一、设计要求

实现一款运行在微软操作系统中的 FTP 客户端软件。该软件具有友好的图形化用户界面，实现基本的 FTP 功能，如连接 FTP 服务器，用户登录，浏览、更改、新建和删除文件和文件目录，能进行文件的上传和下载，设置文件传输方式为 ASCII 或二进制，传输模式为 Port 模式或被动模式，使得用户可以根据自己的需求，服务器端系统，防火墙设置和实际网络状况进行适当的参数设置。

为了能兼顾充分利用 WinSock 接口的功能，并且设计出美观、友好的用户交互界面，本软件选择在 VS2017 中搭建 Qt5.9.0 环境，利用 C++进行开发。借助于面向对象编程的思想，我首先设计了一个 FSocket 类对 WinSock 接口提供的一系列实现套接字连接建立和数据传输的函数进行封装，具有套接字建立准备、连接到服务器、套接字状态检测、监听、多种数据和文件发送和接收方法、出错报警的功能，为下一步管理 FTP 客户端与服务器的通信、交互打好了基础。ListenThread 类重写了 QThread 多线程类的 run()方法，主要服务于主动模式下客户端的数据端口的监听，因为这里采用阻塞 accept()接收数据连接，会阻塞主线程，导致无法向服务端发送客户端 IP 地址和其当前正在监听的端口地址。FFtp 这个类就是负责统筹管理所有 FTP 客户端用户向服务器发送命令，并接收答复或出错信息的类。功能类似于用户通过 Windows 命令提示符使用命令行进行的 FTP 操作，但用户使用本软件只需要与图形化界面进行交互，因此 FFtp 类对大部分常用的 FTP 命令进行了进一步的封装，以便第三个类，窗体 MainWindow 类进行调用。用户通过输入用户信息、在文件资源管理器中选择文件、点击按钮、触发动作，完成请求信息的发送，并通过窗体中元件的使能状态，下方的状态栏，标签文字或出错警示框获知 FTP 服务器端的响应，当前与 FTP 服务器的连接状况，以及用户登录状态。

为了贴近用户在本地文件资源管理器中浏览和操作目录的习惯，本软件实现了在用户登陆成功后显示 FTP 服务器的树状文件列表。TreeItem 和 TreeModel 两个类负责实现目录的显示和更改功能，配合在一起完成了对树形界面的显示模式设置，直观地展示出文件目录的层次结构，文件信息，包括文件名，大小和修改日期。用户可以直接鼠标点选目录、文件，在目录上方的文字栏中获取该文件所在的路径，并进行增删改操作。

二、开发环境与工具

1. 操作系统：Windows 10 64 位家庭中文版；

2. C++开发环境：Microsoft Visual Studio Community 2017，Qt VS tools 插件；

3. 编译器：MSVC2017_64；

4. 图形界面设计辅助工具：Qt Designer 5.9.0；

5. 建立 FTP 站点进行测试：Windows Internet Information Server 提供的 FTP 服务。

三、设计原理

1. TCP 协议

在 RFC 765 中，TCP 协议首次被用作 FTP 协议的底层传输协议，延续至今，为 FTP 提供了面向连接的、可靠的端到端传输服务，可以进行全双工的数据传输。

1.1. 建立 FTP 客户进程和 FTP 服务进程之间的 TCP 连接

在使用 FTP 提供的服务，或者是要传输数据、文件之前，都要先建立基于 TCP 的可靠连接，需要提供 TCP 传输地址，包括目标 IP 地址和端口号。可靠的连接建立需要经历“三次握手”，通过 wireshark 抓包分析工具，抓取与另一个主机的 FTP 服务器建立连接的过程中产生的数据包进行观察，可以发现连续有三个包首先建立起 TCP 连接，接下来才会发送含有 FTP 指令序列的数据包。类似的，在 FTP 数据连接或控制连接断开时，也会经历“四次挥手”，断开 TCP 连接。

1.2. 错误恢复和重启

在传输中出现的丢失字节或者数据包混乱等错误，将交由 TCP 控制。TCP 协议的内在特性使得 FTP 数据传输可以处理并挽回一些错误，如 TCP 的超时重传机制、快速重传、流量控制等。

2. FTP 协议

文件传输协议的设计初衷是为了方便程序与数据文件的共享，用户可以（通过程序）使用远程计算机，并且不用担心不同主机之间文件系统的差异。

2.1. 概要

FTP 遵从 C/S 的工作模式，用户可在本地主机上使用命令行或专门的客户端图形界面访问服务器端进行交互。在 FTP 服务器端使用访问控制（access

controls），限定了特定用户对系统的使用和文件的访问权级，避免了未授权或意外的文件访问。

FTP 协议在工作时会建立两类连接，这个过程虽然更为复杂，但也可以提供数据传输的效率。一种是进行 USER-PI 和 SERVER-PI（用户和服务器协议解析器）之间发送指令和接收回复的连接，贯穿始终，从用户请求会话到结束访问服务器为止，称为控制连接。FTP 协议拥有一个指令集，包含了从用户 FTP 进程（user-FTP process）发送到服务器 FTP 进程（server-FTP process）可以完成的全部控制信息。这些命令以 7 比特 ASCII 码在控制连接上传送，命令内容可读，由 4 个大写字母 ASCII 字符开头，有些命令还要继续给出可选参数。每行以回车换行符<CRLF>作为行结束符序列（End-of-Line）。用户发出的命令首先到达 USER-PI，一一对应地转换为用户 FTP 进程发送给 SERVER-PI 的命令。SERVER-PI 会在指定端口监听，响应控制连接请求，并控制服务器 DTP（data transfer process）。对每条命令，都会有一个发回客户端的回答，以三位完成码或错误码开头，方便程序解析，后跟一段用户可读的文本字符串。

另一种是数据连接，工作在全双工模式下，通常每一次服务器传送目录信息、传输文件都要建立一条新的数据连接，传输完毕即自动断开，但传输的内容可以是不完整的文件，也可以是多个文件。FTP 使用 DTP 建立和管理数据连接，可以是主动也可以是被动的。处于被动一方的数据传输进程会在其数据端口（data port）上一直监听，时刻准备打开数据连接。

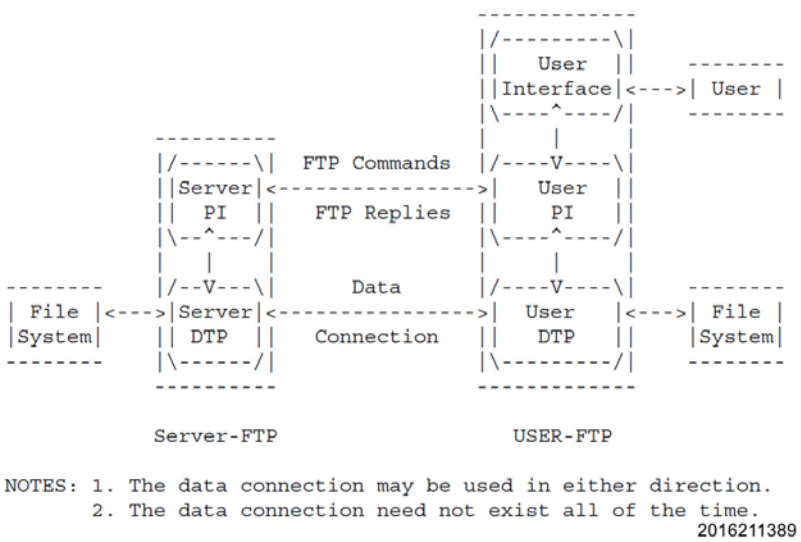

图 1FTP 协议模型

2.2.　数据传输

在使用 FTP 服务之前，本地主机主动发起连接请求，使用一个大于等于 1024 的本地端口号作为源端口号，服务器端的 21 号作为目的端口号。经历 TCP 连接管理，建立了一个可靠的连接。当客户通过控制连接准备向服务器发送传输文件、数据或程序的命令时，有两种传输方式：Port（主动）模式，和 Passive（被动）模式。

主被动模式是站在服务器角度定义的。如果发起数据连接的一方是服务器而监听的一方是客户端，则为 Port 模式；反之为被动模式。Port 模式工作原理如图 2，客户端的数据端口为任意选择，因此要告知服务器，端口号满足 L = h1 * 256 + h2。服务器端知晓后，便由其制定的 20 端口主动发起到客户端 L 端口的数据连接。

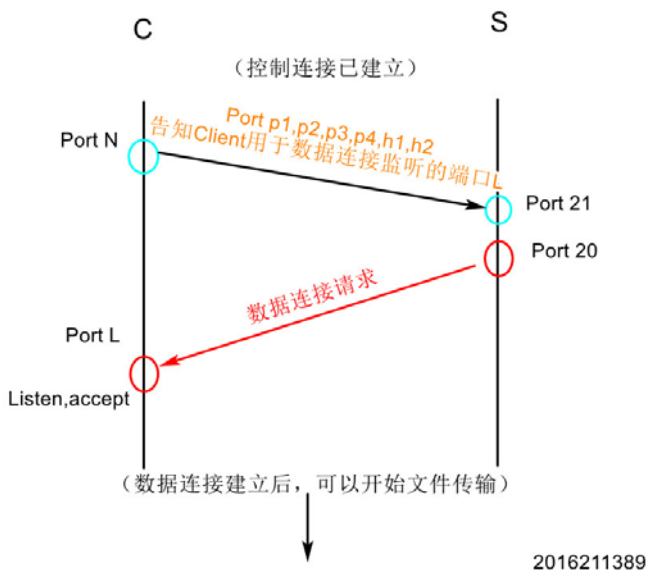

图 2Port 模式工作原理

被动模式如图 3，客户端为了知道数据连接应当向服务器哪一个端口发出请求，要主动发出 PASV 向服务器问询。待服务器端通过控制连接发回响应后，客户端就可以从自己任选的客户端数据口向服务器给出的正在监听的端口发出数据连接请求。

