中山大学计算机院本科生实验报告

（2024 学年春季学期）

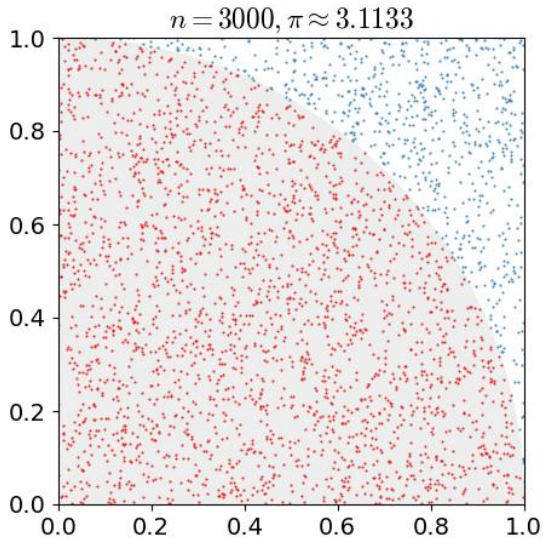
课程名称：并行程序设计　　批改人：

实验	4-Pthreads 并行方程求解及蒙特卡洛	专业（方向）	计算机科学与技术
学号	21307174	姓名	刘俊杰
Email	liujj255@mail2.sysu.edu.cn	完成日期	2024/4/15

1. 实验目的

蒙特卡洛方法求π的近似值

基于 Pthreads 编写多线程程序，使用蒙特卡洛方法求圆周率π近似值。



蒙特卡洛方法与圆周率近似：蒙特卡洛方法是一种基于随机采样的数值计算方法，通过模拟随机时间的发生，来解决各类数学、物理和工程上的问题，尤其是直接解析解决困难或无法求解的问题。其基本思想是：当问题的确切解析解

难以获得时，可以通过随机采样的方式，生成大量的模拟数据，然后利用这些数据的统计特性来近似求解问题。在计算圆周率π值时，可以随机地将点撒在一个正方形内。当点足够多时（见上图），总采样点数量与落在内切圆内采样点数量的比例将趋近于$\pi/4$，可据此来估计π的值。

输入：整数n，取值范围为$[1024, 65536]$

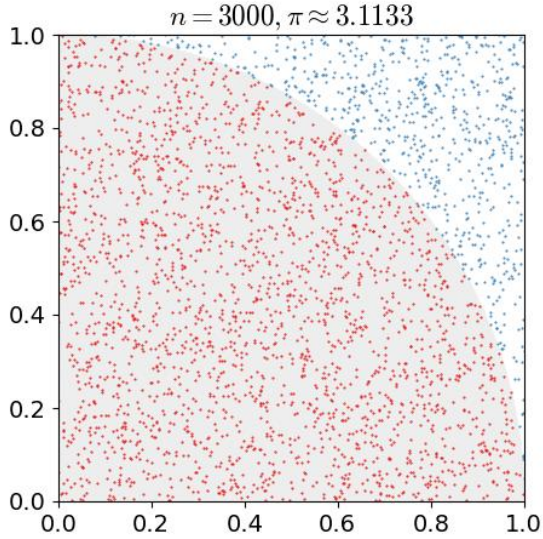
问题描述：随机生成正方形内的n个采样点，并据此估算π的值。

输出：总点数n，落在内切圆内点数m，估算的π值，及消耗的时间t。

要求：基于 Pthreads 编写多线程程序，使用蒙特卡洛方法求圆周率π近似值。讨论程序并行性能。

2. 实验过程和核心代码

2.1 实验思路



蒙特卡洛方法与圆周率近似：蒙特卡洛方法是一种基于随机采样的数值计算方法，通过模拟随机时间的发生，来解决各类数学、物理和工程上的问题，尤其是直接解析解决困难或无法求解的问题。其基本思想是：当问题的确切解析解

难以获得时，可以通过随机采样的方式，生成大量的模拟数据，然后利用这些数据的统计特性来近似求解问题。在计算圆周率π值时，可以随机地将点撒在一个正方形内。当点足够多时（见上图），总采样点数量与落在内切圆内采样点数量的比例将趋近于$\pi/4$，可据此来估计π的值。

