LE/EECS 3221 – Operating System Fundamentals

Winter 2019

Programming Assignment 3

Submission Deadline: March 31, 2019 before 23:59

Objectives

- Learn the concept of process synchronization
- Learn how to use Linux system calls related to processes synchronization

Submission Requirements

Please submit your results/output in the solution template provided.

The questions that involve performing task on the computer, your answer will be the screen shot of your steps; writing the answer in the text form is not acceptable.

The questions that involve coding/script, you have to write your code/script in text form in the document. Also, separately upload the source code (.c etc.) files.

The submission deadline is March 31, 2019 23:59. There are no extra submission days for this assignment.

Assignment Requirements and Setup

You are required to perform the tasks in a Linux environment using C/C++ language. Before starting the tasks, first change your command prompt as discussed earlier. As a result, make sure that you see your Student ID in command prompt every time you write a new Linux command.

Implementation of "The Dining Philosopher's Problem" using POSIX Semaphores

The Dining Philosopher's Problem is discussed and described in detail in Chapter 7. An algorithm to solve this problem is also provided (You are not strictly required to follow this algorithm). However, this solution is based on Monitors. In this assignment, we want to implement the same solution but using POSIX Semaphores. The examples and usage of POSIX Semaphores is discussed in Chapter 7. Further, details can be found through the man pages. Also, read the programming projects section at the end of the chapter 7. You are required to implement Deadlock free solution as in section 7.1.3; Starvation free is not required.

An additional task that you need to perform in your code is to run the simulation of this algorithm. In order to do that, create a struct/class to represent philosophers. Each philosopher will have Name/Id and State (refer to the section 7.1.3). Initial state for each

philosopher is "Thinking". Also in the philosopher struct/class, add two intervals: eat_interval (represents how many time units the philosopher will eat) and think_interval (represents after how many time units the state should be changed from "Thinking" to "Hungry"). Select the values for these intervals randomly for each philosopher at the time of instance creation.

Once you have created and initialized the 5 instances of the philosophers, the rest of your logic/code will be in a loop; set the number of iterations to 500 (simulation length). In each, iteration, perform the tasks (checking the status, updating the status, take necessary action when the state is "Hungry") in an appropriate order.

In order to make your simulation understandable, add appropriate logging messages whenever an action is performed.

- **#1-A:** Write/type your source code in your submission file/document. Also, submit the .c file separately.
- **#1-B**: In your submission file/document, add a screenshot which displays the execution of your program. In your program, add appropriate logging messages to show the progress of your program.

Bonus Question: Worth 5 Marks (= weight of one assignment)

Implement a Deadlock free as well as Starvation free solution for the Dining Philosophers Problem. You are permitted to extend your code beyond the basic logic/requirement described in the Question 1.

- **#Bonus-A:** Write/type your source code in your submission file/document. Also, submit the .c file separately.
- **#Bonus-B:** In your submission file/document, add a screenshot which displays the execution of your program. In your program, add appropriate logging messages to show the progress of your program.