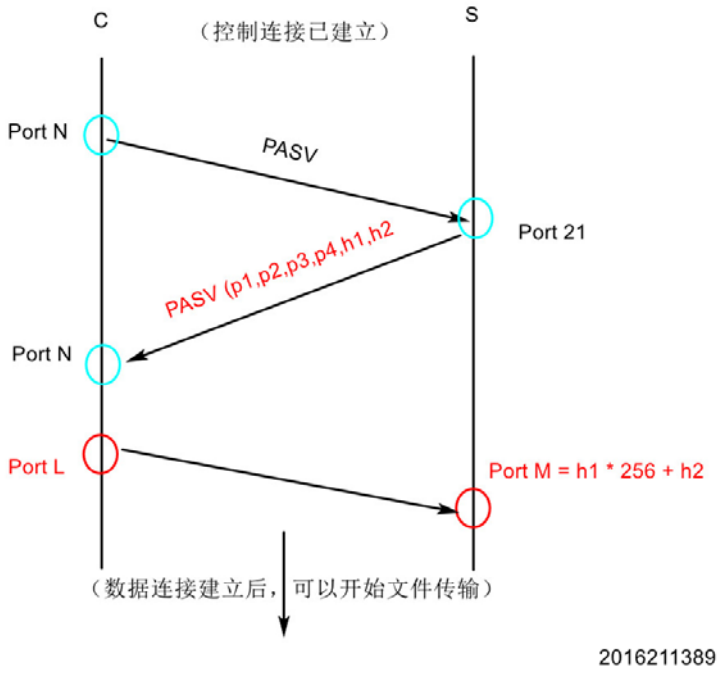

图 3Passive **模式工作原理**

2.3. 传输模式

FTP 提供不同的传输模式，但要保证在传输同一个文件的过程中，必须固定参数。

2.3.1. ASCII 方式

FTP 会自动调整 ASCII 文本，将它解析为传输对端主机存储文本文件的格式。例如，从微软主机发送到远程的 Linux 主机，使用该模式会将回车换行符转变为\n。

2.3.2. Binary 方式

该方式保证了在传输任何非文本文件时，FTP 将不会对其中任何字节进行处理变动。

3. Telnet 协议

Telnet 协议规定了远程登录的标准和方式，应用于 FTP 协议中的主要功能是建立 FTP 控制连接，数据连接。同时，FTP 协议也使用了虚拟终端（NVT）的功能，确保任何字符表示或命令解释的一致性。

4. 基于 WinSock 的网络编程

WinSock API 是在 Windows 下支持多种网络协议的网络编程接口和标准。现在主要使用 WinSock2.2 版本，但能向下兼容旧版本。

定义一个套接字相当于指定网络传输的一个端点，通信链的句柄，通过套接字我们可以发送或应答网络请求。通信连接到建立必须要有成对的套接字，一个是客户端套接字，另一个是服务器端套接字。

有一类套接字为流式套接字，它提供一种可靠的、面向连接的双向数据传输服务，被传输的数据被看作无记录边界的字节流。客户端套接字建立连接的过程包括：检查协议栈的安装情况，如版本和动态链接库；创建一个字节流套接字；向服务器端发出请求。服务器端套接字建立连接的过程包括：检查协议栈的安装情况；创建一个字节流套接字；将服务器地址绑定在该套接字上；监听连接请求，时刻做好建立连接的准备。

四、系统功能描述及软件模块划分

1. 系统功能描述

FClient 是一款运行在微软操作系统中的 FTP 客户端软件。该软件具有友好的图形化用户界面，实现了多种功能：连接到 FTP 服务器，用户身份认证，显示文件目录信息，可以方便地进行文件目录的新建、更改、删除，文件的上传和下载。除此之外，用户还可以根据自己的需求，服务器端操作系统，或本地电脑的防火墙设置和网络状况进行适当的传输参数设置，如选择文件的传输方式为 ASCII 传输或二进制传输，数据连接建立的方式为主动模式或被动模式。

本软件实现了一个 QMainWindow 类作为交互主窗体，被三个 QGroupBox 对象，位于窗体上方的 menuBar 对象和下方的 satusBar 对象划分为五个功能区域。同一区域内的部件使用水平或垂直布局对仗工整，紧凑布局；各个组合而成的区域之间采用栅格布局，以便较好地适应主窗口的不同大小，并保持各组件间相对位置基本不发生变化。

用户登录 QGroupBox 负责获取 FTP 服务器 IP 地址端口（默认 21 号），用户验证信息，并通过点击按钮触发登录。通过窗体使能区域（可使用、可点击）的变化，并结合下方 statusBar 可以判断登录状态。用户输入的 FTP 服务器 IP 地址框被正则表达式限定为只可以输入 IPv4 点分十进制格式，因此要开启正确的英文输入法，只输入数字和点分十进制规定位置上的小数点。考虑到一些 FTP 服务器支持匿名 FTP，此时用户名固定为 Anonymous，因此用户名一栏的下拉菜单中预先提供该匿名名称。密码框固定为英文输入法，以“•”一一代替输入字

符。若用户选择以匿名方式登入，则无须再填写密码框，便可以直接点击登录按钮。当且仅当在这种情况下，密码输入框可为空，否则必须合法填充三行文本输入框，违规则会有警示框跳出，不会发起登录请求。

文件目录 QGroupBox 主体实现了一个 QTreeView 对象，自定义的 TreeModel 类设置了该 TreeView 的模型，共三栏，将于用户验证信息正确后显示。该区域可以显示树状的服务器文件目录，包括文件名称、大小、最近修改日期，根据各栏字符长度适应性地调整列宽，也可以手动拖拽进行调整。目录区域可以鼠标点选，如果是文件夹，使用双击文件名前方空心箭头打开子目录，若该文件夹为空，则弹出空目录一行。树形视图上方为两栏文本编辑行和三个按钮。第一行显示为“Index of /”，用于实时根据鼠标点选项，显示该文件或目录在 FTP 服务器中的目录。右侧两个 QToolButton 工具按钮，显示为一“+”和一“-”的图标，用于对当前工作目录（文件）进行新建目录，删除文件或选定目录及其下所有文件。第二行是 Change to 一栏，用于给出需要更改的文件或目录（即第一行 Index 中当前显示的）将要更改到的新路径、新名称，change 按钮用于触发该动作。

文件传输 QGroupBox 用于管理显示文件的上传和下载。用户通过一个工具按钮对本地文件进行选择，文本框显示传输文件基本信息的功能。具体的传输参数由菜单栏细化实现。

上方的 menuBar 中，用户可以选择登出，关闭软件；可进行传输参数设置，包括文件数据的传输模式，数据连接建立的方式等。灰色的不可点击区域说明了当前软件所正在使用的参数设置，因此不能重复设定。

2. 系统模块描述

2.1. 控制命令的发送和响应的接收解析

2.1.1. 控制命令发送

通过控制连接建立得到的套接字，进行命令发送，由 FFtp 的方法 sendRequest(const QByteArray &command)得到。该方法非常重要，所有控制命令都由它发出，此方法中调用了 FSocket 类的 write(const QByteArray &byteArray)，其中具体完成发送操作的是 WinSocket 的发送函数 send()，在已经建立的套接字上传送数据。

```
void FFtp::sendRequest(const QByteArray &command)
{
    tcpSocket->write(command + "\r\n");
}
```
2016211389

图 4 发送 FTP 命令的实现

2.1.2. 响应接收及解析

一般的 sendRequest 调用后都要调用 recvReply 进行相应地读取，通过 FSocket 的 readAll 方法封装的 recv()完成，在连接已建立的套接字上接收数据。读回的 QByteArray 的前三位尤为重要，需要将此三位的完成码（有时是错误码）提取，并 emit 一个信号 SIGNAL responseCodeReceived，连接触发 SLOT parseCode，这将对应到不同的反馈状态中有一些序列化的指令因此而被调用。有一些则是错误码，可由此分析并纠正。

2.1.3. 错误提示

在 FSocket 中实现的与 socket 连接、数据传输的所有方法都有可能因某种网络错误或其它原因而没能正常执行，因此单独设计一个 warningMsg 槽函数，当错误发生时，emit errorCode，触发该槽函数，及时通过弹出警告窗告知用户哪一个通信阶段出错，以及错误代码是多少，具体的含义可以上微软的 WinSock API 官方文档中查阅。

