CIS*3110: Operating Systems

Assignment 4: Synchronizing Threads with POSIX Semaphores

Q1) Could you predict the output of the program before execution? (Yes/No)

No, I cannot accurately predict the exact output of the program before execution. However, I can predict that it will not be the desired value of 2,000,000 because the global variable is not blocked from the other thread while it's being accessed. Of all the times I've run *ibadcnt*, it is always lower than the desired 2,000,000 due to the aforementioned issue.

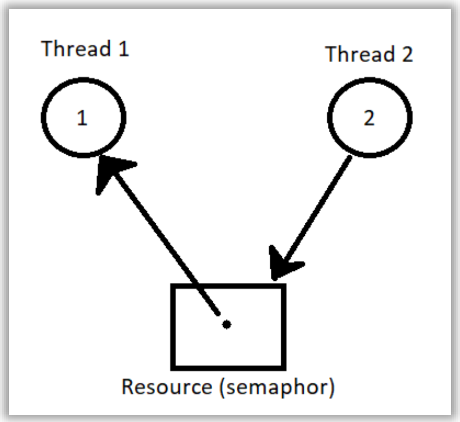
Q2) Please record your execution output of the executable *ibadcnt*. You need to run the executable *ibadcnt* three times and record all the outputs. You can save your execution output into a file named *a4_badoutput.txt* by entering: `./ibadcnt >> a4_badoutput.txt`

Please see *a4_badoutput.txt* for Q2 results.

Q3) In your modified program, do you think it is possible to become deadlocked? In other words, your program is in a situation in which two threads are waiting for each other to increment the counter, and thus neither ever does. Give Resource-Allocation Graph to help in your explanation. Note that in your Resource allocation graph, the thread is represented by a Circle node while the Semaphore or the Resource is represented by a rectangle node.

In my modified program, I do not think it is possible to become deadlocked because a deadlock in this situation occurs when the two threads are waiting for each other to increment the counter, but this can never possibly happen. In other words, for a possibility of deadlock, there must be a cycle in the Resource Allocation Graph; however, as you can see in the following image, there are no cycles, therefore, it is not possible to be deadlocked in my modified program, *igoodcnt*.

The first thread will iterate the global count NITER times, then release its hold of the resource so the second thread can increment the global count another NITE times. This is accomplished using the POSIX semaphore.



Due: Monday, April 1, 2019 at 23:59:59