ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

Exercise
Operating System Lab6

GV: Vũ Văn Thống
SV: Võ Minh Long 1812951

Ho Chi Minh City, 05/2020

Mục lục

1 Problem1

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 These two functions are passed the amount that is to be deposited or withdrawn from the bank account balance.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.

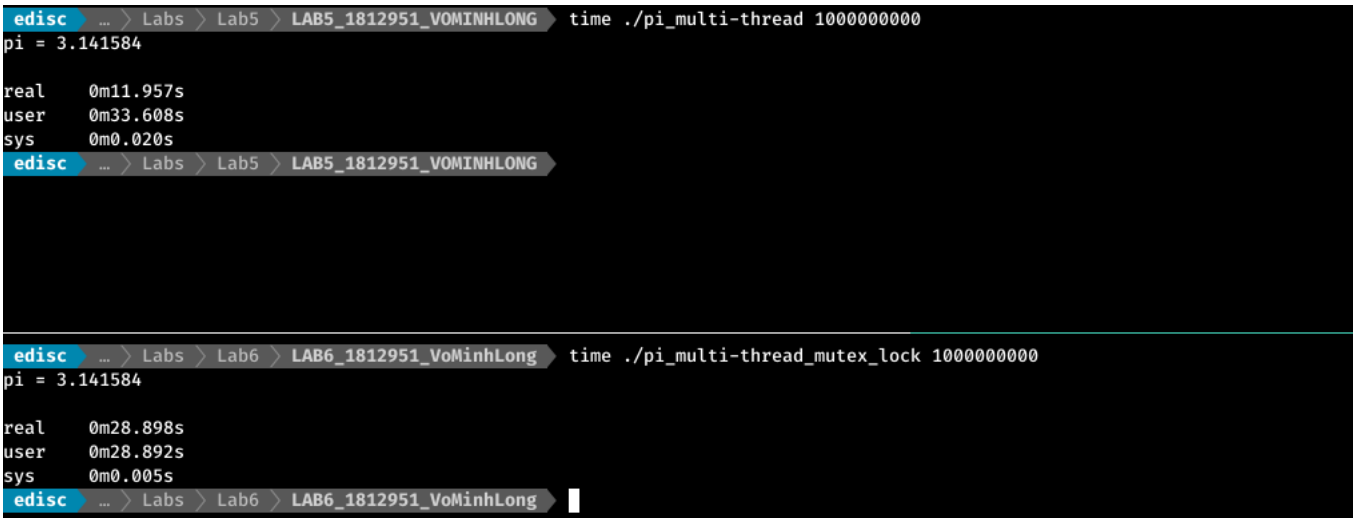
Các tình huống có thể xảy ra:

- Bình thường
    1. Người chồng gọi hàm withdraw() và thực hiện xong, người vợ gọi hàm deposit() và thực hiện.
    2. Người vợ gọi hàm withdraw() và thực hiện xong, người chồng gọi hàm deposit() và thực hiện.
- Race condition
    1. Người chồng gọi hàm withdraw() và hàm này đang thực hiện thì người vợ gọi hàm deposit()
    2. Người vợ gọi hàm withdraw() và hàm này đang thực hiện thì người chồng gọi hàm deposit()
    3. Cả vợ chồng đều gửi yêu cầu thực hiện hàm withdraw() / deposit() cùng một thời điểm.
    4. Chồng vừa gửi yêu cầu thực hiện hàm withdraw() lên hệ thống và hàm chưa chạy thì vợ lại gửi tiếp yêu cầu deposit() lên hệ thống
    5. Vợ vừa gửi yêu cầu thực hiện hàm withdraw() lên hệ thống và hàm chưa chạy thì chồng lại gửi tiếp yêu cầu deposit() lên hệ thống

2 Problem 2

In the Exercise 1 of Lab 5, we wrote a simple multi-thread 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 on updates to the shared global variable by using mutex locks. Compare the performance of this approach with the previous one in Lab 5

Sau khi chỉnh sửa chương trình và chạy kiểm tra, ta được kết quả như sau:



Hình 1: So sánh thời gian chạy của hai chương trình

Từ kết quả thời gian thực (real time) của hệ thống cho thấy, so với việc sử dụng mutex lock thì phương pháp ở Lab5 cho thời gian thực hiện nhanh hơn. Nguyên nhân là khi sử dụng mutex lock, chúng ta sẽ chỉ có một thread được phép vào critical section tại một thời điểm và những thread khác nếu có nhu cầu thì sẽ được chuyển sang trạng thái nghỉ cho đến khi thread trong critical section hoàn thành xong và đi ra.