2.2. 登录模块

2.2.1. 控制连接建立

在 FFtp 类的连接到服务器方法 open()中，调用 FSocket 类中的 connectToHost(const QString hostAddress, const int port)进行套接字准备，和连接建立。该 hostAddress 是从 IP 输入框中读取的。如果改地址不合法，会跳出警告框提示检查 IP 地址输入。定义 SocketState 枚举类型，记录 socket 当前状态，避免发生重复的错误。

```
if(tcpSocket->state() == FSocket::UnconnectedState)
{
    if(!tcpSocket->connectToHost(m_ip, port))
    {
        recvReply();
        user();
    }
}
```
2016211389

图 5 请求建立控制连接

2.2.2. 用户登录

在调用 open()成功建立连接后，使用 FFtp 类中的 user()、userInfo()

从图形界面获取用户数据，进而发出用户登录请求，如图 6。若连接已建立，但用户名或密码不正确，则需要重新检查用户名密码的输入，并向服务器重新发送用户验证信息。此时 ftp 已经处于 ConnectedState 中，连接无需重复建立，这样会导致错误，除非先断开此连接。

```
if (userName == "Anonymous")
    password = "";
ftp->userInfo(userName, password);
if(ftp->status() == FSocket::UnconnectedState)
    ftp->open(ip, port);
else
    ftp->user();
```


图 6 控制连接建立，发送用户信息

若用户名正确，且进一步需要密码，则客户端会收到完成码 331，此时发送 PASS 命令，如图 7。

```
case 331: // FTP 用户名正确，需要口令 --> 成功，则转230
    sendRequest("PASS " + m_password.toUtf8());
    recvReply();
    break;
```


图 7 登录需要密码

2.3. 目录操作模块

2.3.1. 目录获取与显示

在用户登录后，将会自动显示该 FTP 服务器端的根目录。通过调用 FFtp 类中专门设计的 fileResourceManager 方法，递归调用“LIST”FTP 指令，获取当前目录中的所有文件信息。处于不同目录中的文件的层次结构由该方法在每一行（一个文件或文件夹）文本前添加代表文件 hierarchy 的数字，从根目录为 0 开始，层层递增。该方法最终将获得一个包含此 FTP 服务器中所有路径中的所有文件信息，以及处理后增加的文件结构前缀记录的 QString 列表，每一行文件项按序占有列表中的一个元素。

```
void FFtp::fileResourceManager(int &position, int hierarchy, QStringList &CompleteList, QString pathName)
{
    QStringList rawList = ls(pathName).remove("\"").split("\r\n");
    int lines = qMax(rawList.count() - 1, 1);
    for (int i = 0; i < lines; i++)
    {
        position += i;
        CompleteList.insert(position, QString::number(hierarchy) + rawList[i]);
        if (rawList[i].indexOf(" <DIR>") != -1)
        {
            fileResourceManager(++position, hierarchy + 1, CompleteList, pathName + "/" + rawList[i].mid(39));
        }
    }
}
2016211389
```

图 8FFtp 类的 fileResourceManager 方法实现获取文件结构

2.3.2. 获取用户当前所处的 FTP 服务器目录

由于用户可以通过点击 TreeView 中的各行文件浏览目录，我将用户最近一次点击的文件或文件夹的目录作为当前所在目录，因此，需要设计一个以鼠标选择改变为信号源所触发的一个槽函数，即 MainWindow 类中的 updateSelection 函数。其中调用 getPath 函数获得当前项的目录，每一级由“/”划分，若选定的是一个文件夹的目录，注意文件夹要以“/”结尾。

```
void MainWindow::updateSelection()
{
    bool hasSelection = ui->treeView->selectionModel()->currentIndex().isValid();

    if (hasSelection)
    {
        ui->treeView->closePersistentEditor(ui->treeView->selectionModel()->currentIndex());
        bool isFolder;
        QString filepath = getPath(ui->treeView->selectionModel()->currentIndex(), isFolder);
        if (isFolder)
            filepath += "/";
        ui->serverWorkingDirectory->setText(filepath);
    }
}
2016211389
```

图 9 鼠标选择改变触发的槽函数

2.3.3. 目录（文件）的增删改

在 2.3.2 的实现基础上，可以方便地获取用户当前期望操作的特定目录，再调用 FFtp 类中的 mkdir()、folderDele(),并辅助设计了 nameNewFolder 类用以处理新建文件夹重名的情况。folderDele()其实是扩充、综合了 FTP 本身提供的两项功能，原本仅是发送“RMD <sp> file-path”删除一个空目录，但现在当用户选择的是一个文件时，删除该文件，FTP 命令为“DELE <sp> file-name”，当选择的是任一个目录时，删除该目录下所有文件。

2.4. 文件传输模块

2.4.1. 文件传输显示

在用户登录的状态下可上传下载文件。通过工具按钮打开本地文件资源管理器选择要上传的文件，点击 Upload 按钮触发打开文件，设置该文件只读，读出为 QByteArray 类型进行传输。为了显示文件上传进度，尤其是百兆以上的文件，在每读取 2 的 26 次方字节，就打印一次正在传输的进度。同时，在窗体上还显示该传输文件的基本信息，传输完毕仍然保留，直到下次有新文件传输时清空（因为即使很大的文件速度也比较快，这样设计更便于观察）。

2.4.2. 文件传输实现

在 Upload 按钮触发的槽函数中调用 FFtp 类中的 put 函数，参数为文件信息和文件的字节序列。选用当前用户设定的文件传输模式进行传输（主动或被动传输）。文件传输调用 FSocket 中的 write 方法，使用 winsocket 的 send()函数在已经建立的数据连接中进行传输。

2.5. 菜单选择模块

每一个触发动作，都对应于一个槽函数，通过 FTP 控制连接发送请求。如果出错，也会由 FSocket 类中的报错信号触发弹出警告框。

五、设计步骤

1. class MainWindow

本软件的基本目标是设计一款用户可以通过图形界面与 FTP 服务器进行交互的软件，因此设计一个继承自 QMainWindow 的窗体类 MainWindow。FTP 几个常用且重要组成部分是访问控制、文件数据传输、文件目录浏览，因此先大致在主窗口上用 QGroupBox 框出这几个部分。首先实现建立连接和用户登录，因为这是可以访问 FTP 服务器的必要条件。类似于日常经常使用的登录窗口，先设计出了一个非常常见的用户登录界面，有用户名和密码框，不需要注册按钮，但是必须要输入 FTP 服务器的 IP 地址，以及可以选择地更改 FTP 服务器默认控制连接端口（21）的端口号输入框。

2. class FSocket

接下来考虑建立连接和用户登录的具体实现。因为使用的是 Windows 网络编

程标准 API WinSock，所以它提供的是一个普适的套接字建立方法，要建立不同的网络数据连接，选择不同的参数即可。在已经建立连接的套接字之上，FTP 协议提供有特色的远程文件传输服务。因此，可以把建立套接字的过程和进行 FTP 操作的实现分开。如下图所示，在 WinSock 提供的接口之上，我封装了一层 FScoket 类。FSocket 类作为 WinSock API 和我设计的管理 FTP 功能的 FFtp 类之间“交流”的“中间人”，负责 FTP 客户端主机与 FTP 服务器建立控制连接、发送 FTP 命令、接收服务器响应、可以以两种不同的方式建立专门传输文件数据的连接、传输数据、报告错误。

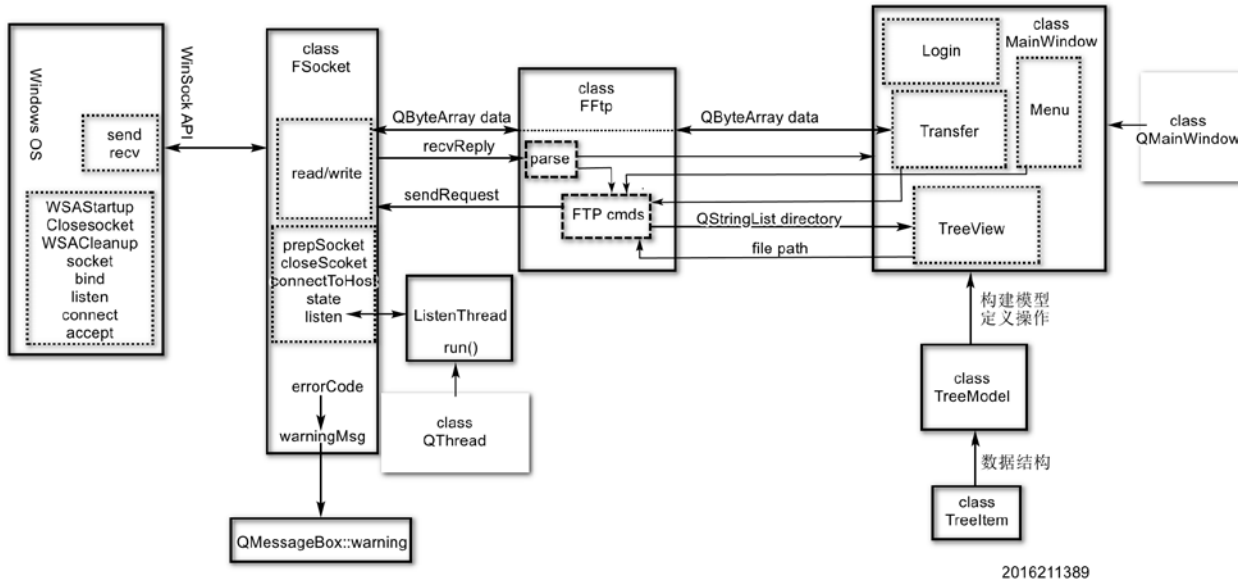

图 10 类设计图

3. class FFtp

设计 FFtp 类的目的一方面是接收用户在图形界面中触发的响应，读取用户输入的数据，将这些命令和参数以标准格式首先传递给 USER-PI，形成 FTP 指令通过控制连接传给 SERVER-PI，当有响应从服务器发回时及时读取并解析，一些完成码要传回用户界面进行交互，如登入成功，若是错误码也要及时反馈，如用户名密码错误，目录不存在，无访问控制权限，或者是自动断连等等。另一方面是管理 DTP，受 PI 解析后的命令控制，完成相应的操作，如以何种结构传输文件，如何建立数据连接等。因此无论用户界面要进行何种有关 FTP 的操作，只要组合使用管理 FTP 客户端进程和服务器进程交互的 FFtp 类提供的方法，基本都能满足需求。