2.2 实验过程

①接收运行参数中的线程数数量和产生随机点的个数。

②创建线程，每个线程都运行函数。

③每个线程计算自己负责计算随机点的个数，随机生成 在矩形中的点，利用其与原点的距离判断是否在圆中，如果在则统计圆中点数的数字加 1(利用互斥锁解决线程的竞争问题)。

④根据公式和随机点在圆中的概率计算出Π值，输出计算时间和误差。

2.3 核心代码

①接收运行参数中的线程数数量和产生随机点的个数。

```
double pi; // 估计出的pi值
long long num_threads; // 线程数目
long long num_total_points; // 随机产生点的个数
long long num_points_inside_circle = 0; // 随机产生点落在圆内的个数
pthread_mutex_t sum_mutex; // 定义互斥锁
```

```
num_total_points = atoll(argv[1]);
num_threads = atoll(argv[2]);

printf("产生点的个数: %lld\n", num_total_points);
printf("线程数量: %lld\n", num_threads);

// 初始化互斥锁
pthread_mutex_init(&sum_mutex, NULL);

// 获取开始时间
struct timeval start_time, end_time;
gettimeofday(&start_time, NULL);
```

②创建线程，每个线程都运行函数。

```
// 创建线程
pthread_t threads[num_threads];

for (int i = 0; i < num_threads; i++) {
    pthread_create(&threads[i], NULL, calculate_pi, (void*)i);
}
```

③每个线程计算自己负责计算随机点的个数，随机生成 在矩形中的点，利用其与原点的距离判断是否在圆中，如果在则统计圆中点数的数字加 1(利用互斥锁解决线程的竞争问题)。

```
// 线程函数，计算pi
void *calculate_pi(void *rank) {
    int my_rank = (long)rank;
    long long per_calculate_num = num_total_points / num_threads;
    long long extra_num = num_total_points % num_threads;
    long long start = my_rank < extra_num ? (per_calculate_num + 1) * my_rank : per_calculate_num * my_rank + extra_num;
    long long end = my_rank < extra_num ? (per_calculate_num + 1) * (my_rank + 1) : per_calculate_num * (my_rank + 1) + extra_num;

    unsigned int seed = (unsigned int)time(NULL) + my_rank; // 使用当前时间和线程编号作为种子
    for (long long i = start; i < end; i++) {
        // 使用rand_r保证多线程随机数的安全
        double x = (double)rand_r(&seed) / RAND_MAX;
        double y = (double)rand_r(&seed) / RAND_MAX;
        if (x * x + y * y <= 1) {
            // 使用互斥锁保护临界区
            pthread_mutex_lock(&sum_mutex);
            num_points_inside_circle += 1;
            pthread_mutex_unlock(&sum_mutex);
        }
    }
    return NULL;
}
```

④根据公式和随机点在圆中的概率计算出Π值，输出计算时间和误差。

```c
for (int i = 0; i < num_threads; i++) {
    pthread_create(&threads[i], NULL, calculate_pi, (void*)i);
}

// 等待线程结束
for (int i = 0; i < num_threads; i++) {
    pthread_join(threads[i], NULL);
}

// 获取结束时间
gettimeofday(&end_time, NULL);

// 计算消耗的时间
double elapsed_time = (end_time.tv_sec - start_time.tv_sec) * 1000.0; // 将秒转换为毫秒
elapsed_time += (end_time.tv_usec - start_time.tv_usec) / 1000.0; // 将微秒转换为毫秒

// 计算估计的π值
pi = (double)num_points_inside_circle / num_total_points * 4;

// 输出π的误差值
double error = fabs(pi - M_PI);
printf("The pi: %lf\n", pi);
printf("Error: %lf\n", error);
printf("Computing time: %.2f ms\n", elapsed_time);

// 销毁互斥锁
pthread_mutex_destroy(&sum_mutex);

return 0;
```

