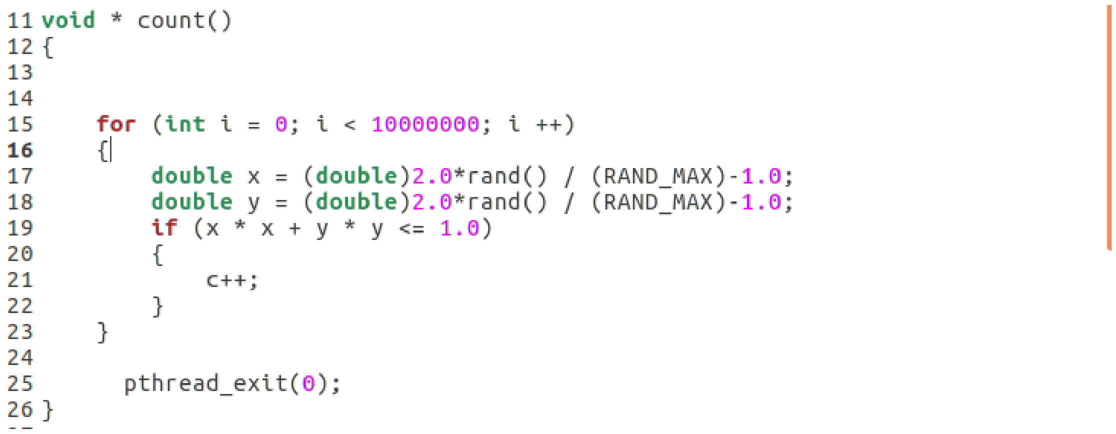
Calculating PI using *Monte Carlo Method*

刘宇 2014213404

In the project, the program should have two threads. One thread can count the number of points that occur within the circle and store that result in a global variable. And the parent thread can calculate and output the estimated value of PI.
Firstly, I create a function called count. And the thread that I create will call the function.

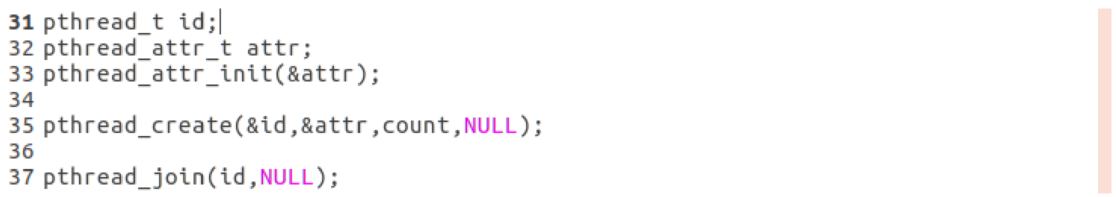


```c
void * count()
{


    for (int i = 0; i < 10000000; i ++)
    {
        double x = (double)2.0*rand() / (RAND_MAX)-1.0;
        double y = (double)2.0*rand() / (RAND_MAX)-1.0;
        if (x * x + y * y <= 1.0)
        {
            c++;
        }
    }

      pthread_exit(0);
}
```

In the function, the number of loop is the number of (x,y) points.
What's more, x and y are all random numbers ranging from -1,+1. Because the range of rand() / (RAND_MAX) is from 0 to 1, 2.0*rand() / (RAND_MAX)-1.0 is from -1 to +1.

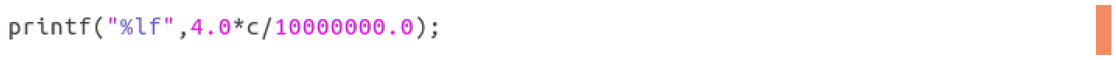
c is a global variable and it is the number of points located in cycle.
pthread_exit(0) means that once thread finishes the function, the thread will exit.

In the main function, I create the thread by using pthread_create(). After creating the thread, I use pthread_join() to let thread start executing.



```c
pthread_t id;
pthread_attr_t attr;
pthread_attr_init(&attr);

pthread_create(&id,&attr,count,NULL);

pthread_join(id,NULL);
```

After the thread finishing tasks and exiting, the parent thread will calculate PI by
using

