CISC324: Operating Systems

Assignment 3

due Monday, Feb 11 at the 8:30 lecture

January 29, 2019

Student Full Name:

Student Number:

Group Number:

Copyright © Karim Lounis, 2019.
1. What is a critical section in a given program.

2. Explain how old uniprocessor operating systems (e.g., earlier version of UNIX) handled the critical section problem.

3. Explain the two styles of using semaphores.

4. Can a process be interrupted (context switched) during the execution of its critical section? Explain your answer.

5. Why disabling interrupts system during critical section execution is not a good idea in multiprocessor systems?

6. What are the three requirements for the critical section problem.

Copyright © Karim Lounis, 2019.
7. What are the two main advantages of implementing the primitives acquire() and release() using queues instead of using a busy waiting statement in acquire() and an incrementation statement in release?

_____________________________________________________________________

_____________________________________________________________________

8. Consider the Peterson's algorithm for the critical section problem with two processes.

Process 0:   

Begin
while(true)
{
    (1) Flag[0]=true;
    (2) turn=1;
    (3) while(flag[1] && turn==1);
    (4) ... Critical Section ...;
    (5) flag[0] = false;
}
End

Process 1:   

Begin
while(true)
{
    (1) Flag[1]=true;
    (2) turn=0;
    (3) while(flag[0] && turn==0);
    (4) ... Critical Section ...;
    (5) flag[1] = false;
}
End

a) If the two statements (1) and (2) are swapped in both processes, state whether the altered code still satisfies the three requirements of the critical section problem. Explain which requirement has been violated.

_____________________________________________________________________

_____________________________________________________________________

_____________________________________________________________________

_____________________________________________________________________

b) If we change the logical operator in the condition of the while statement (Line 3) from and to or (i.e., while(flag[1-i] && turn==1-i) becomes while(flag[1-i] || turn==1-i)), state whether the altered code still satisfies the three requirements of the critical section problem. Explain which requirement has been violated.

Copyright © Karim Lounis, 2019.
9. Hardware solutions to the critical section problem use read-modify-write instructions: these instructions are indivisible because the CPU gets to hold onto the memory bus for two cycles, to both read and then update the shared memory location. As an example, here is code for implementing mutual exclusion using a Swap() machine-code instruction; that instruction has the opcode EXCH in the Pentium instruction set. Each process $P_i$ executes the same code to access its critical section, where key is a boolean local variable and lock is a shared memory location among processes $P_i$ that contains a boolean value, initially set to false.

**Process $P_i$:**

```cpp
do {
    key = true;
    while (key == true) Swap(&lock, &key);
    ... Critical section of $P_i$...
    lock = false
} while(true);
```

Describe what happens if three processes try to enter their critical section at about the same time. How does the given code ensure that only one process gets entry?
10. Write CoBegin/CoEnd code for the following precedence graph. Make your program express as much parallelism as possible within the limitations of CoBegin/CoEnd, while being sure to enforce all the constraints that are in the precedence graph. (The best solutions introduce two or three extra precedence edges. Try to find one of them.)