涉及到跨线程通信时，我们可以使用Qt::QueuedConnection作为连接类型来实现异步的信号槽通信。下面是一个简单的示例代码：

#include <QCoreApplication>

#include <QThread>

#include <QDebug>

/\*

定义一个名为Worker的类，继承自QObject，用于执行耗时操作。其中process()函数模拟耗时操作，每隔1秒打印一次"Working..."，共打印5次。当操作完成后，发送结果信号resultReady

\*/

class Worker : public QObject

{

Q\_OBJECT

public:

explicit Worker(QObject \*parent = nullptr) : QObject(parent) {}

signals:

void resultReady(const QString &result);

public slots:

void process()

{

// 模拟耗时操作

for (int i = 0; i < 5; ++i) {

QThread::sleep(1);

qDebug() << "Working...";

}

// 发送结果信号

emit resultReady("Done");

}

};

/\*

定义一个名为Receiver的类，继承自QObject，用于处理Worker发送的结果信号。其中handleResult()函数用于打印结果并退出应用程序。

\*/

class Receiver : public QObject

{

Q\_OBJECT

public:

explicit Receiver(QObject \*parent = nullptr) : QObject(parent) {}

public slots:

void handleResult(const QString &result)

{

qDebug() << "Result:" << result;

QCoreApplication::quit();

}

};

int main(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

Worker worker;

Receiver receiver;

QThread workerThread;

worker.moveToThread(&workerThread); //创建新线程

QObject::connect(&workerThread, &QThread::finished, &worker, &QObject::deleteLater);

//线程结束，清除线程

QObject::connect(&worker, &Worker::resultReady, &receiver, &Receiver::handleResult, Qt::QueuedConnection);

workerThread.start();

QMetaObject::invokeMethod(&worker, &Worker::process);

return a.exec();

}

#include "main.moc"

在这个示例中，Worker类表示工作线程，其中的process方法模拟一个耗时操作，并在完成后发送结果信号resultReady。Receiver类表示接收线程，其中的handleResult方法用于处理接收到的结果。

在main函数中，我们创建了一个Worker实例和一个Receiver实例，并将Worker实例移动到一个单独的线程workerThread中。然后，我们使用QObject::connect函数将Worker的resultReady信号连接到Receiver的handleResult槽函数，连接类型设置为Qt::QueuedConnection。最后，我们启动workerThread线程，并通过QMetaObject::invokeMethod调用Worker的process方法。

当Worker的process方法执行完成后，会发送结果信号resultReady，由于连接类型为Qt::QueuedConnection，这个信号会被加入到接收线程的事件队列中，然后在接收线程的事件循环中调用handleResult槽函数。在handleResult槽函数中，我们打印出接收到的结果，并调用QCoreApplication::quit方法来退出应用程序。

通过这样的方式，我们实现了跨线程的信号槽通信。工作线程中的耗时操作不会阻塞接收线程的事件循环，而是在完成后异步地通知接收线程处理结果。