Due: Tuesday, October 3, 2017

1. Provide relevant parts of C-like codes for processes X, Y, Z, W and V in the following synchronization problem. Each process has a special (synchronization) point in its code, and each process may cross its synchronization point only if all other processes have reached or crossed their synchronization points.

2. Provide relevant parts of C-like codes for processes X, Y, Z, W, V and S in the following synchronization problem. Each process has a special (synchronization) point in its code, and each process may cross its synchronization point only if at least three other processes reached or crossed their synchronization points. Assume that each process will cross its synchronization point only once.

3. Provide relevant parts of C-like codes for processes X1, X2, X3, X4 and X5 in the following synchronization problem. Each process has a special (synchronization) point in its code, and:
   - Process X1 may cross its synchronization point unconditionally
   - Process X4 has to wait for processes X3 and X1 to cross or reach their synchronization points, and only then X4 may cross its synchronization point,
   - Process X2 has to wait for process X1 to cross or reach its synchronization point, and only then X2 may cross its synchronization point,
   - Process X3 has to wait for process X4 to cross or reach its synchronization point, and only then X3 may cross its synchronization point,
   - Process X5 has to wait for processes X1, X3 and X4 to cross their synchronization points, and only then X5 may cross its synchronization point.

   Assume that each process will cross its synchronization point only once.

4. This problem is an extension of ‘Consumer-Producer Problem’. Provide C-like codes for processes in the problem ‘One Consumer and all three Producers’. In this problem, there are three producers and each producer has its own buffer (with A, B and C slots, respectively). Producers behave in the usual way. The only consumer takes total of 3 items in each iteration, one item from each buffer.

5. You are given C codes for Producer and Consumer. Codes use two hypothetical system calls: sleep() and wakeup(X).
   - sleep() - The issuing process X is blocked and it will stay blocked until waken up by some process that issues wakeup(X) system call.
   - wakeup(X) - This system call does not effect in any way the issuing process. If the process X has been blocked because it previously issued sleep(), the process X is unblocked and put in a ready state, otherwise this system call does not have any effect.
#define N = 100
int count = 0;
typedef struct
{
    ...
} yy;
yy buffer [N];

Producer code:

{  
    yy Prod;
    int i=0;
    while (true)
    {
        ...  /*produce item in Prod*/
        ...
        if (count == N) sleep();
        buffer[i]= Prod;
        i = (i+1)% N;
        count = count + 1;
        if (count == 1) wakeup (Consumer);
    }
}

Consumer code:

{  
    yy Cons;
    int i=0;
    while (true)
    {
        ...    /* consume item from Cons*/
        ...
        if (count == 0) sleep();
        Cons = buffer[i];
        i = (i+1)% N;
        count = count - 1;
        if (count == N-1)wakeup(Producer)
        ...
    }
}

The codes have an unresolved critical section problem related with the following two statements:

count = count + 1  /* in producer code
count = count - 1  /* in consumer code

We discussed that problem in the class and it is not of concern for this problem, i.e. assume that the variable count is updated in those two statements in such way that its value is always consistent.

But, there is still a problem with the given codes. It is possible that both processes may execute sleep() system call, thus without any of them being able to execute a wakeup system call.

Provide one such scenario illustrating that something like that can happen. Your scenario should be along the following lines: one process runs first and at certain point it is interrupted, and then the second process runs and after some time is interrupted and so on, until both processes issue sleep() system call. In your scenario, Producer must first issue sleep().

Note: You are not asked to fix this code.