基于 MFC 下重叠 I/O 模型的 FTP 文件传输设计与实现

张效源 白俊鸽
四川大学锦城学院计算机与软件学院 四川 成都 610000

【摘 要】随着信息技术的不断发展与信息传输的大量需求，资源共享和信息传播成为了信息生活中最基础且最重要的组成部分，网络传输是实现信息共享的重要方式，其中 ftp 是在实现数据传输与信息传递的过程中使用最为广泛的，同时如何让文件传输在高效的系统性能下实现传输过程也是所应追求的目标。本项目设计并实现了重叠 I/O 模型下的 ftp 服务端和客户端的系统，在 MFC 的基础下，以重叠 io 模型的方式将系统性能提高的前提下。采用 visual studio 2019 的开发环境，实现了文件上传以及下载，新建目录，登录，关闭连接等功能。

【关键词】MFC；FTP；文件传输；重叠 I/O 模型

1 FTP 概述

FTP 是用于控制文件传输的协议之一。由两个部分构成，即 FTP 服务器和 FTP 客户端。其中服务器用于存储资源文件，用户通过 FTP 连接上服务端并通过连接访问服务器上的资源。客户端可以从远程计算机上下载文件到本地，或者将本地文件上传至远程计算机。由于其传输文件的效率非常高，一般在网络上传输大的文件时通常采用该协议进行传输。

在通常的情况下 FTP 协议使用 20 号端口和 21 号端口，这两个端口分别用于创建数据连接用以数据传输和创建控制连接用以控制信息传输。但是最终使用的传输数据的端口号与 FTP 的传输模式有关。在主动模式下，使用 20 号端口创建数据连接；若是被动模式，则服务端将会在本地开放一个相应的端口，然后将服务端开放的端口告诉客户端，客户端再与其开放的端口创建数据连接。

2 系统内容

本系统站在 FTP 的文件传输系统和性能高效的角度上进行设计。实现 FTP 的文件传输，目录显示和客户端发送响应命令服务端执行相应的操作的功能。

3 总体框架设计

本系统在 win10 的开发环境下，采用 visual studio 2019 开发环境，利用 mfc 对整体的软件界面进行设计，并通过使用重叠 I/O 模型来提高系统的性能进行开发，其中服务器和客户端包含界面和事件处理两个部分，界面框架主要是对用户在界面的操作响应，事件处理包括 socket 之间的通信连接以及客户端与服务端数据交互等操作。

