**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP MÔN HỌC NMHĐH**

**LAB 5**

**Nhóm:** **22** **Tổ: 01**

**Tên : Nguyễn Thị Anh Thư**

**MSSV: 51900564**

**NỘI DUNG BÁO CÁO KẾT QUẢ**

**Bài 2 A :** 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:

- 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.

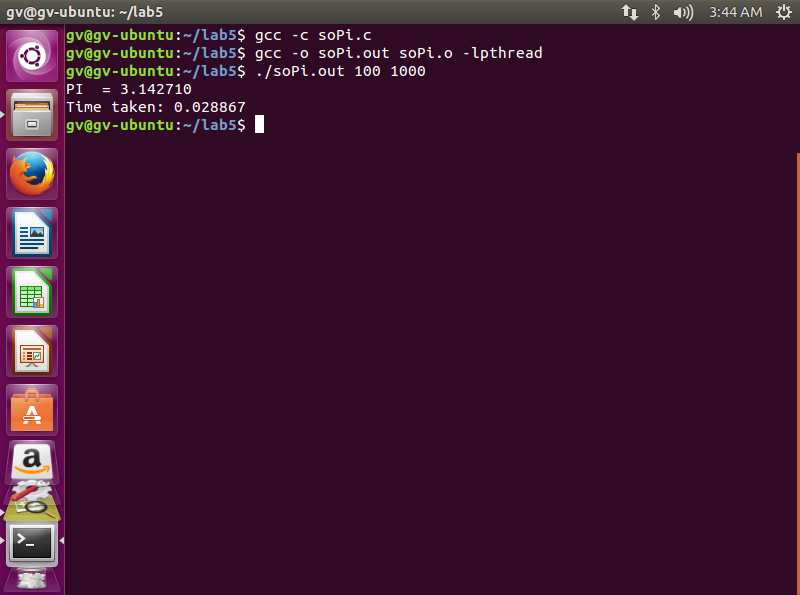
Mỗi tiểu trình cũng cần ghi giá trị các điểm sinh ra và một tập tin m\_point.txt.

**KẾT QUẢ THỰC HIỆN**

**2.1. Phần source code:**

1. #include <stdio.h>
2. #include <pthread.h>
3. #include <time.h>
4. #include <sys/time.h>
5. #include <sys/syscall.h>
6. #include <sys/types.h>
7. #include <sys/stat.h>
8. #include <fcntl.h>
9. #include <string.h>
10. #include <stdlib.h>
11. #define MAX 16
12. #define MAX\_THREAD 40
13. #define MAX\_POINT 10000
14. int counter = 0;
15. int in = 0;
16. void\* count(void\* arg)
17. {
18. int temp =0;
19. float x;
20. float y;
21. for(int i=0; i<MAX\_POINT;i++){
22. x = ((float)rand()/(float)(RAND\_MAX)) \* 2 -1;
23. y = ((float)rand()/(float)(RAND\_MAX)) \* 2 -1;
24. if(x\*x + y\*y <= 1){
25. temp++;
26. }
27. }
28. in+= temp;
29. counter += MAX\_POINT;
30. }
31. int main() {
32. clock\_t t1, t2;
33. t1 = clock();
34. pthread\_t threads[MAX\_THREAD];
35. *// Creating 4 threads*
36. for (int i = 0; i < MAX\_THREAD ; i++)
37. pthread\_create(&threads[i],NULL,&count,(void\*) &counter);
38. *// joining 4 threads i.e. waiting for all 4 threads to complete*
39. for (int i = 0; i < MAX\_THREAD ; i++)
40. pthread\_join(threads[i],NULL);
41. printf("PI = %f\n",(double)in/(double)counter \* 4);
42. t2 = clock();
43. *// time taken by merge sort in seconds*
44. printf("Time taken: %f\n",(t2 - t1) / (double)CLOCKS\_PER\_SEC);
45. return 0;
46. }

**2.2. Kết quả chạy chương trình:**

******

**Bài 2B**: (Bài tập 7.17) Trong câu a, chỉ có một tiểu trình sinh điểm ngẫu nhiên, hãy thay đổi chương trình để sinh

ra nhiều tiểu trình, mỗi tiểu trình tạo ra các điểm ngẫu nhiên và xác định xem các điểm có nằm trong vòng

tròn hay không. Mỗi tiểu trình sẽ phải cập nhật tổng số điểm nằm trong và ngoài vòng tròn (là các biến toàn