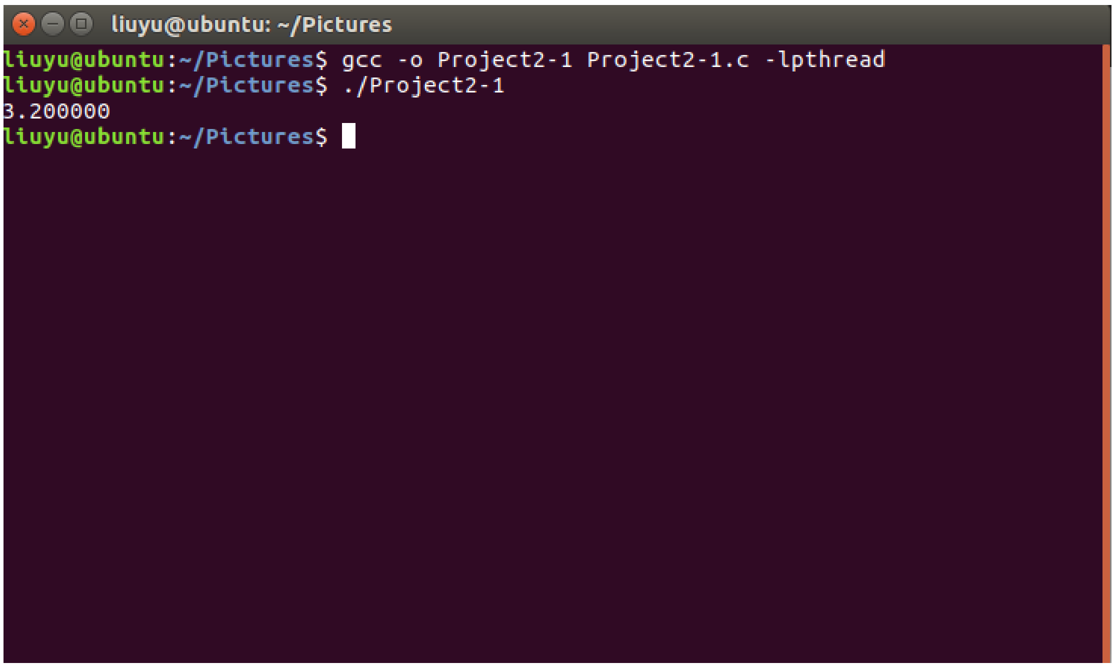


```c
printf("%lf",4.0*c/10000000.0);
```

And the value of PI will also be displayed.

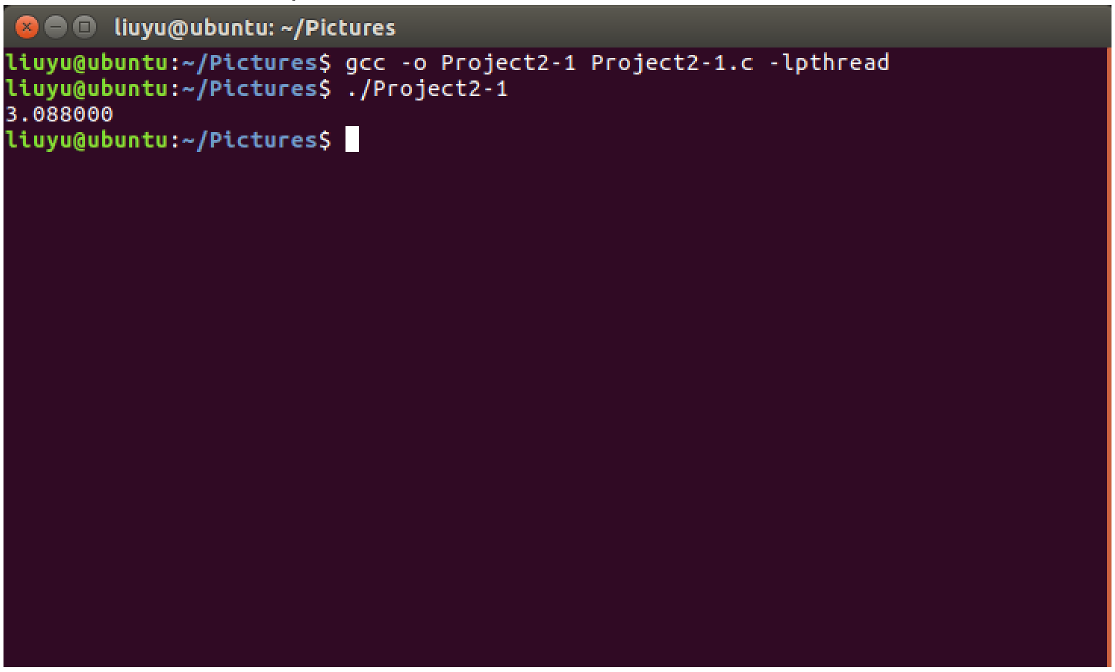
Test part:

