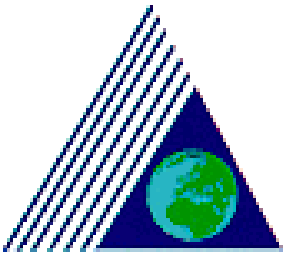

YEDITEPE UNIVERSITY

CSE331 OPERATING SYSTEMS DESIGN
SPRING 2024

ASSIGNMENT II

Last Submission Date:
19 May 2024, 23:59

SYNCHRONIZATION WITH SEMAPHORES

BRIDGE IN REPAIR PROBLEM

The local Bridge is undergoing repairs and only one lane is open for traffic. To prevent accidents, traffic lights have been installed at either end of the bridge to synchronize the traffic going in different directions. ***A car can only cross the bridge if there are no cars going the opposite direction on the bridge and there is empty space on the bridge.*** Sensors at either end of the bridge detect when cars arrive and depart from the bridge, and these sensors control the traffic lights.

You are required to write a multi-threaded and a multi-process program by using semaphores for the “bridge in repair” problem.

MULTI THREADED PROGRAM: (30 pts)

- Your program will take bridge capacity and the total number of cars as command line arguments.
- Then, it starts to create threads for each car by determining the direction of the car randomly (***GO_LEFT*** or ***GO_RIGHT).***
- When a thread starts to operate, it calls the ***Arrive(direction)*** function. This function is used for the car that arrives at the bridge and the **direction** parameter shows the direction of the car.
- After a car leaves the bridge, it calls the ***Depart(direction)*** function. For the right operation of the threads which is defined in the introduction part, a right synchronization method should be used and these functions are responsible for the synchronization.
- These procedures should arrange arriving cars to the bridge safely. Remember that, all the synchronization operations must be mutually exclusive. You should use POSIX semaphores to enable synchronization. Change the number of the cars 20 to 300000 and the capacity of the car 2 to 20 .

MULTI PROCESS PROGRAM: (30 pts)

The multi process program that you are required to code will firstly ask the total number of cars and the car capacity of the bridge from the user. Then, it starts to create processes for the cars and these processes will randomly determine the direction of the car as ***GO_LEFT*** or ***GO_RIGHT.*** When a process starts to operate, it calls the ***Arrive(direction)*** function. This function is used for the car to arrive at the bridge and the ***direction*** parameter will be set to the direction where the car goes. After a car leaves the bridge, it calls the ***Depart(direction)*** function. For the right operation of the processes which is defined in the introduction part, a right synchronization method should be used and these functions are responsible for the synchronization. These procedures should arrange arriving cars to the bridge safely. Remember that, all the synchronization operations must be mutually exclusive. You should use shared semaphores to enable synchronization. Change the number of the cars 20 to 300000 and the capacity of the car 2 to 20 .

Each program should display the arrangement of the car on the bridge at each time. After each run of the program, total execution time should be displayed on the screen.

REPORT: (40 points)

You are required to write a report that expresses the design, implementation and execution flow of your programs including your understanding from processes, threads and synchronization primitives. By comparing **execution times** from each program you have written, make comparisons between multi-process programming with shared semaphores and multi-threaded programming with POSIX semaphores. In addition, compare the performance of the programming solutions using **another choice of your own performance metric** which you consider as crucial for the expression of the difference. Explain your results clearly and support them with graphics in your report.

SUBMISSION RULES:

- Writing clean, readable code and using comments are recommended.

- Including the student name and ID in the top of the code, as comments are required.

- Filenames of the C sources must be **hw2a.c** and **hw2b.c**.

- For the report, only PDF format is accepted. The filename of the report must be **Report2.pdf**.

- This homework will be sent only to **YULEARN**.

- Title of your submission must include the following information:
    - Course Code
    - Student number
    - The attached file(s).

- Zip all files related to the assignment.

- The name of the zip file should consist of **student ID**, **assignment number**, excluded from space and special characters.

- **Important Note for Work: If you use semaphores in your work, you will not use a mix of semaphores and monitors (i.e, condition variable and mutual exclusion [mutex]) and you should implement synchronization primitives with only semaphores. Note that it is not safe to use mutex with semaphores. In the case that you use a mix of these concepts and even do a complete work, your work will not be graded.**

- Important Note for Grading: Compile/Run is a critical process while developing your solution. At the end of the Assignment, you should assign a compilable/runnable code (i.e., code without compile time errors and run time errors such as segmentation fault, deadlocks) for a total grade. Otherwise, a major grade reduction will be done for the assignment. Your assigned code will not be changed after assignment and the assigned work in YULEARN will be taken into account for grading.

You are welcomed to ask questions, come up with new ideas about the homework, but reading the instructions explained here carefully, also studying from the course book are highly recommended to have a general understanding before asking questions. Playing with the program implementation is encouraged. More creative ideas can get higher points.