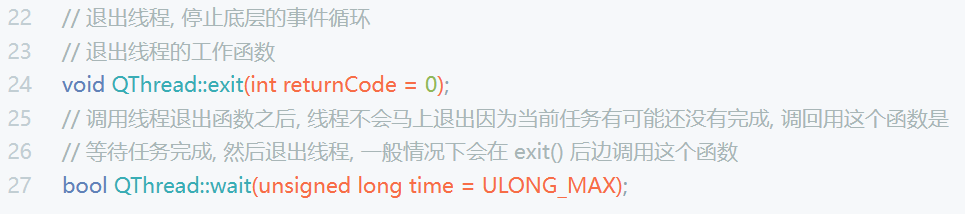
[Qt中多线程的使用 | 爱编程的大丙 (subingwen.cn)](https://subingwen.cn/qt/thread/)

1. 默认是单线程

eg：比如在一个qwidget中进行1亿个数的冒泡排序，则在排序的过程中想拖动窗口（触发鼠标事件），程序是会卡死的，因为当前线程在处理冒泡排序，无法处理标题栏的鼠标事件

，对于此类逻辑复杂的事件（冒泡排序），应该交由子线程（负责后台业务逻辑相关处理）去做，ui界面的移动等操作交给主线程（即窗口线程）去做（主线程与子线程操作不能互换）

注意：主线程与子线程的数据交互使用信号槽，只能调用start()，尽量不要用函数调用，因为

1. 对象树中对象之间的关系不一定是继承关系（注意对象树中存的是对象不是类），是从属关系
2. 

如果在调用exit()时该线程的任务只执行了一半，则调用wait()函数，等待任务继续执行至完毕才会退出，一般情况下，两者搭配着用。

1. 

调用exit()和quit()之后，都要再调用wait()等待该线程的任务执行完毕再退出。

注意：terminate()表示强制终止进程，调用该函数后不用调用wait()

1. Tongguo currentThread()判断两线程是不是同一线程
2. 基本流程：写一个子类继承QTread，该线程要执行的任务写在run()（注意是protected）中，在主线程中调用子线程的start()方法即可

[基于TCP的Qt网络通信 | 爱编程的大丙 (subingwen.cn)](https://subingwen.cn/qt/socket-tcp/)

TCP：面向连接的流式（发送端和接收端同时处理的数据可以不一样）传输协议（安全）

UDP：面向无连接的报式（同时处理的数据一样）传输协议（不安全）

注意：C++中没有提供专门用于套接字通信的类，只能用C的。但在Qt中可以使用qt封装的socket

TcpServer类：

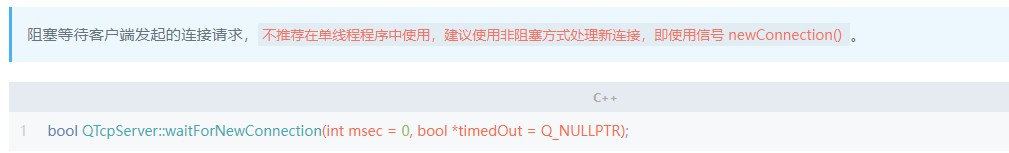
listen(…,…)：用于绑定ip地址（默认为所有ip地址）和端口（第二个参数）（必须手动设置，否则随机分配，客户端不知道服务器端口就没法链接，范围是0-65535）以及监听，相当于C中的bind()+listen()

serverAddress()：检测绑定的ip地址，否则返回0

serverPort()：检测绑定的端口，否则返回0



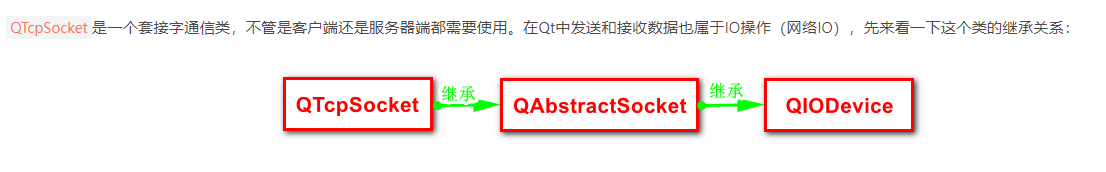
显然，QTcpServer和QTcpSocket形成对象树，即tcpserver的父对象(不是父类！)是tcpsocket



