Kyle Pamintuan

12/4/17

CECS 326 – Section 1

Prof. Ngo

Semaphore Homework

**Problem #1**

Initial values 🡪 sem1, sem2, sem3 = 0

|  |  |  |
| --- | --- | --- |
| Process 1 | Process 2 | Process 3 |
| loop  F1()  V(sem1)  V(sem1)  P(sem2)  P(sem3)  G1()  H1()  V(sem1)  V(sem1)  P(sem2)  P(sem3)  M1()  N1()  V(sem1)  P(sem3)  endloop | loop  F2()  V(sem2)  V(sem2)  P(sem1)  P(sem3)  G2()  V(sem2)  P(sem3)  H2()  V(sem2)  V(sem2)  P(sem1)  P(sem3)  M2()  N2()  endloop | loop  F3()  V(sem3)  V(sem3)  P(sem1)  P(sem2)  G3()  V(sem3)  P(sem2)  H3()  V(sem3)  V(sem3)  P(sem1)  P(sem2)  M3()  N3()  V(sem3)  P(sem1)  endloop |

First, I’ve chosen all semaphore variables to have the initial value of zero because the directions require us to block processes to wait for other processes to finish.

Since all three processes must halt until functions F1, F2, and F3 have completed execution, I’ve placed two V semaphores followed by two P semaphores after each of these functions to satisfy this requirement.

The two V semaphores after each function is to resume that process in case it was placed in the blocked queue by the other two processes. The two P semaphores that follow are used to halt that process and wait for the other processes to complete the requirement above.

Since process 2 and 3 must halt until functions G2 and G3 have completed execution. For this requirement I’ve placed one V semaphore and one P semaphore after each function. Again, the V semaphore is to resume that process in case that function was placed in the blocked queue and the P semaphore is to block that process until the other process is finished satisfying the requirement.

By now, you can see the repeating pattern of semaphores that I used for each requirement. I used that same pattern for the remaining requirements of the problem based on which processes to block and how many processes to block. Semaphores are not placed after certain functions if they don’t require to wait for other processes to proceed to the next line of code.

By placing the V semaphores BEFORE the P semaphores and keeping 1:1 ratio for P and V semaphores, I have prevented any deadlocks to occur. And since my pattern of semaphores causes processes to wait on other processes, it creates temporary starvation for the waiting processes.

**Problem #2**

Initial values 🡪 a0, a1, a2, b0, b1, b2 = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Robot A | Robot B | Robot C | Robot D | Robot E |
| loop  compute briefly  P(b0)  P(a0)  P(a1)  Occupy A0  Return to base  V(a1)  V(a0)  V(b0)  Compute a little  P(b0)  P(a0)  Occupy B0  Return to base  V(a0)  V(b0)  endloop | loop  calculate briefly  P(b0)  P(a0)  Occupy B0  Return to base  V(a0)  V(b0)  calculate a little  P(b2)  P(a2)  P(b1)  Occupy B1  Return to base  V(b1)  V(a2)  V(b2)  endloop | loop  await shipment  P(a0)  P(a1)  Occupy A1  Return to base  V(a1)  V(a0)  await shipment  P(b2)  P(a2)  P(b1)  Occupy B1  Return to base  V(b1)  V(a2)  V(b2)  endloop | loop  P(b0)  P(a0)  P(a1)  Occupy A0  Return to base  V(a1)  V(a0)  V(b0)  Intermission  P(b1)  P(a2)  P(b2)  Occupy B2  Return to base  V(b2)  V(a2)  V(b1)  DL new instruct.  endloop | loop  check inventory  P(a0)  P(a1)  Occupy A1  Return to base  V(a1)  V(a0)  Quick calculation  P(b2)  P(b1)  P(a2)  Occupy A2  Return to base  V(a2)  V(b1)  V(b2)  endloop |

First, the most crucial requirement is that no work area can be occupied by more than one robot. For this reason, I’ve chosen to initialize all semaphore variables to 1 and placed P semaphores before the critical sections highlighted above. This way, a robot will decrement the variable to 0 and in case that robot loses its turn in the critical section, the other robots be blocked off from that work area.

Now, since A0 and A1 cannot be occupied once A0 is occupied by a certain robot, I’ve placed P semaphores before any critical sections containing the occupation of A0. In this problem, that includes Robot A and D. But if we look at another requirement that states B0 and A0 cannot be occupied once B0 is occupied by a certain robot, we can now see that Robot A and D must also block off B0 before occupying A0. That makes a total of three P semaphore before the occupation of A0.

Before the occupation of B0 though, we only need to block off B0 and A0.

Before the occupation of A1, we must block off A0 and A1.

And lastly, before the occupation of B1, B2, or A2 – we must block off all B1, B2, and A2 as per the final requirement.

By initializing all semaphore variables to 1 and keeping a 1:1 ratio between all P and V semaphores, I’ve prevented any deadlocks from occurring. And since robots are required to wait for a certain work area when it’s occupied, this causes a temporary starvation for those waiting robots.