Multithreaded program estimating PI

In the project, I use some functions to achieve mutex locks.

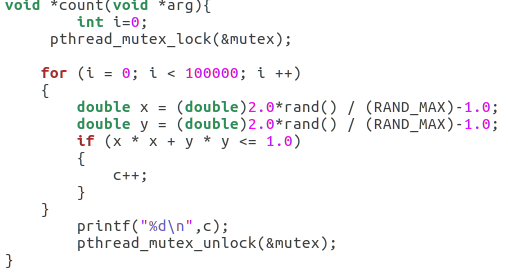
pthread\_mutex\_t is used to define a mutex lock. And the mutex lock I define is the global value which can be available to all threads.

pthread\_mutex\_init(&mutex,NULL) is used to initialize the mutex lock.

pthread\_mutex\_lock(&mutex) means only one thread can get the mutex lock, other threads can only wait.

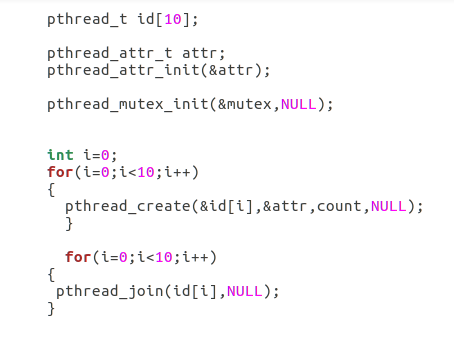
pthread\_mutex\_unlock(&mutex) means that the mutex lock will be released and other threads will have chance to get the mutex lock.

pthread\_mutex\_destroy(&mutex) means that the mutex lock will be destroyed if no threads will use it.



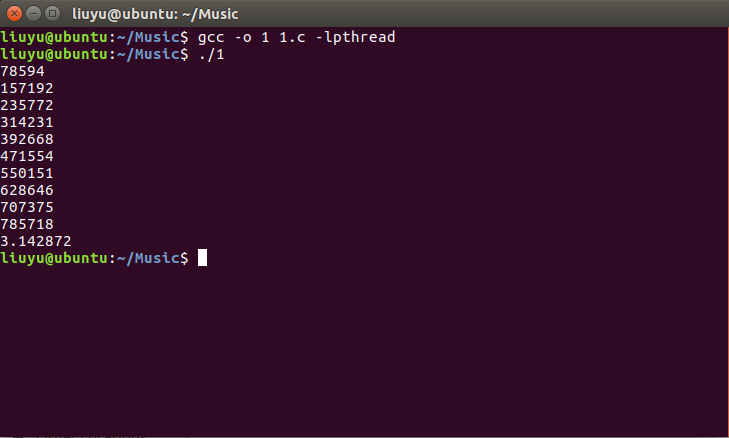
You can see from the picture. If a thread get the mutex lock, it will generate 1000000 random points. If the random points are in the circle, variable c is increased by one. Other threads that don’t get the mutex lock, will not access to the variable c.

Then I use array to generate ten threads.



And these ten threads will execute the function count. But only the thread that get the mutex lock can access to the variable c so that variable c will be available to only one thread at one time.

When I execute my program, the PI is 3.142872. And variable c is printed orderly. You can see that c is incremented by almost same value for each c. That means that only one thread can access to the variable c at each time. Mutex lock is effective.



But if I don’t add mutex lock to my program, c value is disorder. Many threads can access to c at the same time. And PI is not correct.