4. class TreeItem 和 class TreeModel

受到平常浏览文件目录的习惯影响，初步的想法是要将 FTP 服务器上的文件同样以目录树的形式展示出来。查阅 Qt 官方文档，发现 QTreeView 作为一个展示树形界面的强大 View，其显示方式必须要设置一个模型 model，继承自 QAbstractItemModel。因此要实现一个记录目录层次结构的数据结构类，和一个将这种迭代的数据结构良好地布局于 QTreeView 中的模型类。

六、关键问题及其解决方法

1. 主动模式的数据连接建立

开始时这个问题困扰了我。首先是因为对数据连接建立的时机和关键作用没有很好的理解。建立数据连接的目的就是为了传输数据，因此在客户端要发起文件或数据传输之前，就应当建立好数据连接。其次是对于 PORT 命令的理解。工作中在主动模式下的服务器的 20 号端口会主动向客户端的数据端口发起连接请求，因此需要知道是哪一个客户端的端口正在监听数据连接请求。这正是 PORT 命令的意义所在，即告诉服务器，它要发出的数据连接请求的目的端口号应当是多少。因此，要进行数据传输时，客户端要先选择一个空闲的端口作为数据连接监听端口，发送 PORT 命令告知服务器这一端口号，当受到 PORT 命令成功的回复时，就表明服务器要向这个客户端的端口发起请求了。图 2 直观地说明了 PORT 的原理。

还有一个问题是在监听中使用的 accept()函数默认为阻塞的，因此建立一个 ListenThread 类，重写其 run()方法，让会产生阻塞的 accept()在这个新线程中持续等待接受 connect()发出的连接请求。

以 LIST 命令为例，它返回的目录信息是需要通过数据连接传输的。如果当前用户设置的传输模式为主动模式，则调用 FSocket 的 listen 函数在客户端选定的一个端口上实例化一个新的线程进行监听。

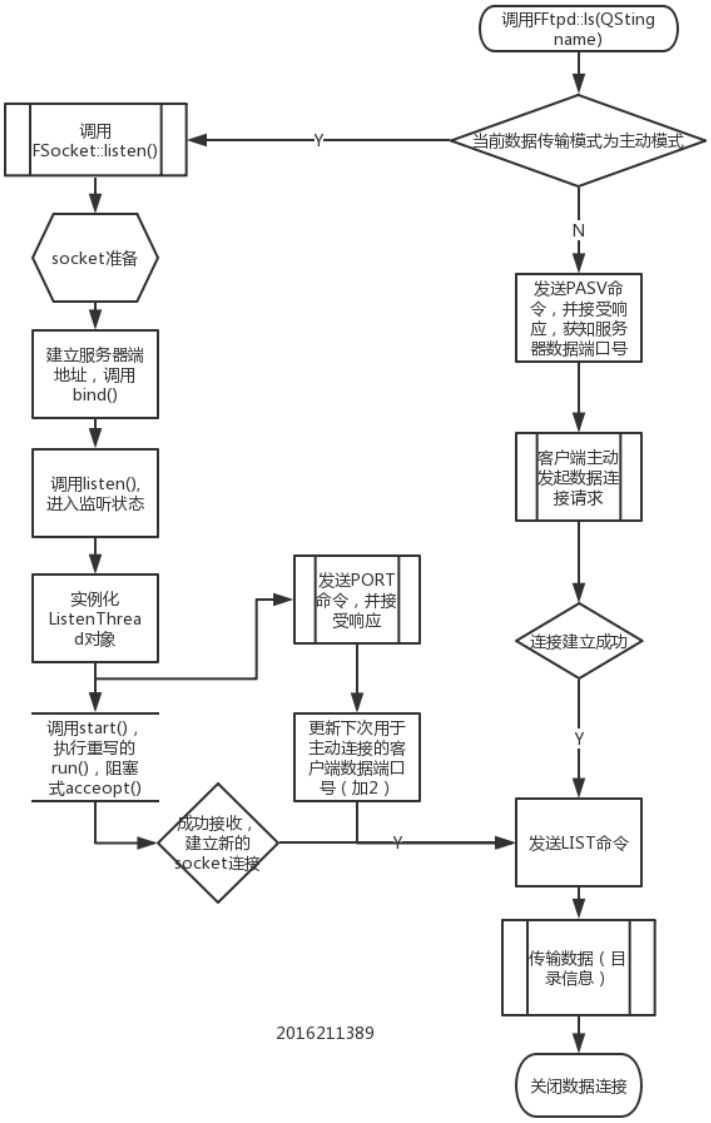

图 11 以 ls()为例说明两种传输模式

接下来产生了一个问题：程序每隔一段时间才能成功运行一次，但首次运行一般都是成功的。通过命令窗口中的报错信息，我发现是 bind()总是出错，WSAGetLastError 返回的错误码是 10048，应该是因为 bind()绑定的端口仍被占用，再次绑定同一个端口就会发生错误。虽然在获得数据后就调用 closesocket“关闭”了数据连接，但是仅仅 closesocket 执行完毕，端口不一定会立刻释放，会有一段等待时间，处于 TIME_WAIT 状态。知道了这些，我密集地跑程序出现这样有规律的错误似乎找到了原因。针对这个问题我修改了我原本直接指定特定的客户端数据端口号，而改为每一次准备建立主动模式的数据连接时端口号自动加 2，这样的改变是借鉴了被动模式下每一次服务器返回的端口值的大致的变化规

律，我观察到每一次返回到端口号，也就是 6 个参数中的最后一个，都增加了 2。

2. 控制命令的响应接收

控制命令的响应接收由 FFtp 类中的 recvReply()完成，其内部其实是调用了 WinSock 接口的 recv 接收函数，接收作为控制连接的套接字上到达到的字符。针对每一个 FTP 指令，都有一个或一系列对应的响应码。比如在数据传输过程的开始和结束分别有一段响应字符串。由于 recvReply()读响应不是同步的，也即是有一定的延迟，因此，一次性接收到的响应可能是两条在一起，如在传送一个小文件时，“125 Data connection already open; Transfer starting.”和“226 Transfer complete.”可能会一起读出。但是我在开始实现时，对每一个会产生响应的 FTP 命令请求，都在其后调用了一次 recvReply()。当很不幸出现上述情形时，程序会陷入 recv()函数。在测试中，开始我没有重视这个问题，导致程序窗口出现未响应时分析不出出错原因。自从意识到这个问题后，我就不再在 sendRequest 函数中无差异地调用 recvReply 了，而是将其放置于较为可靠的位置上。

3. 文件传输进程的可视化

Qt 已经提供了一个非常方便的进度条类，主要需要设置其最大值，和及时更新当前进度值。不使用 Qt 专门提供的网络连接管理类，必须自定义一个可以同步触发的信号，将文件当前进度传递给控制进度条显示的槽函数。这类信号直接使用 emit 似乎无法完成。因此只能在命令行实现了文件传输的进度显示，提示用户此时文件正在传输。

4. 文件目录的浏览和修改

可能还是由于对 Qt 的不熟悉，导致在实现这一步时花费了大量时间。开始自己实现的界面显示出来总看起来是灰色的，每一个文件名称无法直接选中，似乎是使选中模式关闭了。我调用相关函数直接进行设置似乎也没有任何改变。我也考虑了是不是在设计界面时该 QTreeView 被遮挡，但经过检查，没有出现这样的情况。在这里卡顿了很长时间，多次修改 TreeModel 类，也尝试了其他窗口显示方式，比如在 Split 窗口中同时显示两个独立界面，等等，都是无法“激活”树状界面。最后一次，我直接在 Qt Creator 中打开这个.pro,进行了项目清理，重新 qmake，重新构建项目，竟然成功了。

完成了这一关键一步，接下来就是跟踪用户点击的目录路径了，这是一个十分重要的信息，要随时获取，决定了用户进行目录操作的目标对象。定义连接：

connect(ui->treeView->selectionModel(), &QItemSelectionModel::selectionChanged, this, &MainWindow::updateSelection);