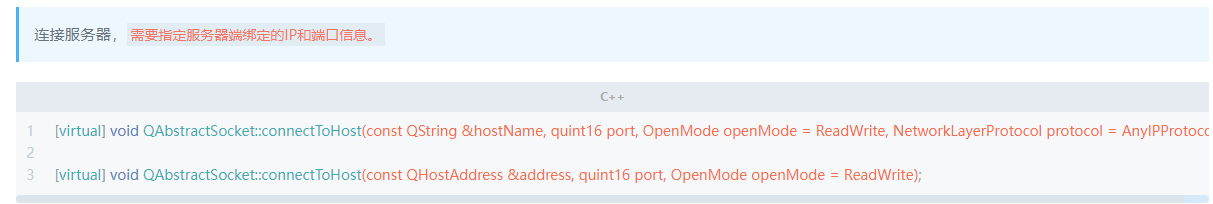
第二个参数为传出值参数，可以作为返回值来用，具体操作是先new一个bool对象，然后传入该函数，执行完毕后可以使用

但是该函数为阻塞函数，qt编程中不推荐用该函数检测客户端连接，推荐使用信号槽的方式检测，即在发送newconntction信号后用nextPendingConnection()槽函数接受连接的对象并返回可以用于接受传输信息的socket对象

TcpSocket类



注意：QFile的祖先类也是QIODevice，只不过一个是文件IO，一个是磁盘IO，调用的都是QIODevice中的数据

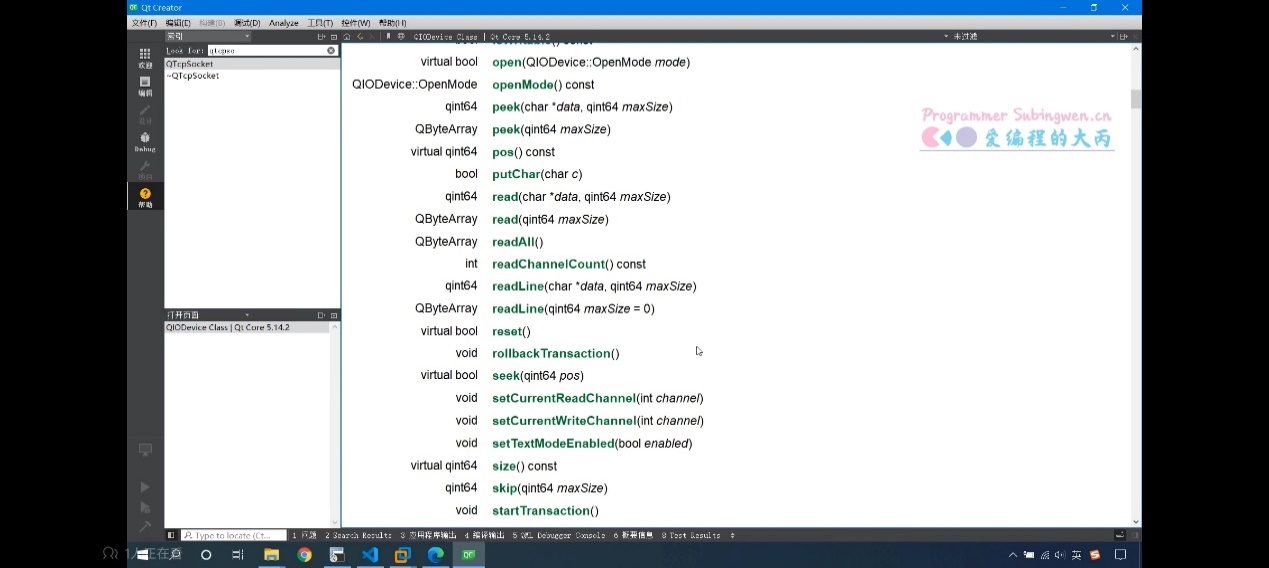


注意：1. 第一个和第二个参数分别指的是服务器端的ip和端口，第三个参数表示默认情况下的权限为读/写（控制的是虚拟地址空间高地址中tcp虚拟内存模块的读写权限）

