>>> LAB TUTORIAL <<<

INTRODUCTION TO OPERATING SYSTEM / Course ID 502047

Lab Part 9 – Synchronization Examples

Mục tiêu	Lý thuyết liên quan	Tài nguyên
Bài toán ước lượng giá trị số PI		
Bài toán gửi rút tiền Ngân hàng		
Bài toán đặt Đấu giá		
Bài toán pop và push của STACK		
Bài toán Writer - Reader		
Bài toán Xe qua cầu hẹp		
Bài toán Triết gia ăn tối		

Yêu cầu nộp bài: các tập tin mã nguồn .c và tập tin khả thực thi .out của các "ví dụ" và bài tập cuối hướng dẫn.

Preferences

[1] Abraham Silberschatz, Peter B. Galvin, Greg Gagne, [2018], Operating System Concepts, 10th edition, John Wiley & Sons, New Jersey.

Programming Problems of Chapter 6 and 7.

1. Bài toán ước lượng giá trị số PI

Một cách giá trị π khá thú vị là sử dụng kỹ thuật Monte Carlo, liên quan đến ngẫu nhiên. Kỹ thuật này hoạt động như sau: Giả sử bạn có một vòng tròn bán kính là 1 nội tiếp trong một hình vuông cạnh là 2, như thể hiện trong hình sau:

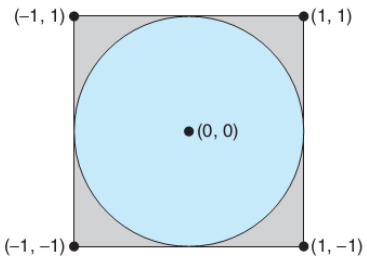


Figure 4.25 Monte Carlo technique for calculating π.

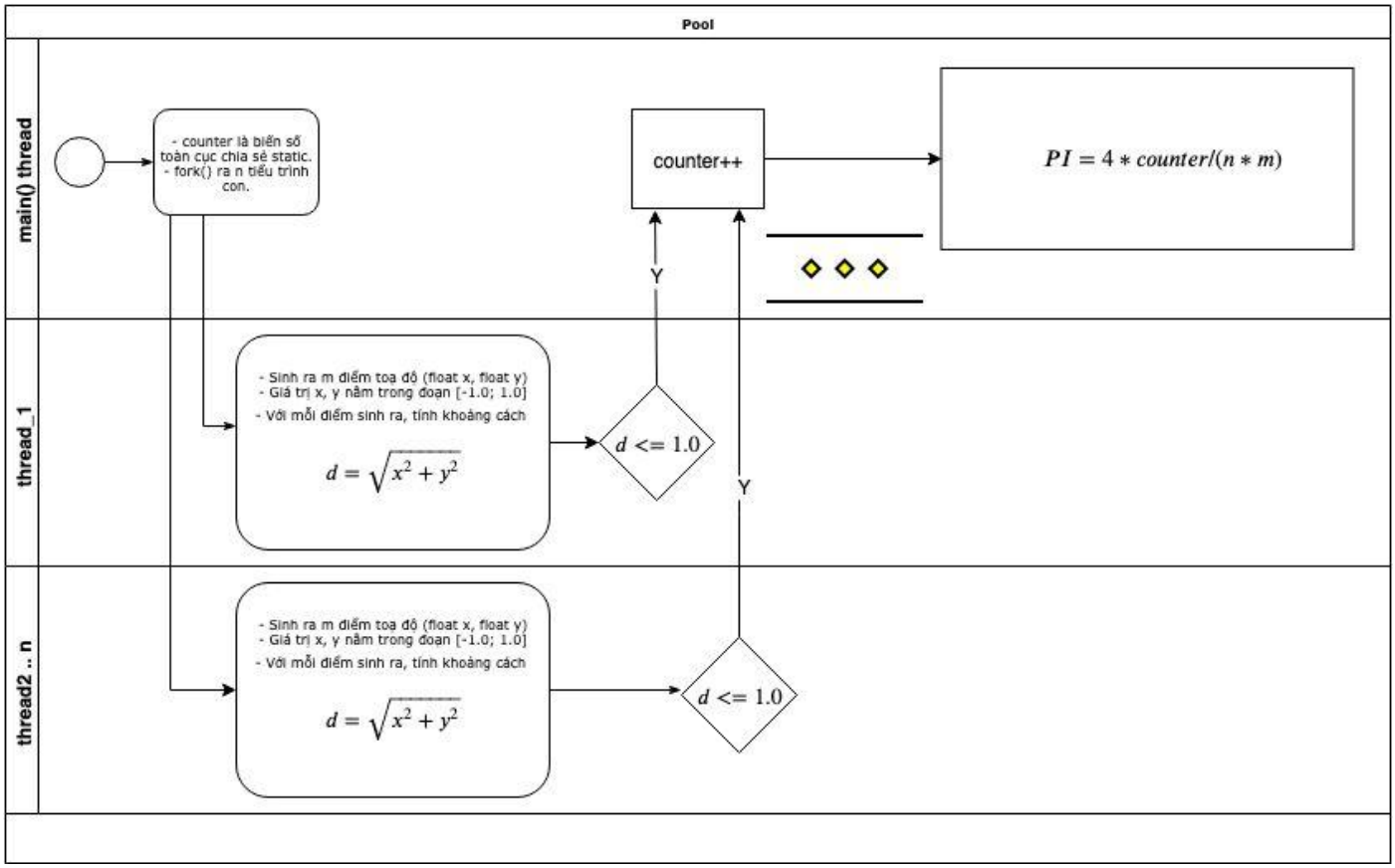
- Đầu tiên, tạo một chuỗi các điểm ngẫu nhiên dưới dạng tọa độ (x, y) đơn giản. Những điểm này phải nằm trong tọa độ Descartes bị ràng buộc hình vuông. Trong tổng số điểm ngẫu nhiên được tạo, một số sẽ nằm trong vòng tròn.
- Tiếp theo, ước tính π bằng cách thực hiện phép tính sau: π = 4 × (số điểm trong vòng tròn) / (tổng số điểm)