3. 实验结果

随机点个数为 1024

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 1028 1
产生点的个数：1028
线程数量：1
The pi: 3.128405
Error: 0.013188
Computing time: 0.94 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 1028 2
产生点的个数：1028
线程数量：2
The pi: 3.178988
Error: 0.037396
Computing time: 0.45 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 1028 4
产生点的个数：1028
线程数量：4
The pi: 3.249027
Error: 0.107435
Computing time: 0.51 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 1028 8
产生点的个数：1028
线程数量：8
The pi: 3.260700
Error: 0.119108
Computing time: 0.94 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 1028 16
产生点的个数：1028
线程数量：16
The pi: 3.163424
Error: 0.021831
Computing time: 1.77 ms
ljj@ljj-virtual-machine:~/parrallel_programming$
```

随机点个数为 2048

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 2048 1
产生点的个数：2048
线程数量：1
The pi: 3.107422
Error: 0.034171
Computing time: 0.40 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 2048 2
产生点的个数：2048
线程数量：2
The pi: 3.087891
Error: 0.053702
Computing time: 0.36 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 2048 4
产生点的个数：2048
线程数量：4
The pi: 3.117188
Error: 0.024405
Computing time: 0.58 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 2048 8
产生点的个数：2048
线程数量：8
The pi: 3.199219
Error: 0.057626
Computing time: 1.03 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 2048 16
产生点的个数：2048
线程数量：16
The pi: 3.121094
Error: 0.020499
Computing time: 1.41 ms
ljj@ljj-virtual-machine:~/parrallel_programming$
```

随机点个数为 4096

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 4096 1
产生点的个数：4096
线程数量：1
The pi: 3.158203
Error: 0.016610
Computing time: 0.60 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 4096 2
产生点的个数：4096
线程数量：2
The pi: 3.166016
Error: 0.024423
Computing time: 0.40 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 4096 4
产生点的个数：4096
线程数量：4
The pi: 3.141602
Error: 0.000009
Computing time: 0.76 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 4096 8
产生点的个数：4096
线程数量：8
The pi: 3.175781
Error: 0.034189
Computing time: 1.20 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 4096 16
产生点的个数：4096
线程数量：16
The pi: 3.206055
Error: 0.064462
Computing time: 2.05 ms
ljj@ljj-virtual-machine:~/parrallel_programming$
```

随机点个数为 8192

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 8192 1
产生点的个数：8192
线程数量：1
The pi: 3.116699
Error: 0.024893
Computing time: 0.66 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 8192 2
产生点的个数：8192
线程数量：2
The pi: 3.118652
Error: 0.022940
Computing time: 0.50 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 8192 4
产生点的个数：8192
线程数量：4
The pi: 3.150391
Error: 0.008798
Computing time: 0.76 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 8192 8
产生点的个数：8192
线程数量：8
The pi: 3.152344
Error: 0.010751
Computing time: 1.34 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 8192 16
产生点的个数：8192
线程数量：16
The pi: 3.162109
Error: 0.020517
Computing time: 2.08 ms
ljj@ljj-virtual-machine:~/parrallel_programming$
```

随机点个数为 16384

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 16384 1
产生点的个数：16384
线程数量：1
The pi: 3.128418
Error: 0.013175
Computing time: 0.89 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 16384 2
产生点的个数：16384
线程数量：2
The pi: 3.140869
Error: 0.000724
Computing time: 0.85 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 16384 4
产生点的个数：16384
线程数量：4
The pi: 3.150635
Error: 0.009042
Computing time: 1.30 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 16384 8
产生点的个数：16384
线程数量：8
The pi: 3.132080
Error: 0.009513
Computing time: 1.50 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 16384 16
产生点的个数：16384
线程数量：16
The pi: 3.135498
Error: 0.006095
Computing time: 2.53 ms
ljj@ljj-virtual-machine:~/parrallel_programming$
```

随机点个数为 32768

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 32768 1
产生点的个数：32768
线程数量：1
The pi: 3.140869
Error: 0.000724
Computing time: 1.90 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 32768 2
产生点的个数：32768
线程数量：2
The pi: 3.160278
Error: 0.018686
Computing time: 1.73 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 32768 4
产生点的个数：32768
线程数量：4
The pi: 3.136597
Error: 0.004996
Computing time: 1.98 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 32768 8
产生点的个数：32768
线程数量：8
The pi: 3.154053
Error: 0.012460
Computing time: 2.67 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 32768 16
产生点的个数：32768
线程数量：16
The pi: 3.143799
Error: 0.002206
Computing time: 3.69 ms
ljj@ljj-virtual-machine:~/parrallel_programming$
```

