//Anthony Zovich

//Programming Languages 355

//09/06/15

// Threads Programing

//Description: Re-write the Monte Carlo program with threads and then compare the time difference

CODE:

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <stdint.h>

#include <math.h>

#define NUM\_THREADS 8

#define NITER 10000055

#define SEED 32754545

void \*calc\_pi(void \*thread\_id)

{

double x;

double y;

double result;

double pi;

long thread =(long)thread\_id;

int t\_id = (int)(intptr\_t)thread;

float \*count = (float \*)malloc(sizeof(float));

\*count=0;

double total= NITER/NUM\_THREADS;

srand(SEED);

for(int i=0;i<total; i++){

x=(double) rand()/RAND\_MAX;

y=(double) rand()/RAND\_MAX;

result = sqrt((x\*x) + (y\*y));

if(result<=1){

\*count+=1;

}

}

if(t\_id==0){

double remainder = NITER%NUM\_THREADS;

for(int i=0; i<remainder; i++){

double x = (double) rand()/RAND\_MAX;

double y = (double) rand()/RAND\_MAX;

double result = sqrt((x\*x) + (y\*y));

if(result<=1){

\*count+=1; //check if the generated value is inside a unit circle

}

}

}

pthread\_exit((void \*)count); //return the in count

}

int main(int argc, char \*argv[])

{

pthread\_t threads[NUM\_THREADS];

int go;

void \*status;

float total=0;

clock\_t start=clock(), diff;

for(int i=0; i<NUM\_THREADS; i++){

go = pthread\_create(&threads[i], NULL, calc\_pi, (void \*)(intptr\_t)i);

if (go){

printf("ERROR");

exit(-1);

}

}

//join the threads

for(int i=0; i<NUM\_THREADS;i++){

pthread\_join(threads[i], &status);

//printf("Return from thread %ld is : %f\n",t, \*(float\*)status);

total+=\*(float\*)status; //keep track of the total in count

}

diff= clock()- start;

int time= diff\*1000 /CLOCKS\_PER\_SEC;

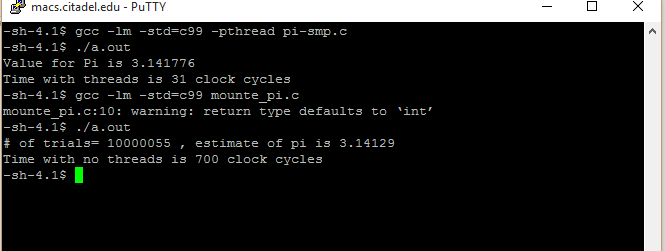
printf("Value for Pi is %f \n",1, 4\*(total/NITER));

printf("Time with threads is %d clock cycles \n", time/1000);

pthread\_exit(NULL);

}

//OUTPUT:



//Conclusion: Programs with threads run faster than programs that do not have threads in them.