Chương trình cần tạo ra n tiểu trình và mỗi tiểu trình sẽ sinh ra m điểm, cũng chính tiểu trình sẽ tính khoảng cách d và cập nhật vào biến counter (là tổng số điểm nằm trong hình tròn, biến toàn cục chia sẻ). Xem lưu đồ kèm theo.

Mỗi tiểu trình cũng cần ghi giá trị các điểm sinh ra và một tập tin m_point.txt.

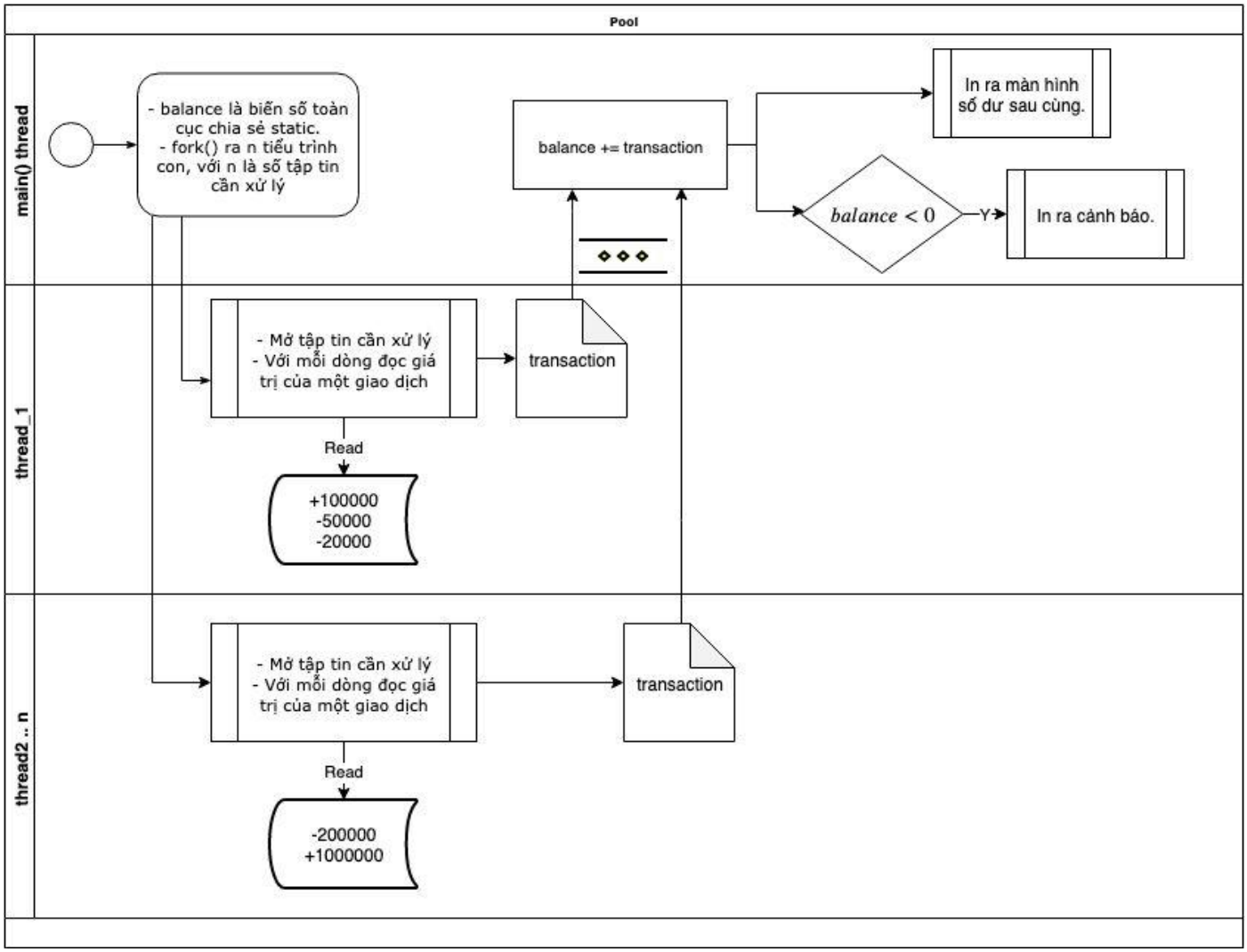
Lời gọi

```
>./pi.out 100 1000
PI = 3.1412
```


