CSE 330 - Operating Systems

Project 5, **Due: December 4th 2020, 23:59 PM**

Strictly Individual Projects, No late submission is possible on this project

**1.Dining Philosophers Problem:**

Using the semaphores you have implemented, detect deadlocks in Dining Philosophers problem.

**2.Test case:**

The first line of the test case contains two numbers

A,B

A is the number of philosophers, and B is the number of entries coming after the first line

The number of chopsticks is the same as number of philosophers

Your program should have an array of semaphores where each semaphore (each element of the array) corresponds to each chopstick.

Each philosopher gets access to “the right” chopstick by doing

P(Sem[(philosopher ID – 1)% (number of Chopsticks)])

Then yields to the next philosopher

When the philosopher comes back the philosopher attempts to access “the left” chopstick by doing

P(Sem[philosopher ID % (number of Chopsticks) ])

Then yields to the next philosopher

When it comes back it prints

Printf(“\n Philosopher %d is eating \n”, philosopher ID);

Then it releases the right chopstick by doing

V(Sem[(philosopher ID – 1)% (number of Chopsticks)])

Then it releases the left chopstick by doing

V(Sem[(philosopher ID)% (number of Chopsticks)])

Then it deletes itself from the readyQ.

Then it yields to the next philosopher

The next B lines has philosopher numbers in order to populate the Ready Queue.

The program starts with the first element of the Ready Queue.

Test case example

4,2

1

3

Output

Philosopher 1 is eating

Philosopher 3 is eating

Test case example 2

4,2

1

2

Output

Philosopher 2 is eating

Philosopher 1 is eating

Your project must consist of 5 files

1. TCB.h (uses ucontext.h)
2. q.h (includes TCB.h)
3. threads.h (includes q.h)
4. sem.h (includes threads.h) if you have written one.
5. proj-5.c (includes threads.h)

(make sure the compile command, “gcc proj-5.c” does the correct compilation).

All 5 files are to be ZIPPED into one zip file and uploaded in Canvas. The name of this file should reflect the name of the student (abbreviated, do not make it too long).

**Note: Grading is on Ubuntu. It is in your interest to make sure the program compiles and runs in the target test platform.**

**Testing your code on grading test cases**

Grading test cases are different from the test cases provided.

To test your code against grading testcases follow these steps:

Put the four files q.h tcb.h threads.h and proj-5.c in one zip file and name the file YOUR LAST NAMEProject5.zip (no gaps)

Make sure you do not have a folder inside your zip. When we unzip we should only have the files not a folder.

For example BanerjeeProject5.zip

The submission site is: <http://10.218.107.121/index330P5.html>

In order to access the submission site you need to first login to sslvpn.asu.edu using you ASU ID and password. You will be asked to download cisco vpn. Please do so. For any subsequent time you want to access the submission site you have to login to sslvpn.asu.edu.

In the site there will be an option to upload one file. Please upload YOURLASTNAMEProject5.zip file there. Then please hit upload. Wait for 5 seconds and then you can see the evaluation of your code on the grading test cases. To access the evaluation please type in

<http://10.218.107.121/YOURLASTNAMEProject5.output>

such as

[http://10.218.107.121/ BanerjeeProject5.output](http://10.218.107.121/%20BanerjeeProject5.output)

Then download the text file and open with any text editor. If your code passed one test case then the corresponding test case will show “NO ERRORS HERE”

If it failed a given test case then it will print “------------------------------”

At the end of the file the total number of test cases you passed will be displayed.