在槽函数 updateSelection 中调用 getPath 方法，递归地获取 treeview 中选定 index 的完成路径名。

```
QString MainWindow::getPath(QModelIndex index, bool &isFolder)      // isFolder判断当前文件是否是一个文件夹
{
    QAbstractItemModel *model = ui->treeView->model();
    QString current = model->data(index).toString();
    QString parentname = model->data(model->parent(index)).toString();  //父目录名

    if (parentname != "")
        parentname = getPath(model->parent(index), isFolder) + "/"; // 当不处于根目录下时

    isFolder = (model->hasChildren(index)) ? true : false;  // 只关心最后一次的赋值，这才是我们的目标文件（夹）

    return parentname + current;    // 返回当前index的完整路径
}
```

图 12 获取完整路径名的递归方法

七、设计结果

1. 运行界面

沿逆时针方向依次是菜单、用户登录部分、文件传输部分、状态栏、目录显示部分。

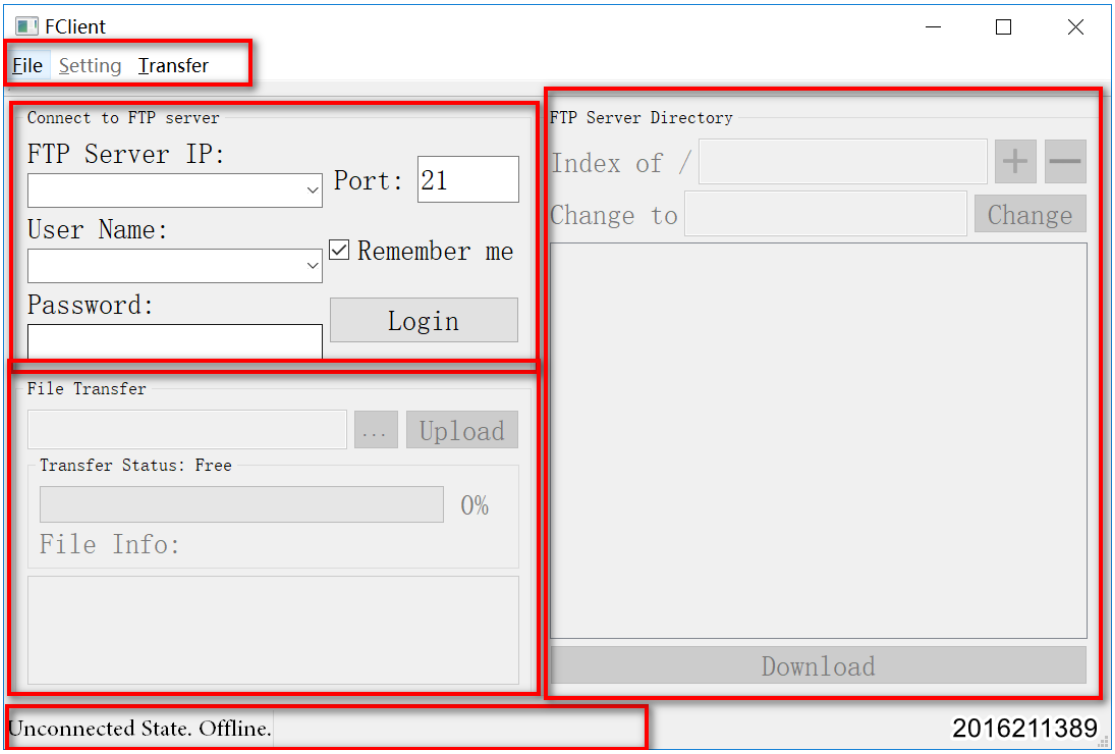

图 13 软件初始界面

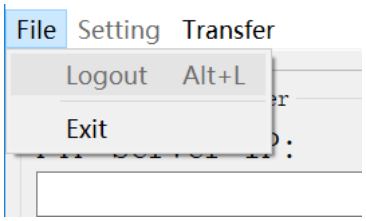

图 14 未登录状态的菜单栏

2. 登录后

菜单栏可选、用户登录部分失效、文件传输部分可选、状态栏显示登录信息、目录显示 FTP 服务器根目录。

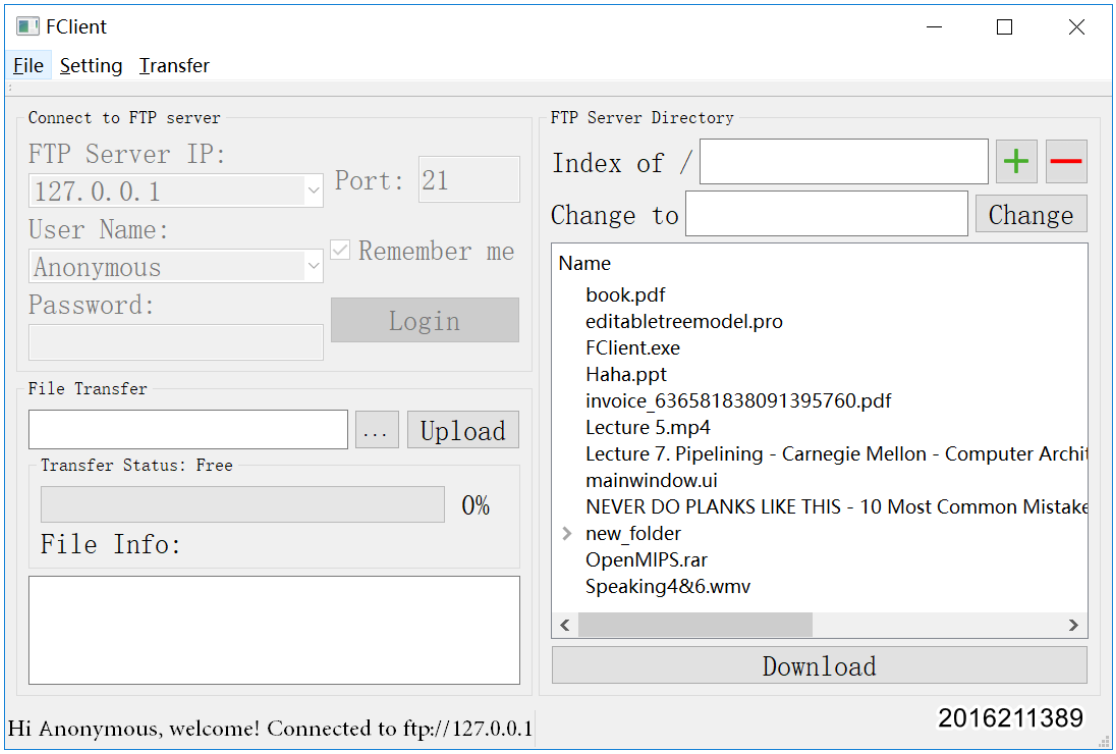

图 15 登录后界面

3. 删除文件

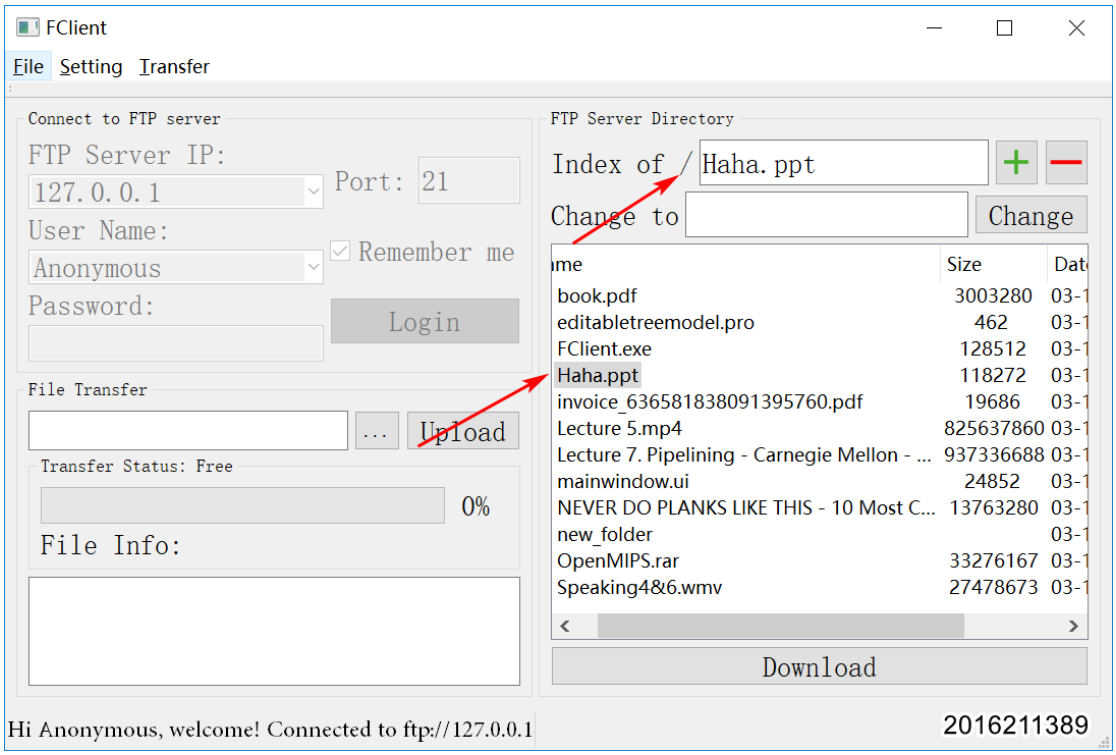

图 16 选中文件

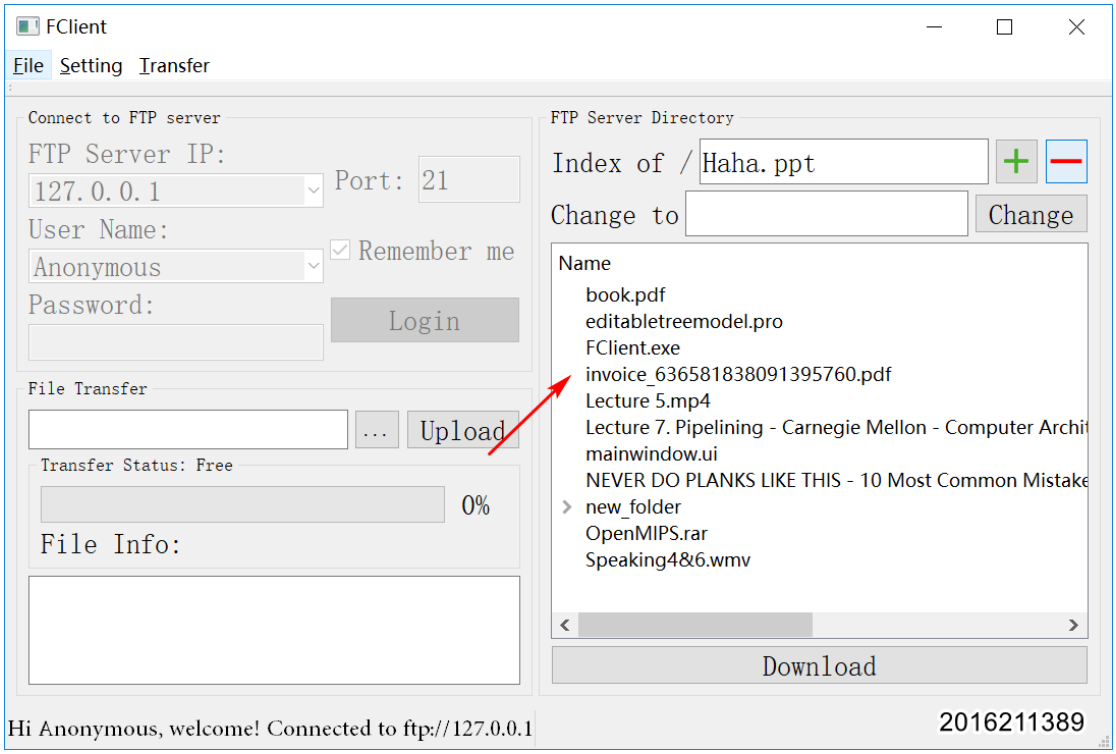