Pool
main() thread
thread_1
thread2 .. n
- counter là biến số toàn cục chia sẻ static.
- fork() ra n tiểu trình con.
counter++
$PI = 4 * counter/(n * m)$
Y
- Sinh ra m điểm toạ độ (float x, float y)
- Giá trị x, y nằm trong đoạn [-1.0; 1.0]
- Với mỗi điểm sinh ra, tính khoảng cách
$d = \sqrt{x^2 + y^2}$
$d <= 1.0$
Y
- Sinh ra m điểm toạ độ (float x, float y)
- Giá trị x, y nằm trong đoạn [-1.0; 1.0]
- Với mỗi điểm sinh ra, tính khoảng cách
$d = \sqrt{x^2 + y^2}$
$d <= 1.0$

2. Bài toán Gửi và Rút tiền ngân hàng.

Tình trạng cạnh tranh (race condition) có thể xuất hiện trong nhiều hệ thống máy tính. Hãy xem xét một hệ thống ngân hàng duy trì số dư tài khoản với hai hàm thực thi: deposit(số tiền) và withdraw(số tiền). Hai hàm này được truyền vào số tiền sẽ được gửi hoặc rút từ số dư tài khoản ngân hàng. Giả sử rằng người chồng và người vợ chia sẻ một tài khoản ngân hàng. Một cách đồng thời, người chồng gọi hàm withdraw() và người vợ gọi hàm deposit(). Mô tả làm thế nào một tình trạng cạnh tranh có thể xảy đến và làm cách nào để ngăn chặn tình trạng cạnh tranh này xảy ra.



Nội dung tập tin a.txt

+10 -5 +20	+ thể hiện gửi tiền vào - thể hiện rút tiền ra

Nội dung tập tin b.txt

+20 -8	+ thể hiện gửi tiền vào - thể hiện rút tiền ra

-10	

Nội dung tập tin c.txt

-20 +50	+ thể hiện gửi tiền vào - thể hiện rút tiền ra

Lời gọi

```
>./bank.out a.txt b.txt
Final Balance = 27
>./bank c.txt
Warning: Balance under 0.00
Final Balance = 30
```

3. Bài toán Stack

Cho một stack được chia sẻ, các tiểu trình có thể thực thi POP() và PUSH(i) lên stack vừa nêu, với các giá trị được ghi trong từng tập tin .txt được cho như sau:

Nội dung tập tin a.txt

push(5) pop() push(10) push(20) pop()	Hành vi push kèm theo giá trị cho vào stack Hành vi pop sẽ thông báo giá trị lấy từ stack ra.

Nội dung tập tin b.txt

push(6) pop() push(13) push(21)	

Khi hành vi push(i) được gọi, giá trị i sẽ thêm vào đỉnh stack; hành vi này cần đồng bộ với nhau để tránh tình trạng cạnh tranh. Hành vi pop() sẽ lấy một giá trị ra khỏi stack và in ra màn hình, hoặc thông báo lỗi nếu stack rỗng.