2. 显然对于服务器端而言，ip地址可以任意绑定（listen），但对于客户端而言，ip地址仅能绑定其中一个去连接唯一的服务器

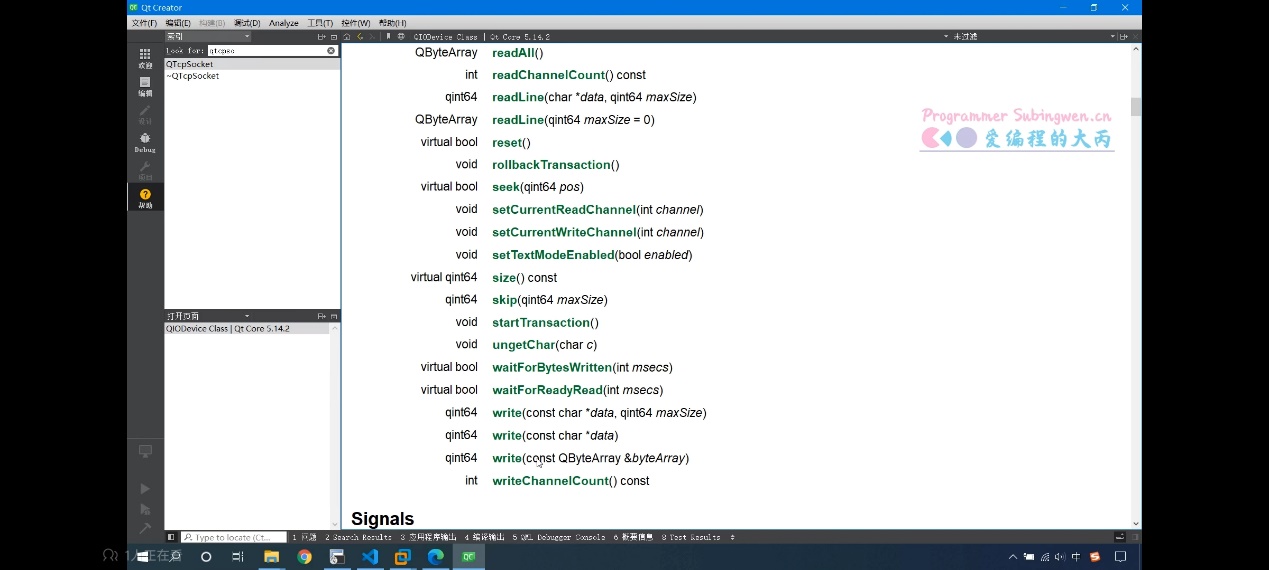
3. 在调用connecttoHost成功后发送connected()信号。显然，这两个函数都只能由客户端调用

文件读操作（位于QIODevice中）



可在F1文档中查看，其中返回值为qint64为C风格的方法，一般用返回值为QByteArray的方法

文件写操作（同上）



一个是c风格的，一个是qt风格的，一般用后者

总结：

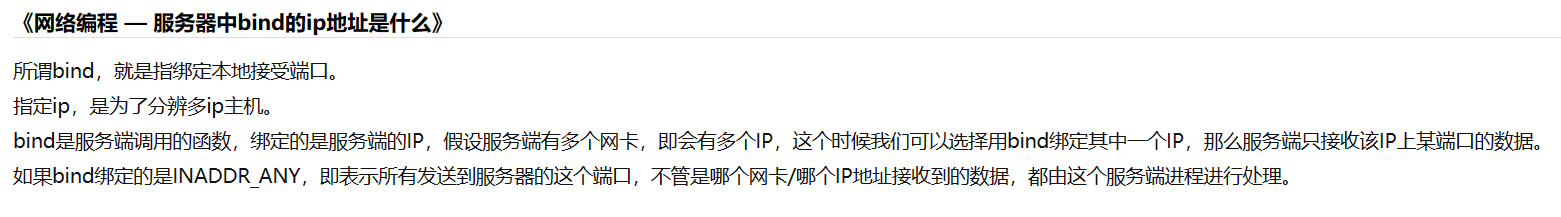
在使用TCP网络编程的时候，分为Client和Server。首先要注意到是要在两者项目中加入network模块，且与平常写的项目不同的是，TCP的对象在构造函数中就需要初始化；其次需要注意的是：