图 17 选中的文件被删除

4. 添加目录

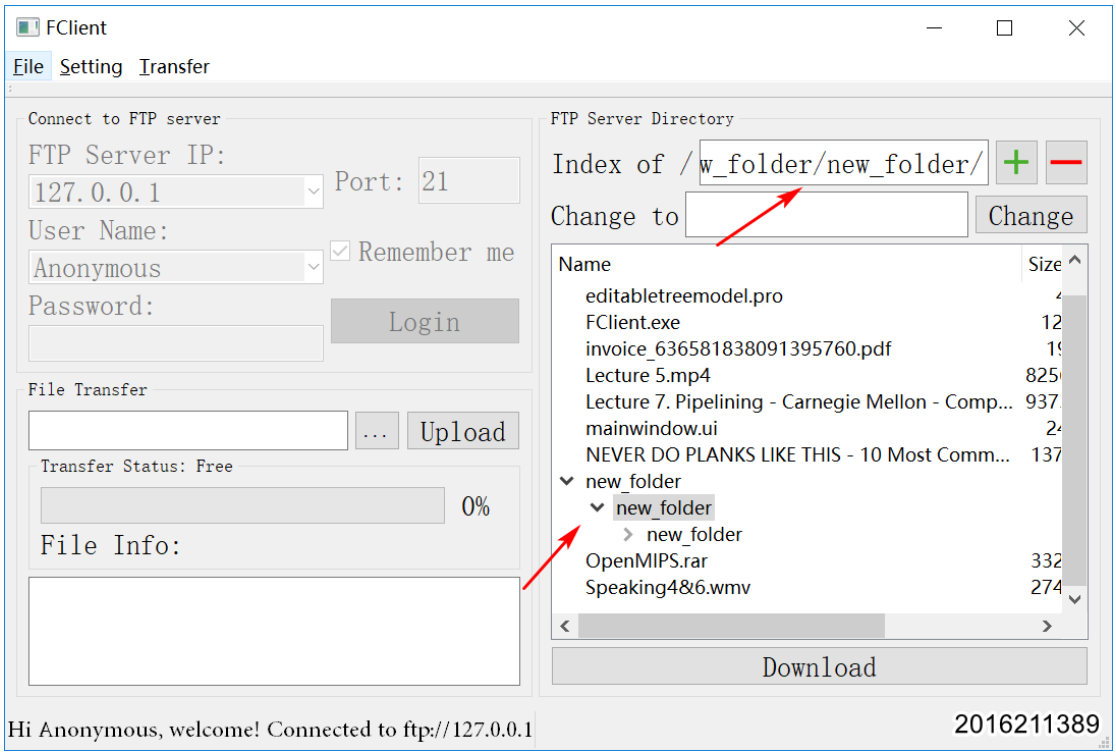

图 18 添加前

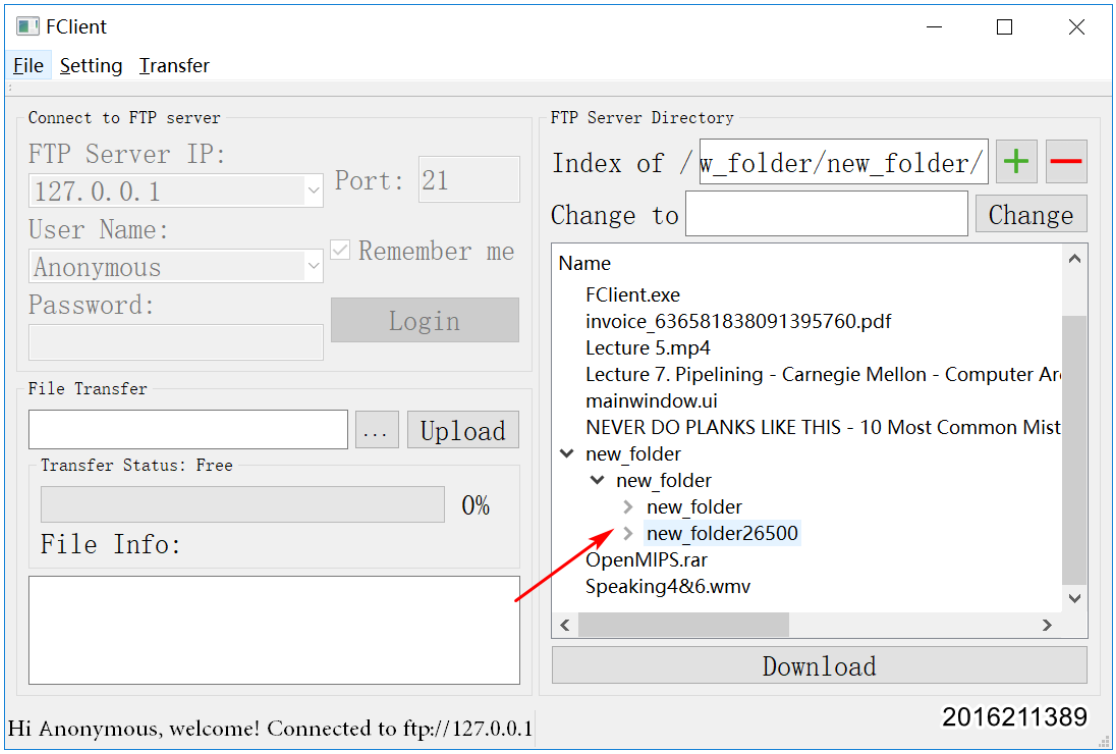

图 19 添加后

5. 修改目录

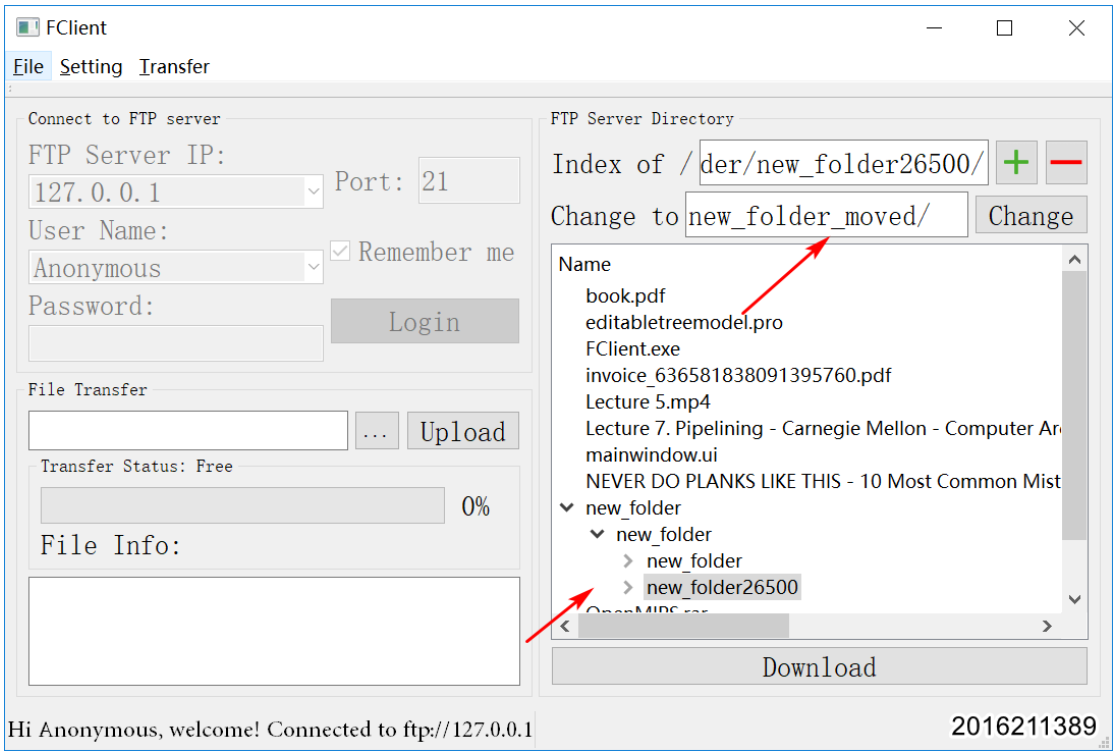

图 20 修改前

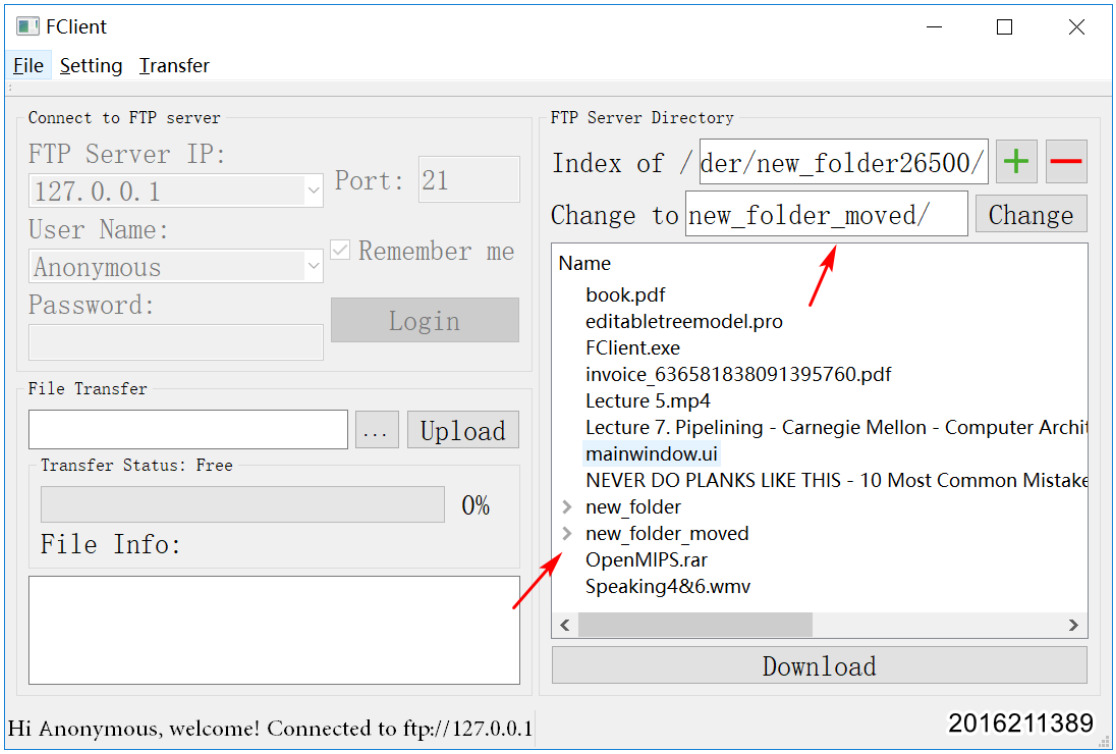

图 21 修改后

6. 传输文件

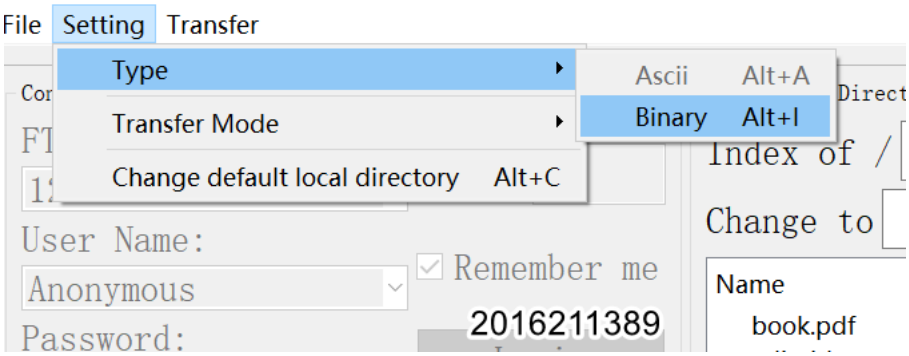

图 22 参数设置：Ascii 模式

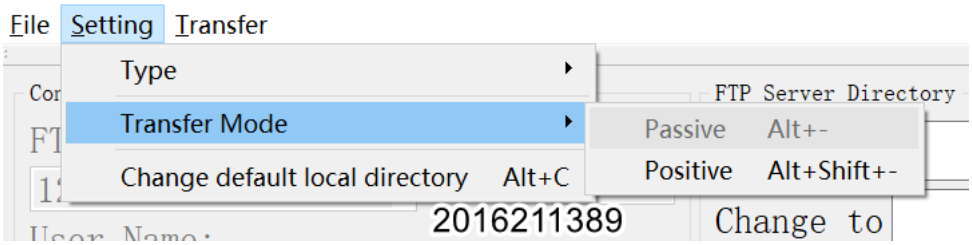

图 23 参数设置：被动模式

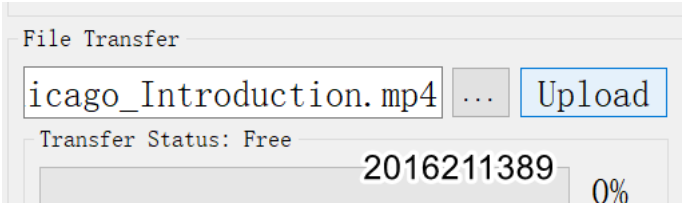

图 24 上传文件

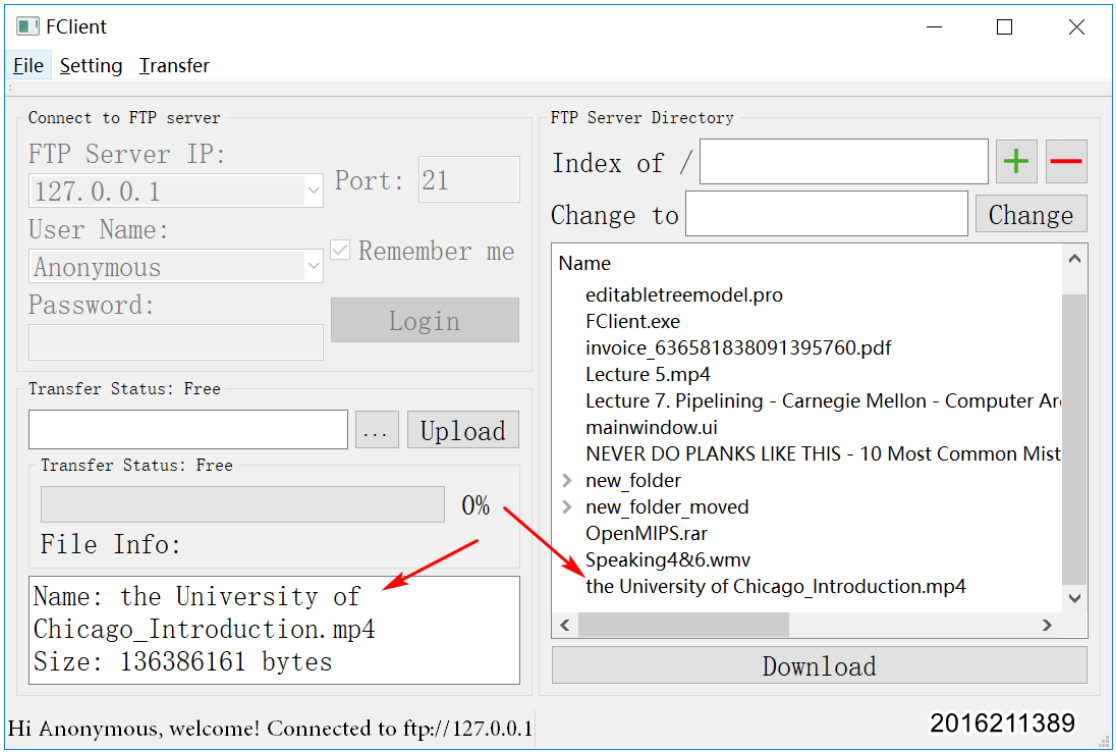

图 25 上传成功

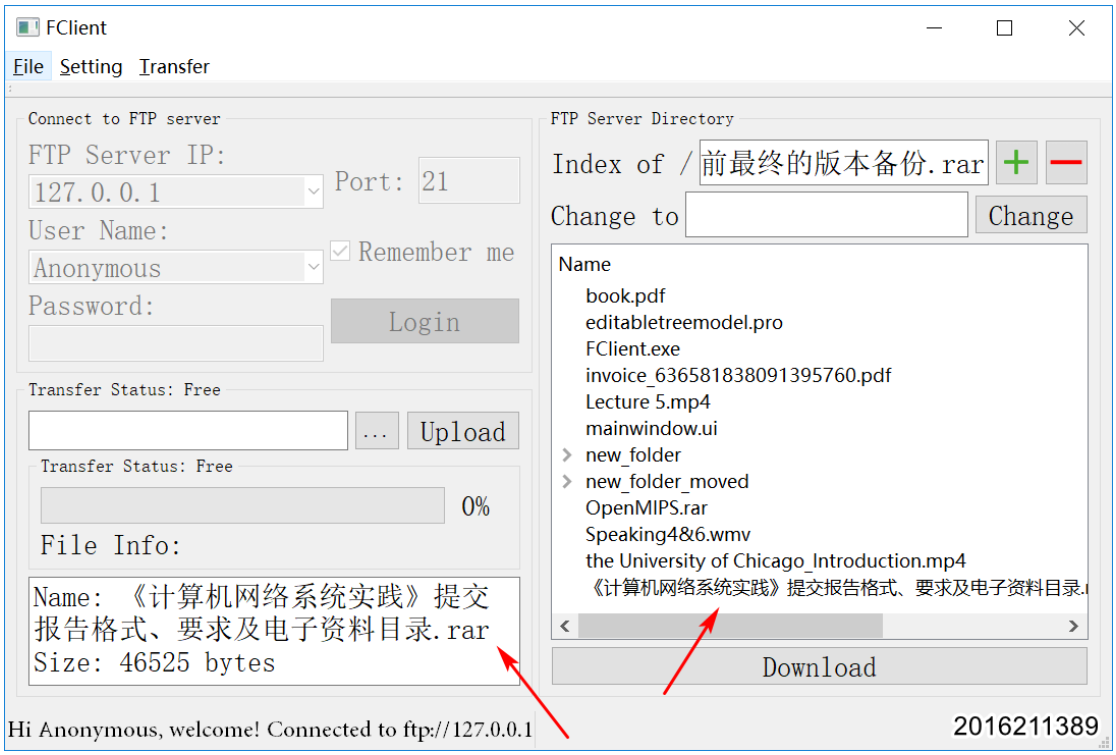

图 26 支持含中文文件名

八、软件使用说明

点击 FClient.exe 文件打开软件，此时的状态为离线，未连接。使用软件的第一步必须连接并登录到一个 FTP 服务器。输入用户信息，FTP 服务器 IP 地址。请注意 IP 地址一栏只能输入形如“192.168.43.17”这样的点分十进制格式信息，

并且要在英文输入法下，其它语言的输入法会发现无法键入。若按下 Login 按钮后报错，有警示框弹出，可能提示的错误有 IP 地址错误，用户名密码错误。您应当检查输入是否正确，其次可以根据错误码检查本地主机网络连接状态，或退出重试。当登陆成功，会看到下方状态栏以及用户界面的变化，看到目录显示。现在您可以自由地浏览目录！

双击可以展开或关闭一个目录。若想删除一个文件，需要先选中该文件或文件夹，点击右上角红色的-按钮。尤其注意的是，若想删除的是一个文件夹，当您按下-按钮，会将该文件夹下的所有文件一并删除！因此，请谨慎操作。若想在某一目录下新建文件夹，应先选中该目录，然后点击右上角绿色+按钮。新建的文件夹以 new_folder 开头，可能会根据相同目录中是否已存在重名（也为 new_folder）的文件进行更名。若想更改文件名，则可通过在 Change to 一栏中输入该文件新的目录(注意是您想要的完整路径名)。可以更改其父级目录到 FTP 服务器目录到任何一个目录，这样便实现了更改文件名的同时，也修改了文件所在位置。这些功能对文件和文件夹都适用。

