Title

Stuart Mashaal

Thursday, November 09, 2017

Question 1

The Question

Using the code below, give a sequence of steps that can lead to deadlock.

Process 0

```
1    flag[0] = True
2    while (flag[1]): pass
3    critical_section()
4    flag[0] = False
```

Process 1

```
1    flag[1] = True
2    while (flag[0]): pass
3    critical_section()
4    flag[1] = False
```

The Answer

Steps:

1. Process 0 runs line 1, setting flag[0] to True
2. Process 1 runs line 1, setting flag[1] to True
3. Process 0 runs line 2, enters spin-lock
4. Process 1 runs line 2, enters spin-lock

Both processes will now loop forever in deadlock if this sequence of steps is followed when the code is run, since they both enter spin locks at the same time.

Question 2

The Question

Using the code below, give a sequence of steps, showing the value of the relevant variables at each step, that will cause the code to fail to provide mutual exclusion.

Process 0

```
1    while (turn != 0): pass
2    critical_section()
3    turn = 1
4    non_critical_section()
```

Process 1

```
1    while (turn != 1): pass
2    critical_section()
3    turn = 0
4    non_critical_section()
```

The Answer

Other than the trivial solution where the initial value of the shared variable 'turn' is not $\in \{0, 1\}$ (in which case neither process ever gets past their spinlock), this code *actually does* provide mutual exclusion. However, this code does not satisfy one of the other conditions of correctness of a solution to the *critical section problem*, namely that **a thread blocked from entering a critical section can only be blocked by a thread in the critical section**.

Steps:

1. Set turn to 0
2. Process 0 runs lines 1 through 4, entering the non_criical_section(). This non_critical_section() halts forever
3. Process 1 runs lines 1 through 4, completing its non_critical_section()

Now Process 0 and 1 have both had their turn at running the critical_section(). However, now that Process 0 is stuck forever inside of its non_critical_section(), Process 1 will never be able to re-enter its critical_section() since 'turn' will never get set to 1 again by Process 0. This means that Process 1 is blocked from entering its critical_section() **despite the fact that Process 0 is not in the critical_section() itself**.