

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	2
2	Problem 2	2

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 These two functions are passed the amount that is to be deposited or withdrawn from the bank account balance.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.
→ Số tài khoản đầu tiên sẽ giảm xuống vì bị rút ra, sau đó sẽ tăng lên vì được thêm vào.
2. Người vợ gọi hàm deposit() và thực hiện xong, người chồng gọi hàm withdraw() và thực hiện.
→ Số tài khoản đầu tiên sẽ tăng lên vì được gửi thêm, sau đó sẽ giảm xuống vì bị rút ra.

- Race condition

1. Người chồng gọi hàm withdraw() và hàm này đang thực hiện thì người vợ thực hiện hàm deposit().
→ chúng ta hy vọng rằng kết quả của số tiền sẽ là giảm xuống do người chồng rút ra, rồi sau đó tăng lên do người người vợ gửi vào. Tuy nhiên ở đây có sự tranh chấp giữa yêu cầu withdraw() và deposit() về thay đổi giá trị của số tiền nên có thể xảy ra kết quả không như ý muốn.
2. Người vợ gọi hàm deposit() và hàm này đang thực hiện thì người chồng thực hiện hàm withdraw().
→ chúng ta hy vọng rằng kết quả của số tiền sẽ là tăng lên do người vợ gửi vào, rồi sau đó giảm xuống do người chồng rút ra. Tuy nhiên ở đây có sự tranh chấp giữa yêu cầu withdraw() và deposit() về thay đổi giá trị của số tiền nên có thể xảy ra kết quả không như ý muốn.
3. Cả người vợ và chồng đều thực hiện hàm deposit()/withdraw() tại cùng 1 thời điểm.
→ Lúc bấy giờ tùy theo cơ chế scheduling của hệ thống mà có thể cho deposit() hoặc withdraw() thực hiện trước, tuy nhiên vẫn sẽ xảy tranh chấp về sự thay đổi giá tiền nên sẽ quay về một trong hai trường hợp trước đó, có thể làm cho kết quả không mong muốn

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:

```
edisc ... > Labs > Lab5 > LAB5_1812951_VOMINHLONG time ./pi_multi-thread 1000000000
pi = 3.141584

real    0m11.957s
user    0m33.608s
sys     0m0.020s
edisc ... > Labs > Lab5 > LAB5_1812951_VOMINHLONG

edisc ... > Labs > Lab6 > LAB6_1812951_VoMinhLong time ./pi_multi-thread_mutex_lock 1000000000
pi = 3.141584

real    0m28.898s
user    0m28.892s
sys     0m0.005s
edisc ... > Labs > Lab6 > LAB6_1812951_VoMinhLong
```

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.