QT新建线程的几种方法：

1.继承Qthread新建一个类，重写run方法，此方法除run之外，其他的都还是在主线程中运行；

2.使用moveToThread将新建线程转移到继承QObject的新类实例中；

## 方法1：

1.新建一个类WorkThread，基类为QThread。

2.重写类WorkThread的虚函数void run();，即新建一个函数protected void run()，然后对其进行定义。

3.在需要用到多线程的地方，实例WorkThread，然后调用函数WorkThread::start()后，则开启一条线程，自动运行函数run()。

4.当停止线程时，调用WorkThread::wait()函数，等待线程结束，并且回收线程资源。

新建一个类继承QThread：

class WorkThread : public QThread

{

public:

WorkThread();

protected:

void run();

};

重写类WorkThread的虚函数void run()，为线程需要执行的函数：

WorkThread::WorkThread()

{

}

void WorkThread::run()

{

while(true)

{

for(int n=0;n<10;n++)

qDebug()<<n<<n<<n<<n<<n<<n<<n<<n;

}

}

在需要使用的地方实例化此类并使用start函数：

WorkThread \*thread1 = new WorkThread();

thread1->start();

在需要停止的地方使用wait()函数，阻塞等待线程结束，并且回收线程资源：

thread1->wait();

## 方法2：

1.新建一个test类，基类为QObject，在此类中写上此线程需要调用的函数作为槽函数；

2.在使用处实例化一个test类；

3.新建一个QThread对象thread，并用connect将信号发送者设置为thread，接收者为实例化的test类；

4.使用test类的moveToThread（）方法，参数填入thread；

5.调用thread的start()方法开启线程；

6.在析构函数中退出线程并释放资源。

新建一个test类，基类为QObject，在此类中写上此线程需要调用的函数作为槽函数：

class test : public QObject

{

Q\_OBJECT

public:

explicit test(QObject \*parent = 0);

public slots:

void first();

signals:

};

在需要使用的地方新建线程和test对象，并绑定对象与槽函数，开启线程：

test \*test\_w = new test();

QThread \*thread = new QThread;

connect(thread, SIGNAL(started()), test\_w, SLOT(first()));

test\_w->moveToThread (thread);

thread->start();

terminate() 终止当前线程。线程或许不会立即被终止，依赖于线程的调度策略。一般情况下，调用该函数之后再调用QThread::wait()来确保线程结束。该线程终止后，等待该线程的其他线程将被唤醒。警告：该函数比较危险，不推荐这样做。线程可能在任何代码处终止。或许在修改数据时被终止，线程结束后自己不能去做清理工作。

wait(unsigned long time = ULONG\_MAX) 阻塞当前的进程，直到满足如下两个条件之一： 1.相关的线程完成其任务，然后如果线程已经结束，则该函数返回true，如果线程没有启动，则该函数也会返回true。 2. 经过了特定长度的时间，如果时间是ULONG\_MAX（默认值），那么wait()函数几乎不会超时。（即该函数必须从run()函数返回）如果wait函数超时，那么该函数会返回false。

quit() 告诉线程的事件循环退出并且返回0值，相当于调用QThread::exit(0)。如果线程没有事件循环，这个函数则什么也不做。

exec() 使线程进入事件循环状态，并且处于wait状态，直到调用exit()函数使其退出。退出时返回值是调用exit()函数时的输入参数。如果调用quit()函数，其退出的返回值为0.该函数一般在run()函数中调用，使线程进入事件循环处理状态。

exit(int returnCode = 0) 告知线程从事件循环状态退出，并且返回returnCode的值。一般说来，返回0表示成功退出，返回非0值表示遇到错误。调用该函数后，线程不会再进行事件处理，除非再次调用exec()函数。如果当前线程不处于执行状态，那么下次调用exec()也会直接返回

一般调用quit()函数之后可以紧接着调用wait()函数确保线程退出。