对于server而言，需要用到network中的QTcpServer以及QTcpSocket两个类；其中QTcpServer用于listen(ip(枚举值), port)，以及检测有无新的客户端socket连接进来，如果有的话则发送newconnection信号，并可以通过nextpendingconnection函数返回连接的socket。其次QTcpSocket的作用是用于连接建立之后的网络通信，即在用nextpendingconnection返回连接该Server的Client对象之后Server和Client才能进行通信；两者通信非常简单：在readyRead的时候read()，在需要发送信息的时候write()；

对于Client而言，只需要用到QTcpSocket类，不过与Server不同的是，在Client中的QTcpSocket不仅需要进行收发信息，还要负责进行与Server的连接，即通过connecttoHost(ip, port)，注意连接成功后由Client发送connected信号。

注意在两者通信期间，我们用QTcpSocket操作的并不是网络中的数据，而是虚拟地址空间高地址的Tcp模块中的数据，这个模块会暂存我们准备收发的数据，实现Tcp三次握手与四次挥手，并负责数据的网络传输。

对于QTcpServer::listen(ip,port)和QTcpScoket::connecttoHost(ip,port)而言，listen的ip可以为任意个，而后者的ip只能为一个，可以借助下图理解：



关于connect原理，详情见Lesson1/TcpServer代码中的注释

关于写ui时label,lineedit,textedit的区别：

QLabel一般用来显示提示性的文本或者说明，用户不可以直接修改内容。QLineEdit用来收集用户输入（单行文本，不能换行）。而QTextEdit用来收集用户输入的一大段文字。

TCP的多线程通信：（线程的运作方式有两种，此处都用了，其中主线程与子线程之间的通信使用信号与槽）

Client：（工作类+子线程）

主线程：负责ui界面操作，这些操作主要与工作类和thread有关①点击连接按钮调用子线程的start()并通知工作类对象启动socket连接服务器。②点击选择文件打开dialog选择文件。③点击发送文件调用工作类对象的socket发送文件④检测到子线程连接成功/断开信号后显示连接成功/断开消息盒子并在连接断开时退出子线程并手动释放子线程和工作类

选择文件：调用QFileDialog中的getOpenFileName显示选择文件的Dialog界面。

创建子线程：点击连接按钮之后new出子线程（可以加入对象树也可以手动释放）和工作类（切记不能指定父对象！但是指定父对象为nullptr可以），并将工作类对象移动到子对象中movetothread()，

显示消息盒子：

子线程的工作类：负责后台数据操作，这些操作都和socket有关①检测到主界面连接信号后负责连接服务器。②检测到主界面发送信号后负责发送文件。③检测到服务器断开信号后负责关闭socket通知主线程

1. 点击连接按钮后调用子线程的start()（调用后子线程会执行里面的run()），然后再调用工作类对象的startConnect()（用于socket连接服务器的ip和port）
2. 创建QFile对象（用于对文件读写）并指定文件路径和打开方式，并调用QFileInfo类获取文件的size发送给服务器用于告诉服务器文件的大小，方便判断。并通过readline方法一次读取一行文件，之后干两件事，第一件是用socket发送文件，第二件是发送当前进度至主线程用于显示进度条进度。
3. 检测到服务器连接与断开信号后分别发送连接成功与连接断开信号通知主线程。

Server：（子线程，通过调用start启动run）

主线程：

主要负责ui界面操作，这些操作主要和TcpServer和thread有关；①点击监听之后在主线程new出tcpserver指明ip和port启动监听②tcpserver检测到newconnection信号后调用nextpendingconnection返回连接client的socket并在new出thread时将socket传入其构造函数，启动thread（启动时自动执行run）③得到子线程文件传输完毕的信号后关闭子线程并显示接收成功消息盒子。

子线程：

主要负责后台数据处理，这些操作和Socket有关；①run()的重载记得添加后置申明符override。②new出QFile，指明路径和读写方式，当检测到readyRead时转到槽函数，并且设置exec进入事件循环③在槽函数中读出文件大小以及内容并将内容写入file中，当文件接受完时关闭socket和文件并发送结束信号给主线程