HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY, VNU HCM
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

OPERATING SYSTEM

Lab 6: Synchronization

Pham Minh Tuan - MSSV: 1752595

1 Exercises

Problem 1: Race conditions are possible in many computer systems. Consider a banking system that maintains an account balance with two functions: `deposit` (amount) and `withdraw` (amount). These two functions are passed the amount that is to be deposited or withdrawn from the bank account balance. Assume that a husband and wife share a bank account. Concurrently, the husband calls the `withdraw()` function and the wife calls `deposit()`. Write a short essay listing possible outcomes we could get and pointing out in which situations those outcomes are produced. Also, propose methods that the bank could apply to avoid unexpected results.

Answer:

There are many possible outcomes that we could get when the husband and the wife sharing a bank account concurrently perform a transaction. Assume that when a husband perform a `deposit()` function while the wife call the `withdraw()` function at the same time, the system will send a copy of the cash in bank account to the husband and wife. However the cash given is the same for both users but each user uses the different operations so the cash will be changed in a different way. So the cash stored in the bank account will be updated concurrently meaning that if the `withdraw()` finish first, the cash will be update once, and then the `deposit()`, so the result will be according to the last operation performed. In case that the `withdraw` and `deposit` is concurrently called, if the `withdraw` finishes last then the `deposit` will not be updated and the loss of cash in the bank deposit would happen or vice versa.

To prevent the loss in cash would happen, we let *only one* operation is entered to change the value of cash in the bank account at specific time, after that it will release to let another operation comes into. If both the `deposit` and `withdraw` is performed concurrently, the system will choose the one will enter first depend on the priority and then the other will enter. The section that for changing the value of cash can be called the **critical section**.

Problem 2: In the Exercise 1 of Lab 5, we wrote a simple multi-threaded program for calculating the value of pi using Monte-Carlo method. In this exercise, we also calculate pi using the same method but with a different implementation. We create a shared (global) count variable and let worker threads update on this variable in each of their iteration instead of on their own local count variable. To make sure the result is correct, remember to avoid race conditions when we update the shared global variable by using mutex locks. Compare the performance of this approach with the previous one in Lab 5.

Answer: The way I implement the Exercise 1 in Lab5 is actually the same approach as you discuss here in the Problem 2 of Lab6 so there is no change in code in submit for Problem 2.