若想进行文件传输，如上传，可以点击左下方框中的三点按钮，会弹出本地文件资源管理器。从中选中你想要上传的文件，再点击 Upload 按钮，传输成功后，便可以在右侧相应目录中看到该文件，包括其名称、大小、最近修改日期。若想下载文件，则可以通过先在右侧显示目录中选中文件，再点击 Download 按钮。若想改变文件传输方式，根据您所使用的主机，访问的 FTP 服务器所在的主机和当前防火墙设置，可以点开左上方 Setting 一栏，进行模式选择。

若完成工作希望登出，可以选择 File 目录，点击 Logout，或直接点击同一目录下的 Exit，这将同时关闭软件。目录栏中的每一项都有键盘快捷键，可以先按下 Alt 键不放松，再在键盘上按下你所希望选择菜单文字下方此刻出现下划线的英文字母，便可以直接触发该按键。

九、参考资料

[1] J. Postel 和 J. Reynolds. RFC 959: File Transfer Protocol[EB/OL]. ISI，1985 十月.

[2] 侯整风等. 网络与信息安全系列课程实践教程[M]. 合肥工业大学出版

社，2006.

十、验收时间及验收情况

2019 年 3 月 7 日验收，当时老师提出用户界面应当更加考虑用户体验，如文件传输的进度要有提示，目录展示的功能要完善。当时老师还提到使用 Qt 中提供的类进行编程也是一种不错的面向对象编程学习，并且 Qt 在封装操作系统提供的接口时，比直接使用 VC 编程，增强了安全性能。只是本次课程设计的目的是让我们熟悉网络编程，而 WinSock 是 Windows 上最为规范的接口，有利于直接观察到 FTP 协议的传输命令及其响应。

十一、设计体会

上学期的计算机网络主要讲述了网络的理论知识，概念多，很抽象。又因为我之前对网络编程知道的比较少，所以一些概念感觉理解的不是很深，如传输层数据的发送响应，只是知道概念和几种典型的成功或出错的情形。所以我选择了这一道题，因为觉得 FTP 是一个很有用的工具，学习它的协议有助于我巩固网络的知识。同时我也很好奇，因为从小到大上机房时都会用到它。

