**TUGAS SISTEM OPERASI**

**Chapter 5.44**

Exercise 4.23 asked you to design a program using OpenMP that estimated using the Monte Carlo technique. Examine your solution to that program looking for any possible race conditions. If you identify a race condition, protect against it using the strategy outlined in Section 5.10.2

Soal :

1. Untuk apakah tujuan program ini?
2. Section berapa yang menjelaskan program ini
3. Jelaskan logika program ini!
4. Apa guna source code ini?
5. File penjelasan dan source code dimasukkan ke github

Jawab :

1. Menurut kami tujuan dari program ini yaitu untuk mencari pi = 3,14. Kemudian tujuan simulasi Monte Carlo untuk mencari nilai yang mendekati nilai sesungguhnya.
2. Soal yang kami dapat berasal dari bab 4, yang dapat dilihat pada penjelasan 4.5.2
3. a) Menentukan nilai acak untuk mencari pi Monte Carlo

b) Mencari panjang kordinat x,y

c) Melakukan penginilisasian jika terjadi error

d) Menampilkan pi yang mendekati nilai 3,14

1. Source code ini mengandung Mutex yang berfungsi untuk mencari tau konsep pemrograman yang sering digunakan untuk pemecahan masalah multi-threading
2. #include<stdio.h>

#include<unistd.h>

#include<pthread.h>

#include<stdlib.h>

#define random\_1(a,b) ((rand()%(b-a))+a) //The random value will contain a without b

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

int value = 0;

long double valid=0,all=1000000;//The initial effective point is 0, and the total number of points is 9000000

long double pi=0;

void\* fun(void\* arg) {

pthread\_mutex\_lock(&mutex);

long double x,y;//Define the coordinates x,y

int i=1;

for(i=1;i<=9000000;i++)

{

//Get the random value of x, y, 0<=x<100000, 0<=y<100000

x=random\_1(0,100000);

y=random\_1(0,100000);

if(x\*x+y\*y<=10000000000)

valid++;

}

pthread\_mutex\_unlock(&mutex);

return (void\*)0;

}

//Main function

int main()

{

pthread\_t theradId1, threadId2;//Define two child threads

int error;//Define the thread return value error, if the thread is successfully established, return 0, otherwise return the error number

pthread\_mutex\_init(&mutex, NULL);//Mutual exclusion lock initialization

error = pthread\_create(&theradId1, NULL, fun, NULL);

sleep(2);//Use sleep() to control the different concurrent processes of sub-threads

if(error) {

printf("Thread1 creat failed!\n");

exit(1);

}

error = pthread\_create(&threadId2, NULL, fun, NULL);

if(error) {

printf("Thread2 creat failed!\n");

exit(1);

}

pi=valid/9000000\*4;

printf("The valid value is: %Lf\n",valid);

printf("The value of total points is 9000000\n");

printf("The value of π is: %Lf\n",pi);

//Process lock destruction

pthread\_mutex\_destroy(&mutex);

return 0;

}