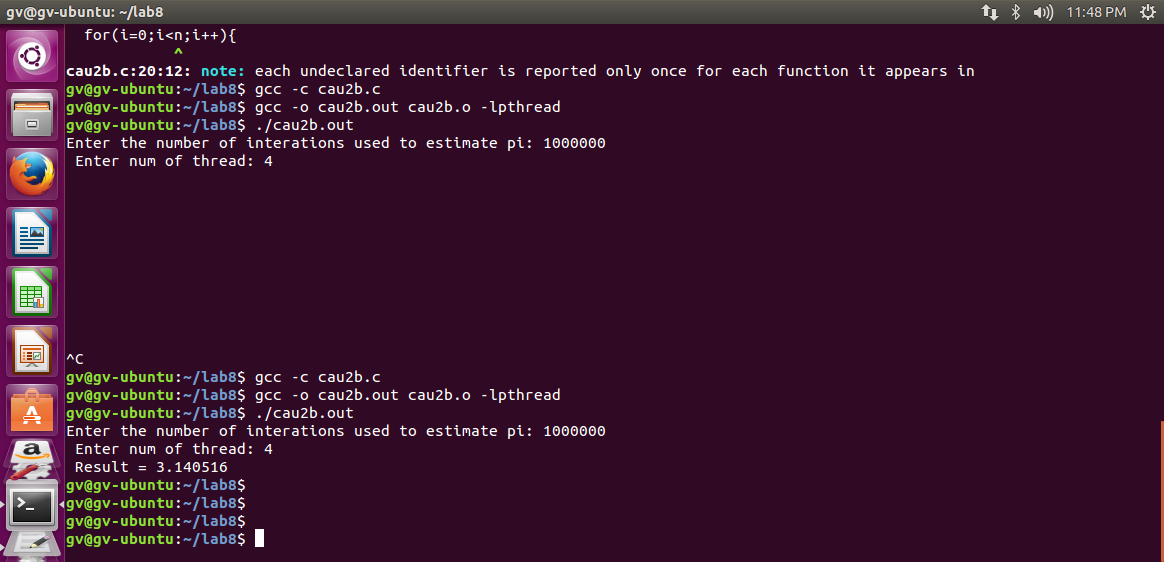
cục). Sử dụng khóa mutex để đảm bảo điều kiện cạnh tranh khi các tiểu trình đồng thời cập nhật giá trị toàn

cục đã mô tả.

**2B.1 Phần source code:**

1. #include <stdio.h>
2. #include <math.h>
3. #include <stdlib.h>
4. #include <string.h>
5. #include <unistd.h>
6. #include<pthread.h>
7. #include <semaphore.h>
8. #define SEED 35791246
9. int count = 0;
10. sem\_t sem;
11. void\* monte(void\* arg)
12. {
13. double x,y,z;
14. int i;
15. srand(SEED);
16. int n = (\*(int\*)arg);
17. for(i=0;i<n;i++){
18. x = (double)rand()/RAND\_MAX;
19. y =(double)rand()/RAND\_MAX;
20. z=x\*x+y\*y;
21. if(z<=1){
22. sem\_wait(&sem);
23. count++;
24. sem\_post(&sem);
25. }
26. }
27. }
28. int main(int argc, char \*argv){
29. nt niter;
30. double pi;
31. printf("Enter the number of interations used to estimate pi: ");
32. scanf("%d", &niter);
33. sem\_init(&sem,0,1);
34. int thread, i;
35. printf(" Enter num of thread: ");
36. scanf("%d", &thread);
37. pthread\_t t[ thread];
38. int leng= niter/thread;
39. for (i =0;i<thread;i++){
40. pthread\_create(&t[i],NULL,monte,&leng);
41. }
42. for(i=0; i<thread;i++){
43. pthread\_join(t[i],NULL);
44. }
45. pi =4\* (double) count/niter;
46. printf( " Result = %lf\n", pi);
47. sem\_destroy(&sem);
48. return 0;
49. }