着手做时，我首先仔细学习了 FTP 协议内容，从开始建立 TCP 连接，到用户登录获得授权，FTP 拥有哪些指令，服务器又会给客户端何种响应。大致了解后，我先跟着教程使用 IIS 建立一个位于我笔记本上的 FTP 服务器，支持匿名登录，具有读写权限，再在命令提示符中试了试，发现微软的 ftp.exe 已经对 FTP 指令经过了一层包装。

接下来我学习了 Qt 的网络编程，并开始用 QTcpSocket 和 QTcpServer 类写程序。熟悉了 Qt 提供的网络编程类之后，就觉得越来越好用，它已经提供多种信号函数，多种连接状态和错误的枚举类型，所以我只需要写一个类来管理 FTP 功能操作。后来“不幸”地发现，要求使用 WinSock 接口，于是在实验指导书上学习了 winsock 的使用流程，函数、参数介绍等。为了尽量少的走回头路，我觉得应该借鉴 QTcpSocket 的思想，也写一个管理套接字连接的类，向 FTP 类提供服务。我在 qtbase 中找到了对应的源码，但发现内容十分复杂，层次很多，没有深挖。自己完成 FSocket 类时，有了前后的对比，发现自己所写的类在网络状况侦测，提供状态改变的信号，都很粗糙。所以我尽量通过及时调用 WSAGetLastError 函数获知错误代码，在会有 FTP 响应发回时调用 recv()读取。

使用自己写的功能并不完善的函数，便可以深刻体会到数据在网络连接中发送的状态,当然，是在遇到了之前没有遇到的一些 bug 后才更加印象深刻的。通过上网查找错误原因，解决方案，学到了很多，对网络端口的工作，之间的联系配合也有了更深的了解。

在实现界面的过程中，由于我中间更换、添加了另外的 ui 界面（虽然在最终版中没有，因为原先的错误解决了，添加界面也就不必要了），导致突然出现了很多错误。仔细的观察 Qt designer 界面，发现导致一些错误的原因（往往编译器无法给出这类切中要害的错误提示）是没有删除干净一些 Action 及其连接的信号，之后还要清理项目，重新编译该 ui 文件（在 Qt Creator 中则是清理项目，并重新 qmake）。

通过本次网络工程师综合实训，我学习了很多，巩固了很多传输层网络知识，学习了应用层的 FTP 协议，还拥有了一款自己的 FTP 客户端软件 FClient。