Haiqiang Zou

CS340

10/13/2013

**Homework Assignment 3**

**A. Consider the 2nd attempt (from the lecture notes). Is the “No Starvation” condition satisfied?**

**Hint: in your proof you might want to check if there is a particular execution sequence by which a process might be able to use the CS over and over, while the other process is starving in the while loop.**

Second Attempt

Each process should have its own key to the CS so that if one process is executing outside the CS the other one is able to get in to its CS.

Replace the turn variable with a shared global variable initialized to false.

Flag is initialized to 'false'

P0 P1

while(true) { while(true){

while (flag[1]) do no-op; while (flag[0]) do no-op;

flag[0] = true; flag[1] = true;

CS CS

flag[0] = false; flag[1] = false;

remainder section; remainder section;

} }

No starvation condition is not satisfy in this attempt because we can have a case that when P1 is inside the busy wait, and P0 executes the process, however the process of P0 executes fast enough for itself to repeat it over and over once the CS is finish before P1 can get into CS. This situation will cause the P1 to starve; therefore no starvation condition is not satisfy.

**B. Prove that the Peterson Solution is correct by showing that all 3 conditions for a correct solution to the Critical Section Problem are respected.**

**Hint: you can use the textbook comments but your proof should be clearer and more detailed.**

Peterson's solution :

turn = 0;

flag[0] = false;

flag[1] = false;

P0 P1

while(true){ while(true){

flag[0] = true; flag[1] = true;

turn =1; turn = 0;

while (flag[1] and turn ==1) do no-op; while (flag[0] and turn ==0) do no-op;

CS CS

flag[0] = false; flag[1] = false;

remainder section; } remainder section;}

Mutual Exclusion

Assume P0 is in CS, flag[1]=false, turn=0, or both flag[1]=false and turn = 0

In order for flag[1]=false, P1 has to been in the state of finishing CS or P1 has not been executed.

In order for turn=0 and since P0 is in CS we know that flag[0]=true therefore P1 will be in busy wait.

In order for both flag[1]=false and turn = 0, P2 must be in the remainder section.

Therefore, Mutual Exclusion is satisfied.

No Starvation

While P0 is in busy wait and P1 is in CS, in order for P0 to be in busy wait flag[1]=true and turn=1, Once P1 finishes CS the flag[1]=false. P1 will be able to access CS. Even if P1 executes again, it will be stuck in busy wait because flag[0]=true whenever P0 is in busy wait and turn=0. Therefore, no starvation is satisfy.

No Delay

Assume P0 is in remainder section, P1 will be able to execute CS over and over because flag[0]=false so it will not go into busy wait. Therefore, no delay is satisfied.

No Deadlock

Deadlock will not happen because the global variable turn cannot be 0 and 1 at the same time. Two process will not be in busy wait at the same time. Therefore, no deadlock is satisfied.