The number of total points is 10, PI is 3.2



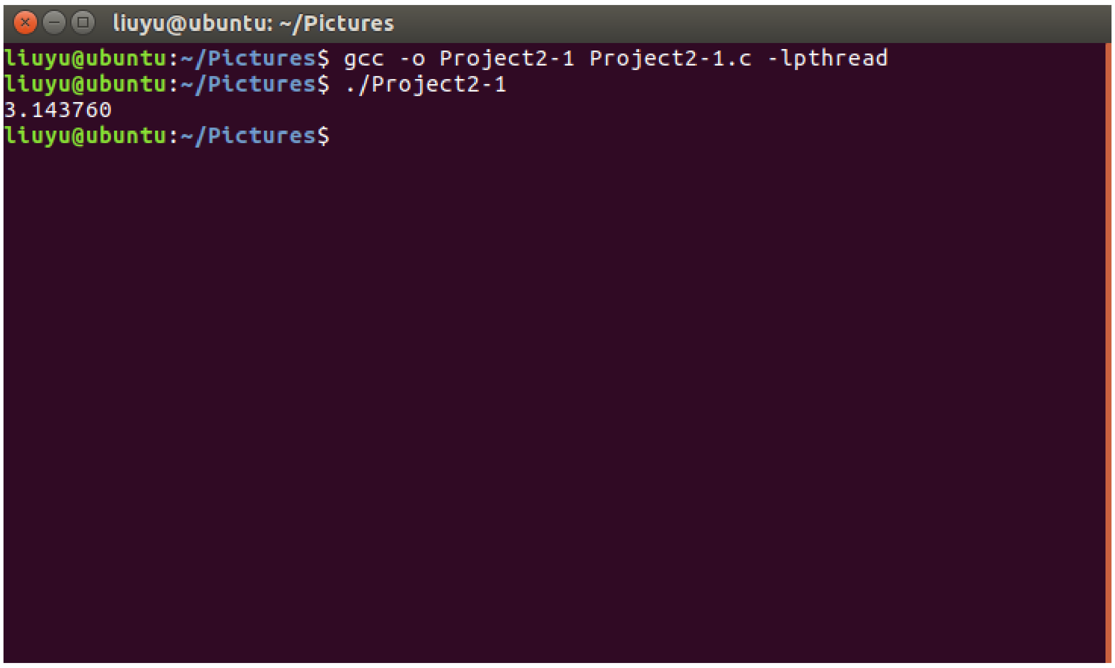
```
liuyu@ubuntu:~/Pictures$ gcc -o Project2-1 Project2-1.c -lpthread
liuyu@ubuntu:~/Pictures$ ./Project2-1
3.200000
liuyu@ubuntu:~/Pictures$
```

The number of total points is 1000, PI is 3.088



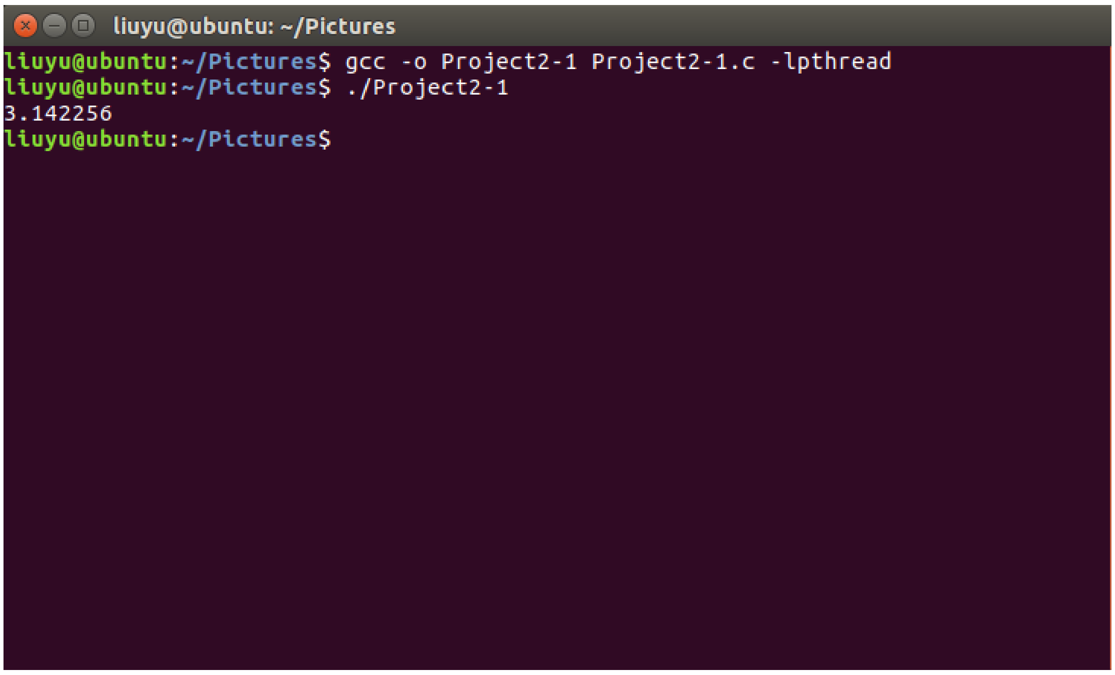
```
liuyu@ubuntu:~/Pictures$ gcc -o Project2-1 Project2-1.c -lpthread
liuyu@ubuntu:~/Pictures$ ./Project2-1
3.088000
liuyu@ubuntu:~/Pictures$
```

The number of total points is 100000, PI is 3.143760



```
liuyu@ubuntu: ~/Pictures
liuyu@ubuntu:~/Pictures$ gcc -o Project2-1 Project2-1.c -lpthread
liuyu@ubuntu:~/Pictures$ ./Project2-1
3.143760
liuyu@ubuntu:~/Pictures$
```

The number of total points is 10000000, PI is 3.142256



```
liuyu@ubuntu: ~/Pictures
liuyu@ubuntu:~/Pictures$ gcc -o Project2-1 Project2-1.c -lpthread
liuyu@ubuntu:~/Pictures$ ./Project2-1
3.142256
liuyu@ubuntu:~/Pictures$
```

So I find that when the number of total points is increased, the value of PI I calculate is more accurate.