[j] && turn == j);
        critical section
flag[i] = false;
        remainder section
}while (true);
```

3. (Exercise 5.20) Consider the code example for allocating...(5pts)

Drew Smith
COP4610
March 30th, 2016
Assignment 2

a. There is a race condition on the variable number of processes.

b. A call to acquire() must be placed upon entering each function and a call to release() immediately before exiting each function.

c. No, because the race occurs in the allocate process() function where the number of processes is first tested in the if statement. Then it is updated based on the results of the conditional statement. It is possible that number of processes equals 254 at the time of the test. But because of the race condition, it is set to 255 by another thread before it is incremented yet again.

4. (Exercise 5.23) Show how to implement the wait() and signal ()...(10pts)

```
int guard = 0;

int semaphore value = 0;

wait()

{

        while (TestAndSet(&guard) == 1);

                if (semaphore value == 0) {

                        //atomically add process to a queue of processes

                        //waiting for the semaphore and set guard to 0;

                } else {

                        semaphore value--;

                        guard = 0;

                }

}

signal()

{

        while (TestAndSet(&guard) == 1);

                if (semaphore value == 0 && there is a process on the wait queue)
```

wake up the first process in the queue of waiting processes

else

semaphore value++;

guard = 0;

}

CPU Scheduling (25 pts)

1. (Exercise 6.9) The traditional UNIX scheduler.... (5pts)

The priorities assigned to the processes are 80, 69, and 65 respectively. The scheduler lowers the relative priority of CPU-bound processes.

2. (Exercise 6.16) Consider the following set of processes, with the length of the CPU burst given in milliseconds:....

a) Gantt charts:

112 3 4 I 5 FCFS

21 1 4 I 5 I 3 SJF

3 5 1 4 I 2 Priority

1 2 3 4 5 3 I 4 7 I 3 5 3 RR

1 2 3 41 5 61 7 81 9J 1O 11 121 13J 141 151 I(17 18 191 20

b) Turnaround time:

FCFS SJF Priority RR

P1 2 3 15 2

P2 3 1 20 3

P3 11 20 8 20

P4 15 7 19 13

P5 20 12 13 18

e) Waiting time:

FCFS SJF Priority RR

Pl 0 1 13 0

P2 2 0 19 2

P3 3 12 0 12

P4 11 3 15 9

P5 15 7 8 15

d) Shortest job first

3. (Exercise 6.19) Which of the following scheduling algorithms could result in starvation... (5 pts)

Shortest job first and priority based scheduling algorithms could result in starvation.