**2.2 Kết quả chạy chương trình:**

****

**Bài 1**:

Tạo 2 tiểu trình con, một tiểu trình in ra các số lẻ từ 1 đến 11, một tiểu trình in ra số chẵn từ 2 đến 10. Hãy

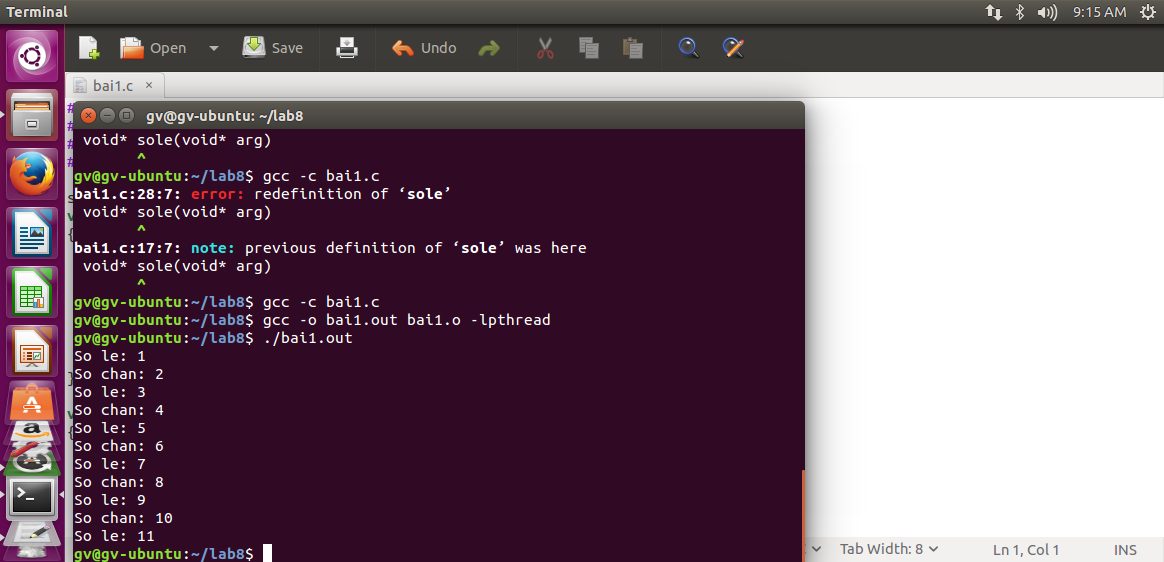
sử dụng semaphore sau cho màn hình in ra dãy số đúng theo thứ tự từ 1 đến 11.

**KẾT QUẢ THỰC HIỆN**

* 1. **Phần source code:**

1. #include <stdio.h>
2. #include <pthread.h>
3. #include <semaphore.h>
4. #include <unistd.h>
5. sem\_t mutex1, mutex2;
6. void\* sochan(void\* arg)
7. {
8. int i;
9. for(i= 2; i <= 11; i = i + 2)
10. {
11. sem\_wait(&mutex1);
12. printf("So chan: %d \n", i);
13. sem\_post(&mutex2);
14. }
15. }
16. void\* sole(void\* arg)
17. {
18. int i;
19. for(i = 1; i <= 11; i=i+2)
20. {
21. sem\_wait(&mutex2);
22. printf("So le: %d\n", i);
23. sem\_post(&mutex1);
24. }
25. }
26. int main()
27. {
28. sem\_init(&mutex1, 0, 0);
29. sem\_init(&mutex2, 0, 1);
30. pthread\_t t1, t2;
31. pthread\_create(&t2, NULL, sole, NULL);
32. pthread\_create(&t1, NULL, sochan, NULL);
33. pthread\_join(t1, NULL);
34. pthread\_join(t2, NULL);
35. sem\_destroy(&mutex1);
36. sem\_destroy(&mutex2);
37. return 0;
38. }
39. return 0;
40. }

**3.2 Kết quả chạy chương trình:**

****

**KẾT LUẬN**

Qua bài thực hành lab8 , em học được cách ước lượng số Pi. Atomic variables, barrier,semaphone, mutex Locks, condition variables, monitor.