20220623

1. Qt 线程间数据通讯及数据共享

方法一：全局变量或全局函数

缺点：全局变量长时间占用内存，影响程序空间使用率，且全局变量修改影响整个程序，程序的安全性无法保证；

方法二：信号槽

只有 QObject 类及其派生的类才能使用信号和槽的机制，在线程间使用信号槽进行通信时，槽函数必须使用元数据类型的参数；如果使用自定义的数据类型，需要在 connect 之前将其注册（qRegisterMetaType）为元数据类型。

```
#include <QMetaType>

typedef struct MYSTRUCT{
    unsigned char data1[4];
    int data2;
    double data3;
    ...
}MYSTRUCT;

MYSTRUCT m_struct;

void Mainwindow::init()
{
    qRegisterMetaType<MYSTRUCT>("mystruct");
    //MYSTRUCT 为自定义数据类型
    //mystruct 为定义的名称
    connect(this, &Mainwindow::signal_send, this, &Mainwindow::slot_receive);
}

void Mainwindow::doSomething()
{
    m_struct.data1[0] = 1;
    m_struct.data1[1] = 1;
    m_struct.data1[2] = 1;
    m_struct.data1[3] = 1;
    m_struct.data2 = 2;
    m_struct.data3 = 3.0;

    emit signal_send(m_struct);
    //在头文件中声明信号 void signal_send(const MYSTRUCT mystruct);
}

void Mainwindow::slot_receive(const MYSTRUCT mystruct)
{

}
```

线程间用信号槽传递参数的话，要加 const，因为 const 文字常量存在常量区中，生命周期和程序一样长，可以避免 slot 调用的时候参数的运行期已过造成引用无效；

connect 函数的第五个参数：

（1）Qt::AutoConnection

如果发射信号的线程和执行槽函数的线程是同一个线程，此时等同于 Qt::DirectConnection；如果不在同一个线程，就等同于 Qt::QueuedConnection，是 connect 函数的默认参数；

（2）Qt::DirectConnection

发射信号和执行槽函数由同一个线程（信号发射的线程）完成，信号发射后槽函数立马执行，执行完毕后才继续执行“emit 信号”后面的代码，即“emit 信号”是阻塞的；

（3）Qt::QueuedConnection

发射信号的线程和执行槽函数的线程不是在同一个线程，此时发射信号的线程不阻塞，马上返回，当执行槽函数的线程被 CPU 调度时，就会执行槽函数；

（4）Qt::BlockingQueuedConnection

和 Qt::QueuedConnection 基本一样，只是发射信号的线程会被阻塞，直到槽函数被执行完毕，如果使用这个属性，需确保发射信号的线程与执行槽函数的线程不同，否则将发生死锁；

（5）Qt::UniqueConnection

唯一关联，该类型可以和上面的类型通过“|”符号配合使用，同一个信号与同一个槽只能调用 connect 一次，不能多次调用；

方法三：共享内存

QSharedMemory