Lời gọi

```
>./stack.out a.txt b.txt
Content of stack: 5
Content of stack: 5, 6
Pop: 6
Content of stack: 5, 10
Content of stack: 5, 10, 20
Pop: 20
Pop: 10
Content of stack: 5, 13
Content of stack: 5, 13, 21
Finish.
```

4. Bài toán đấu giá

Tình trạng cạnh tranh (race condition) có thể xuất hiện trong nhiều hệ thống máy tính. Hãy xem xét một hệ thống đấu giá trực tuyến trong đó giá đấu cao nhất hiện thời cho mỗi mặt hàng phải được duy trì. Một người muốn đặt giá đấu cho một mặt hàng sẽ gọi hàm bid(số_tiền), hàm này sẽ so sánh số tiền đang được đặt giá đấu với giá đấu cao nhất hiện tại. Nếu số tiền vượt quá giá đấu cao nhất hiện thời, giá đấu cao nhất sẽ được đặt thành số tiền mới. Điều này được minh họa dưới đây:

Giả sử rằng có nhiều tập tin mà mỗi tập tin (đại diện cho một người) chứa các số tiền của mỗi lần đặt lệnh. Chương trình cần tạo ra nhiều tiểu trình, mỗi tiểu trình sẽ đọc một tập tin và các tiểu trình đồng thời đặt lệnh đấu giá, có các tình huống sau có thể xảy ra:

- Giá đấu cao hơn giá hiện tại (giá hiện tại là biến số chia sẻ và khởi tạo là 0): giá hiện tại sẽ được cập nhật và tên người thắng cũng sẽ được cập nhật (là tên của tập tin chứa giá đấu đang được xử lý)
- Giá đấu thấp hơn hay bằng giá hiện tại: không cập nhật gì cả.
- Giá đấu cao hơn giá hiện tại nhưng nếu giá hiện tại cao nhất cũng là của người đấu này thì tiểu trình này bị chặn lại cho đến khi người thắng được cập nhật. (Không ai tự bỏ giá cao hơn giá vừa bỏ liền trước đó).

Nội dung tập tin andy.txt

10 50 60 100	Giá đấu lần đầu Giá đấu tiếp theo Giả sử rằng giá đâu luôn dương và tăng dần. Hàng cuối cùng thể hiện giá cao nhất mà người này có thể đấu.

Nội dung tập tin ben.txt

15 40	

Lời gọi

```
>./bid.out andy.txt ben.txt
Value on bid:
Andy 10
Ben 15
Andy 50
Ben 40
Andy 60
The winner is Andy with value 60
```

5. Bài toán Writer - Reader

Xem lại phần giảng lý thuyết slide Ch07.

Nội dung tập tin list.txt

W1R1R2R3W2R3	Hành vi push kèm theo giá trị cho vào stack Hành vi pop sẽ thông báo giá trị lấy từ stack ra.

Lời gọi:

```
>./WR.out list.txt
Writer_1 modifying … Writer_2 done.
Reader_1 reading …
```

```
Reader_2 reading …
Reader_3 reading …
Reader_2 done.
Reader_1 done.
Reader_3 done.
Writer_2 modifying …
```

6. Bài toán xe qua cầu hẹp.

Xem lại phần thảo luận

Nội dung tập tin list.txt

WEWWEEWWE	W thể hiện một xe đi từ phía tây E thể hiện một xe đi từ phía đông.

Lời gọi:

```
>./OB.out list.txt
Car W1 enter
Car W1 out.
Car E1 enter.
Car E1 out.
Car W2 enter.
Car W3 enter.
Car W2 out.
Car W3 out.
Car E2 enter.
Car E3 enter.
Car E2 out.
Car E3 out.
Car W4 enter.
Car W5 enter.
Car W4 out.
Car W5 out.
Car E4 enter.
Car E4 out.
```

7. Hoàn thiện bài định thời đĩa được mô tả trong LAB10_ Contiguous Memory Allocation.

8. Hoàn thiện bài CPU Scheduling, viết theo các tập tin fcfs.c, sjf.c, rr.c để đáp ứng yêu cầu đề bài. Lưu ý chương trình cần xét đến cả khi T-arrival > 0.

9. Hoàn thiện bài định thời đĩa được mô tả trong LAB10_ Disk Scheduling.