4 FTP 服务端设计及实现

4.1 服务端界面设计

本项目服务端设计界面如图 1 所示：

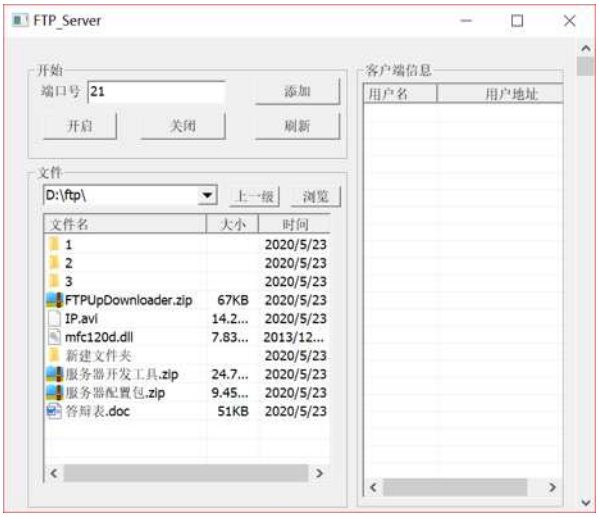

图 1 服务端设计界面

根据服务端的功能目标，将界面分为开始部分，文件列表显示区域，和用户登录显示 3 个部分组成。

窗口设计内容主要包含：

（1）开启关闭服务器部分，该部分主要用于控制服务器端的开启和关闭以及对目录的刷新。

（2）显示文件列表窗口，该窗口用于显示服务器

端的文件目录信息，服务端可根据选择进入相应的路径，其中浏览可以进如服务端本地的目录，服务端可在其自己本地添加文件夹和删除相应的文件等的对文件的操作

（3）用户信息界面显示窗口，还窗口可以显示登录用户的信息，用于查看当前连接的用户数目。

4.2 服务端模型选择及实现

本项目采用重叠 I/O（overlapped）模型，并选择事件通知方式在网络事件完成后通知缓冲区使用数据，它能够使系统达到更好的性能。

4.2.1 重叠 I/O 模型原理

重叠 I/O 模型通过使用重叠的数据结构（WSAOVERLAPPED）[1]。在相应的每次投递请求完成后，程序会收到完成的通知，于是我们就可以对数据进行相应的处理。其中每次发送的一个或多个 I/O 请求都是非阻塞 / 异步的，当请求交由系统后，程序可以去做其他的事情。在发送一个请求后，这个请求会交由系统进行处理，至于请求在什么时候完成，会在操作完成时告诉程序哪一个对应的请求完成即可。

4.2.2 重叠 I/O 模型的使用和注意事项

（1）WSA_FLAG_OVERLAPPED 作为重叠 I/O 的标志，在使用 WSAsocket 进行套接字创建时，需要使用该标志，而在使用 socket 函数时会默认设置。当创建好后，需要将其和本地接口绑定后即可进行操作。

（2）使用重叠结构时，WSASend、WSARecv 等函数将会替代 send、recv 等函数将 [1]。因为重叠函数都包含一个 Overlapped 参数，只有这个结构的参数才能够说明支持重叠的数据结构，即支持投递异步的重叠 I/O 请求。

（3）在调用重叠函数完后不管有没有数据都会马上直接返回。当获取错误码时通常情况下会获取到返回值：WSA_IO_PENDING，这表明函数调用成功，但是 I/O 操作还在进行中。重叠结构与 Windows 的事件对象进行了相关的绑定，每当在调用完函数后就可以等待操作完成后进行相应的通知。随后就可以来根据重叠操作的结果取得我们想要的数据了。

例如在调用 WSARecv 时，不管对方 socket 是否有数据到来，只要调用就会返回告诉函数调用，没有数据到来时操作就会放在系统的一个队列中，系统会记录此套接字当前有一个收数据的操作，如果在以后的某一刻，在此套接字上有数据到来，那么 WSARecv 操作，系统将会通知程序数据已经接收到。

（4）获取 I/O 请求是否成功，主要有事件对象通知和完成例程这两种方法，常用的是事件对象通知这一种方法，对于事件对象通知，使用重叠函数的时候都会提交一个 Overlapped 的对象。这个结构的对象中的 hEvent 字段都必须去关联 Win32 事件对象 / 时间句柄对象。该事件的创建的默认状态是“未传信”状态。当投递一个 I/O 请求之后，事件对象状态但是仍是“未传信”状态，当投递的 I/O 请求完成之后，状态则会转变成“已传信”。

在变成“已传信”状态之后可通过函数 WSA Wait For Multiple Events 等待所设置的事件对象数组，该数组中包含调用的重叠函数中的 Overlapped 结构中的 hEvent 字段，如果返回则说明某个事件状态变成“已传信”状态。说明发现某个 I/O 请求完成。当发现请求完成后，通过调用函数 WSAGetOverlappedResult，用于查看该 I/O 请求是否成功。

