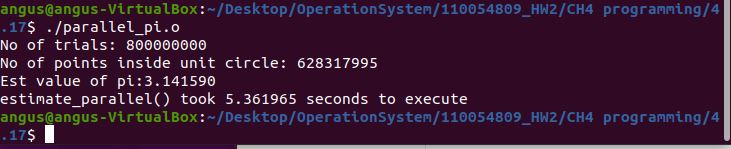
Steps：

1. gcc -pthread parallel\_pi.c -o parallel\_pi.o

2. ./parallel\_pi.o



Source code：

#include <stdio.h>

#include <pthread.h>

#include <math.h>

#include <stdlib.h>

#include <limits.h>

#define SEED 7

unsigned int seed = 7;

// defining the total number of points from which random points are selected

int N = 800000000;

//global variable for number of points inside unit circle

int totalPts = 0;

//Random number generator with linear congruential generator

double RandUint (long i)

{

seed = seed \* 1103515245 + 123456;

return seed / (double)UINT\_MAX;

}

//function to find the number of random points inside the unit circle

void \* Count (void \*X)

{

/\* initialize random numbers \*/

for (int i = 0; i < N; i++)

{

double x = (double)RandUint (i);

double y = (double)RandUint (i);

// checking if it is a unit circle

if (((x \* x) + (y \* y)) <= 1)

{

totalPts++;

}

}

return NULL;

}

//function to estimate pi using pthreads

void EstimateParallel ()

{

srand (SEED);

// defining the thread variable as pthread\_t datatype (4 bytes)

pthread\_t thread;

// creating a pthread which computes the number of points inside the unit circle

pthread\_create (&thread, NULL, &Count, NULL);

//waiting for the termination of the pthread

pthread\_join (thread, NULL);

// estimating the value of pi using the formula given in the question

double estPiValue = (4.0 \* totalPts) / N;

printf ("Number of points: %d \nNumber of points inside unit circle: %d \nEst value of pi:%lf\n",N,totalPts,estPiValue);

}

int main ()

{

// calculating the time taken by the function estimate()

clock\_t t;

t = clock ();

EstimateParallel ();

t = clock () - t;

double timeTaken = ((double)t) / CLOCKS\_PER\_SEC; // in seconds

printf ("EstimateParallel() took %f seconds to execute \n", timeTaken);

return 0;

}