80 QMutex和QMutexLocker

80.1 QMutex

互斥体意在确保程序中的一段代码任何时候最多被一个线程执行。

```
void QMutex::lock();                    // 锁定互斥体，若该互斥体已被其它线程锁定则阻塞，直至其被解锁
void QMutex::unlock();                  // 解锁互斥体，阻塞于锁定该互斥体的线程将被唤醒，从锁定中返回
bool QMutex::try_lock();                // 试锁互斥体，锁定成功返回true，已被其它线程锁定返回false
bool QMutex::tryLock(int timeout = 0); // 试锁互斥体，锁定成功返回true，已被其它线程锁定最多阻塞timeout毫秒后返回false
```

80.2 QMutexLocker

QMutexLocker类的构造函数接受一个QMutex类型的互斥体对象参数并将其锁定，在析构函数中将其解锁。因此QMutexLocker对象的生命周期决定了被互斥体保护的代码范围。

```
QMutexLocker::QMutexLocker(QMutex* mutex);
```

80.3 案例

80.3.1 创建项目

通过QtCreator，在C:\Users\Minwei\Projects\Qt路径下，创建名为Mutex的控制台（Console）项目。

80.3.2 实现功能

C:\Users\Minwei\Projects\Qt\Mutex\main.cpp：

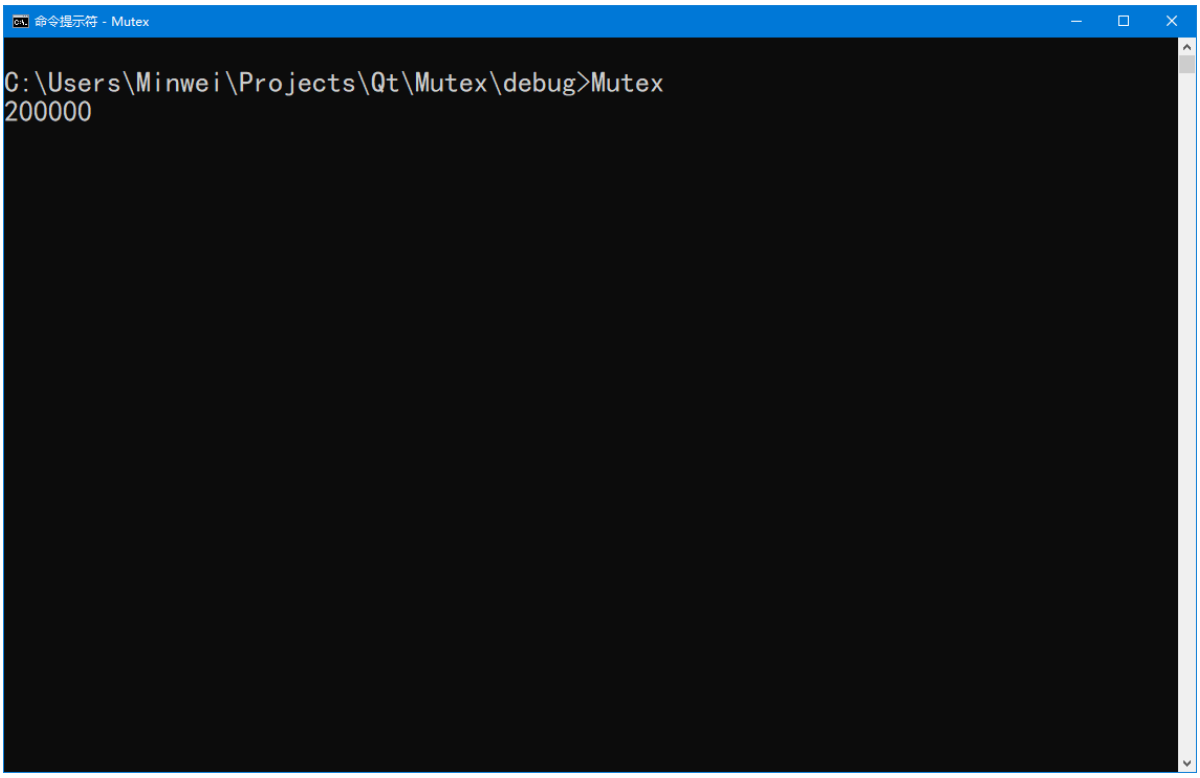
```
#include <iostream>
using namespace std;

#include <QCoreApplication>
#include <QThread>
#include <QMutex>

unsigned int g_cn = 0;
QMutex g_mutex;

class WorkThread: public QThread
{
public:
    WorkThread(char ch): m_ch(ch)
    {
    }

protected:
    /* 调用lock方法锁定互斥体
    void run()
    {
```

```
        for (unsigned int i = 0; i < 1000000; i++) {
            g_mutex.lock();
            ++g_cn;
            g_mutex.unlock();
        }
    }
    */
    /* 调用try_lock方法锁定互斥体
    void run()
    {
        for (unsigned int i = 0; i < 1000000; i++) {
            while (!g_mutex.try_lock()) cout << m_ch << flush;
            ++g_cn;
            g_mutex.unlock();
        }
    }
    */
    /* 调用tryLock方法锁定互斥体
    void run()
    {
        for (unsigned int i = 0; i < 1000000; i++) {
            while (!g_mutex.tryLock(10)) cout << m_ch << flush;
            ++g_cn;
            g_mutex.unlock();
        }
    }
    */
    // 利用QMutexLocker对象的生命周期管理锁定范围
    void run()
    {
        for (unsigned int i = 0; i < 1000000; i++) {
            QMutexLocker locker(&g_mutex);
            ++g_cn;
        }
    }

private:
    char m_ch;
};

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    WorkThread work1('|'), work2('_');

    work1.start();
    work2.start();

    work1.wait();
    work2.wait();

    cout << g_cn << endl;

    return a.exec();
}
```