4.2.3 重叠 I/O 模型的编程步骤：

（1）创建套接字并监听。

（2）接受客户端的连接请求。

（3）创建 WSAOVERLAPPED 结构，并分配事件对象句柄。

（4）在套接字上投递异步请求。

（5）调用 WSAWaitForMultipleEvents，并等待事件进入“已传信”状态。

（6）调用 WSAResetEvent 将“已传信”状态的事件对象重置。

（7）使用 WSAGetOverlappedResult 判断重叠调用状态。

（8）在套接字上投递另一个重叠请求。

（9）重复步骤（5~8）。

4.3 服务端事件处理的实现

服务器后端主要处理与客户端之间的连接和信息传递以及界面操作的响应等，服务端的工作流程及功能实现大致如下：

4.3.1 界面显示处理

（1）文件区的显示，操作和路径的改变

文件显示区主要是显示服务端的文件列表和当前的路径信息，其中包含的函数及操作如下：

a. 通过调用 Init 函数进行界面初始化；

b. 通过调用 etFileIcon 函数 获取文件图标；

c. 通过调用 Findfile 函数查找本地文件；

d. 通过调用 ShowFileList 函数实现文件列表显示；

其中文件夹的大小采用空显示，因为文件夹的大小计算需要采用层层递进的方式计算出，针对很大的文件夹的时候，需要太长时间计算。

e. 通过调用 OnInsertFile 函数实现插入文件内容；

f. 通过调用 GetDiskInfo 函数实现获取磁盘信息；

g. 通过调用 showdir 函数实现将当前的路径显示到 combox 中；

h. 通过调用 OnBnClickedViewBtn 函数实现浏览本地

文件的响应；

i. 通过调用 OnBnClickedBackBtn 函数实现返回路径上一级的响应；

j. 通过调用 OnCbnSelchangeSelectdir 函数实现路径选择改变的响应；

k. 通过调用 OnNMDblclkList1 函数实现右击和双击下一级改变事件的响应；

l. 通过右击鼠标调用 Ondelete 函数实现删除事件的响应；

m. 通过调用 DeleteDirectory 函数实现删除文件夹事件的响应；

n. 通过调用 OnAddNew 函数实现新建文件夹事件的响应；

其中右击鼠标将会有删除和新建目录的选择。

（2）用户区的显示

在用户创建好连接后将会直接执行以下操作用于界面数据的插入。

pSI->pMainWnd->User_List.InsertItem(pSI->pMainWnd->User_List.GetItemCount(), _T(xxx));

pSI->pMainWnd->User_List.SetItemText(rowId, 0, User);

pSI->pMainWnd->User_List.SetItemText(rowId, 1, pSI->CIP);

（3）开始区域

这部分区域主要用创建服务器和客户端两者之间的连接。

主要函数有：

a.void CFServerDlg::OnBnClickedStartBtn() 实现开启连接的按钮

b. void CFServerDlg::OnBnClickedFlashBtn() // 实现刷新

c. void CAboutDlg::OnBnClickedStopBtn() // 实现关闭

4.3.2 服务端连接及事件处理实现

（1）在点击开始按钮后将会执行线程 Listen Thread Func，该线程实现去监听客户端的连接的功能，同时手动开启一个侦听事件，Event[0] = WSACreateEvent() 第一个事件是给侦听事件的，不然会一直收到连接信息但是不会有数据。

（2）在线程中再开启一个 ProcessTreadIO 线程，该线程用于处理 i/o 请求。

（3）当用户创建好了与服务端的陆信息，并要求用户给控制连接后将会返回欢迎登出用户名和密码，开启登录验证用于验证用户输入的用户名和密码是否匹配，该操作将会通过执行函数 WelcomeInfo 和 LoginIn 实现其功能。

（4）当用户连接后，后服务端将会执行 WSAWaitForMultipleEvents 操作来等待用户发送请求，并通过 g_index - WSA_WAIT_EVENT_0，计算出是哪一个用户发送的请求。同时通过 WSAGetOverlappedResult 操作来判断是否有收档相应的数据，日过收到的字节数小于 0 则将相应的 Event 移除，如果是接收状态则将执行 DealCommand 函数去执行响应的操作。

5 FTP 客户端设计及实现

5.1 客户端界面设计

本项目服务端设计界面如图 2 所示：

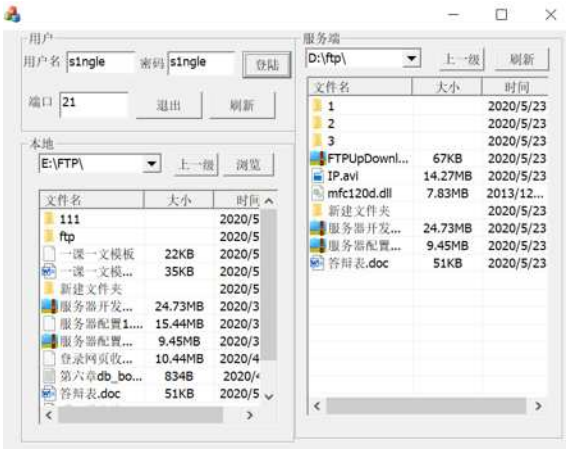

图 2 服务端设计界面

根据客户端的功能目标，将界面分为用户部分，本地文件显示显示框部分，服务端文件列表显示区域 3 个部分组成。

窗口设计内容主要包含：

（1）用户部分，该部分主要用于用户输入登录密码和输入用户名以及退出连接的操作。

（2）显示本地目录窗口，该窗口可以显示本地端的文件目录，本地可根据选择进入相应的路径，其中浏览可以进如本地的目录，用户可在其自己本地添加文件夹和删除相应的文件等的对文件的操作，以及选择文件进行上传。