随机点个数为 65536

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 65536 1
产生点的个数：65536
线程数量：1
The pi: 3.135498
Error: 0.006095
Computing time: 2.32 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 65536 2
产生点的个数：65536
线程数量：2
The pi: 3.142700
Error: 0.001108
Computing time: 3.25 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 65536 4
产生点的个数：65536
线程数量：4
The pi: 3.146484
Error: 0.004892
Computing time: 4.36 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 65536 8
产生点的个数：65536
线程数量：8
The pi: 3.141479
Error: 0.000113
Computing time: 4.32 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab4 65536 16
产生点的个数：65536
线程数量：16
The pi: 3.146851
Error: 0.005258
Computing time: 4.19 ms
ljj@ljj-virtual-machine:~/parrallel_programming$
```

线程数	随机点数量						
	1024	2048	4096	8192	16384	32768	65536
1	3.128405 0.013188 0.94ms	3.107422 0.034171 0.40ms	3.158203 0.016610 0.60ms	3.116699 0.024893 0.66ms	3.128418 0.013175 0.89ms	3.140869 0.000724 1.90ms	3.135498 0.006095 2.32ms
2	3.178988 0.03736 0.45ms	3.08781 0.053702 0.36ms	3.166016 0.024423 0.40ms	3.118652 0.022940 0.50ms	3.140869 0.000724 0.85ms	3.160278 0.018686 1.73ms	3.142700 0.001108 3.25ms
4	3.249027 0.107438 0.51ms	3.117188 0.024405 0.58ms	3.141602 0.000009 0.76ms	3.150391 0.008798 0.76ms	3.150635 0.009042 1.30ms	3.136597 0.004996 1.98ms	3.149484 0.004892 4.36ms
8	3.260700 0.199108 0.94ms	3.199219 0.057626 1.03ms	3.175781 0.034189 1.20ms	3.152344 0,010751 1.34ms	3.132080 0.009513 1.50ms	3.154053 0.12460 2.67ms	3.141479 0.000113 4..32ms
16	3.163424 0.021831 1.77ms	3.121094 0.020499 1.41ms	3.206055 0.064462 2.06ms	3.162109 0.020517 2.08ms	3.135498 0.006095 2.53ms	3.143799 0.002206 3.69ms	3.146851 0.005258 4.19ms

4. 实验感想

4.1 问题:使用 rand 时产生的随机数都一样

解决方法：
① 使用 time()和线程号作为种子
② 使用 rand()

rand()和 rand_r 的区别:

rand() 函数:
rand() 是 C 语言标准库中提供的伪随机数生成函数。
它使用一个全局状态来生成随机数。这个状态在多线程环境下是共享的，因此 rand() 在多线程环境下不是线程安全的。
在多线程环境下，如果多个线程同时调用 rand() 函数，可能会导致竞态条件，使得生成的随机数不是预期的结果。

rand_r() 函数:
rand_r() 是 POSIX 标准中定义的伪随机数生成函数，具有可重入性（reentrant），也就是说，它可以安全地在多线程环境中使用。
rand_r() 函数的使用方式不同于 rand()。它接受一个指向随机数生成器状态的指针作为参数，因此每个线程都可以有自己的状态，从而避免了竞态条件。通过传递不同的状态指针，每个线程可以独立地生成随机数序列，而不会相互干扰。
为了在多线程环境中安全地生成随机数，应该使用 rand_r() 而不是 rand()。通过使用 rand_r()，可以确保每个线程都有自己的随机数生成器状态，从而避免了线程间的竞争和不确定性

4.2 实验感悟

①使用多线程可以加速计算过程，特别是在生成大量随机点的情况下。每个线程负责生成一部分点，并统计落在内切圆内的点数，从而提高计算效率。同时合理选择线程数目很重要，线程数目多使得估计的Π值越准确。

②在多线程环境中，需要特别注意随机数的生成。使用 rand_r() 函数可以确保在多线程情况下生成的随机数是线程安全的，从而避免竞态条件。

③由于随机性是蒙特卡洛方法的本质，因此每次计算的结果可能会有一定的波动。通过多次运行程序并取平均值，可以提高估算结果的准确性。