**Program1**

**a)Write an OpenMp program that computes the value of PI using Monto-Carlo Algorithm.**

**b)Write a MPI program that computes the value of PI using Monto-Carlo Algorithm.**

**a)Write an OpenMp program that computes the value of PI using Monto-Carlo Algorithm.**

#include<omp.h>

#include<stdio.h>

#include<stdlib.h>

#define SEED 35791246

int main(){

int n = 0, i, count = 0,t,r;

double z, pi, x,y;

printf("sizes: 500,1000,10000,100000 \n");

printf("threads : 1,2,4,8,16\n");

int threads[5] = {1,2,4,8,16};

long long int sizes[4]={500,1000,10000,100000};

printf("\t\t1\t\t2\t\t4\t\t8\t\t16\n");

for(int sizeloop = 0; sizeloop<4; sizeloop++)

{

printf("\n%lld", sizes[sizeloop]);

for(int threadloop =0; threadloop<5;threadloop++)

{

n = sizes[sizeloop];

srand(SEED);

double x = omp\_get\_wtime();

omp\_set\_num\_threads(threadloop);

#pragma omp parallel for private(x,y,z)

for ( i=0; i<n; i++)

{

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

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

z = x\*x+y\*y;

if (z<=1) count++;

}

double y = omp\_get\_wtime();

printf("\t%lf\t", y-x);

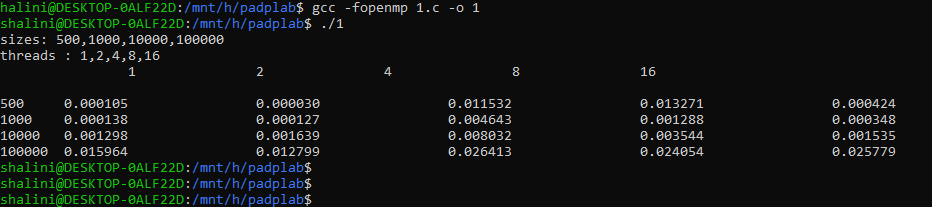
pi=(double)count/n\*4;

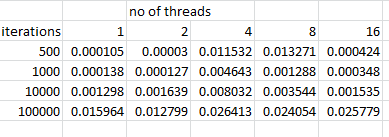
}//threadloop

}//sizeloop

return 0;

}





**b)Write a MPI program that computes the value of PI using Monto-Carlo Algorithm.**

**#include <stdio.h>**

**#include <mpi.h>**

**#include <stdlib.h>**

**#define SEED 3655942**

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

**{**

**long long int iter=1000000;**

**int i,reduced\_count,reduced\_iter,count=0;**

**int rank;**

**double pi,x,y,z,t;**

**//printf("\tsize\t\t1\t\t2\t\t4\t\t8\t\t16\n\n");**

**MPI\_Init(&argc,&argv);**

**iter = 1000000;**

**//printf("input %lld\n",iter);**

**MPI\_Comm\_rank(MPI\_COMM\_WORLD,&rank);**

**if(rank==0)**

**t=MPI\_Wtime();**

**if(rank>0)**

**{**

**srand(SEED);**

**for(i=0;i<iter;i++)**

**{**

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

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

**z=x\*x+y\*y;**

**if(z<=1)**

**count++;**

**}**

**}**

**MPI\_Reduce(&count,&reduced\_count,1,MPI\_INT,MPI\_SUM,0,MPI\_COMM\_WORLD);**

**MPI\_Reduce(&iter,&reduced\_iter,1,MPI\_INT,MPI\_SUM,0,MPI\_COMM\_WORLD);**

**reduced\_iter=reduced\_iter-iter;**

**if(rank==0)**

**{**

**t=MPI\_Wtime()-t;**

**pi=(double)reduced\_count/(double)reduced\_iter \* 4;**

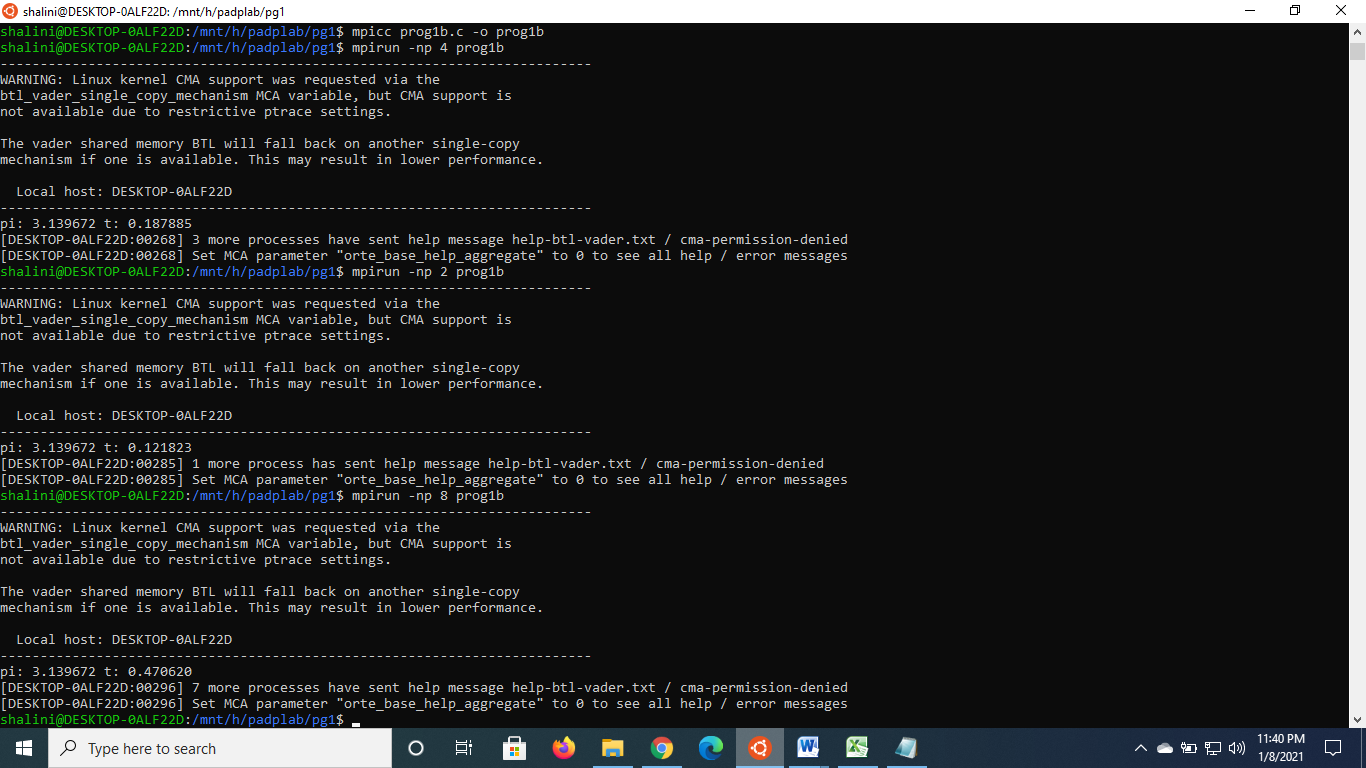
**printf("pi: %lf t: %lf\n",pi,t);**

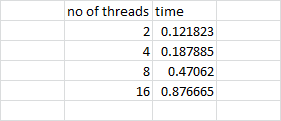
**}**

**MPI\_Finalize();**

**return 0;**

**}**



****