（3）服务端信息界面显示窗口，该窗口可以用于显示服务端的文件内容，用户可以对服务端的文件进行操作，如下载，删除以及创建文件夹等。

5.2 客户端事物处理的实现

服务器后端主要处理与客户端之间的通信和界面操作的响应等，服务端的工作流程及功能实现大致如下：

5.2.1 界面显示处理

（1）文件区的显示，操作和路径的改变

其中客户端的本地界面操作和显示服务端信息及操

作与服务端操作服务器笨的的操作一致，不再赘述。

5.2.2 客户端连接及事件处理实现

（1）在点击登录按钮后将会执行线程 Connect-ThreadFunc 去建立与服务端的连接。

（2）连接建立后将会根据相应的操作发送响应的命令。

6 FTP 客户端和服务端对命令和响应的处理

6.1 数据连接（默认主动）

在进行数据传输的过程中客户端和服务端回重新创建 socket，同时在客户端 accept 之前的时候将会先设定文件传输方式和连接模式，客户端执行 SetType 和 SetModele 函数进行设置，服务端将会返回相应的响应码，在 SetModele 操作中，客户端将会执行 Convert-CommaAddress 函数对地址进行转换，将点转化为逗号，并将地址传给服务端，服务端接受到后会调用 ConvertDotAddress 方式解析出对应的端口号，这些操作成功后客户端将会执行 accept 操作，数据建立连接成功后将会执行数据传输操作。

6.2 显示目录信息

客户端发送命令后，服务端接受到后将会执行 FileListToStr 函数将文件目录转换成字符串存到相应的 buff 中，并在每一个存入的数据后面执行 strcat(buff, \r\n) 操作，这样操作将会在每一个数据后面添加上“\r\n”的标志，有利于客户端对接收到的数据进行对应的解析。客户端接收到消息后将会执行 strtokchar 函数使用 strtok(buffdata, \r\n); 方式将数据以 \r\n 为分割符存入 vector 中，再执行 StarctData 函数将数据抽取出来并插入到列表中。

6.3 上传文件操作

客户端信息显示右击后选择上传操作将会执行上传文件操作，该操作使用 MFC 的 CFile 类来进行操作，cfile 将文件打开并将数据读取到 buff 中进行传输，调用 SendAll 函数将数据发给服务端，服务端回创建一个同名的文件，并将数据写入到文件中。

6.4 下载文件操作

与上传类似。

6.5 显示当前目录和改变父目录以及进入到目标目录

该项目使用定义的 ServerPath（服务端路径的 combox 显示值），将路径做相应的操作后获取需要的目录并将其传给服务端，服务端执行 list 相关的操作获取到目录信息以及调用 DiskinfoToCs 函数获取磁盘信息将其返回，客户端执行 showdisk 和 showlist 操作将数据插入到对应的 combox 和 list 中。

6.6 创建目录 / 删除目录（文件）操作

将路径传输到服务端，若是创建则执行 Create Directory 进行创建 / 若是删除则应执行 Get File Attributes(str) == FILE_ATTRIBUTE_DIRECTORY 操作判断是否是文件夹，若是文件夹则执行 DeleteDirectory 操作，若是文件就执行 DeleteFile 操作。

6.7 退出操作

关闭响应的连接。

以上操作执行完后都将会关闭数据连接。

7 结束语

虽然本次实现的 FTP 文件传输系统已经能够实现传输流程，但是站在使用的角度还可以做更多的完善，比如在支持多个文件选择传输的情况以及断点续传的情况。在以后的学习中，还应该更加努力地完善项目。

【参考文献】

[1] 吴永明 , 何迪 . 基于完成端口的服务器底层通信模块设计 [J]. 信息技术 , 2007(03).

[2] 盛利 , 刘旭 . 用完成端口管理 Windows Sokcet[J]. 现代计算机 (专业版). 2001(07).