80.3.3 测试验证

80.3.3.1 调用lock方法锁定互斥体

运行效果如图所示：

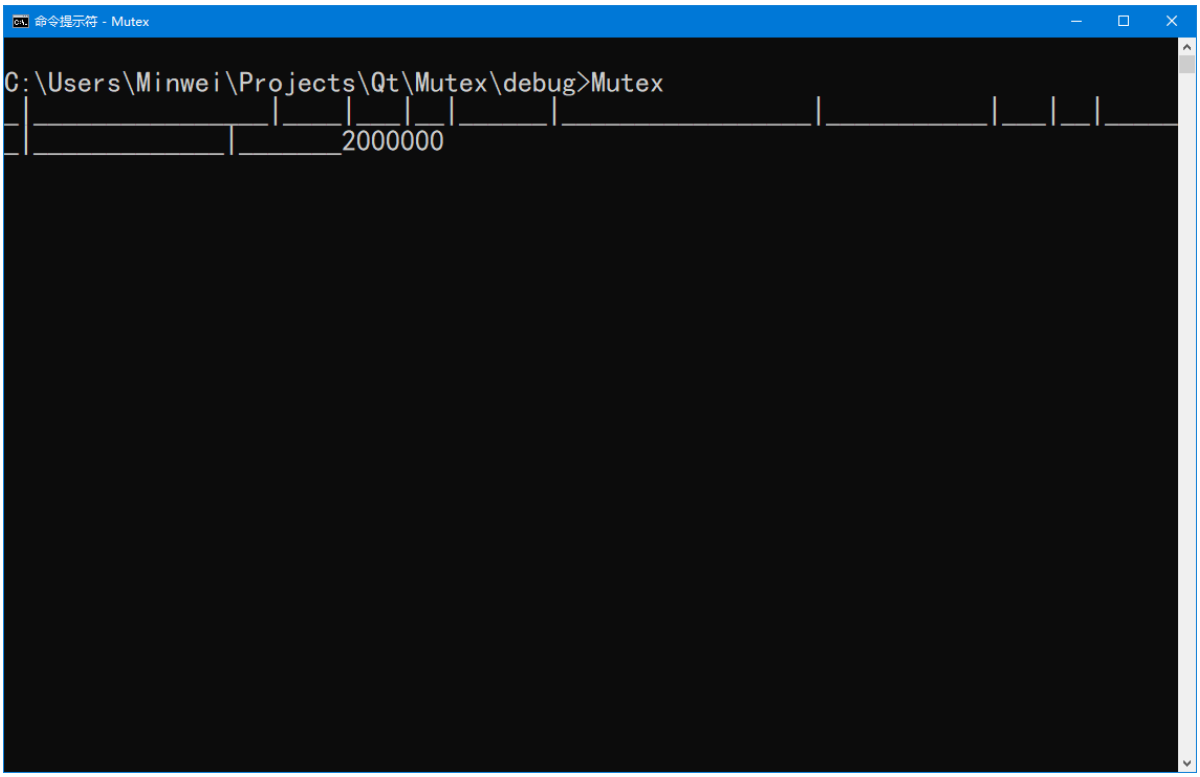


80.3.3.2 调用try_lock方法锁定互斥体

运行效果如图所示：

80.3.3.3 调用tryLock方法锁定互斥体

运行效果如图所示：



80.3.3.4 利用QMutexLocker对象的生命周期管理锁定范围

运行效果如图所示：