实 验 十二　FTP 应用编程

12.1　实验目的

了解和掌握 FTP 协议，能够利用 FileZilla 文件搭建和配置 FTP 服务，采用.NET 平台类实现 FTP 客户端程序，支持文件的上传下载。

12.2　FTP 协议介绍

文件传输协议（File Transfer Protocol：FTP）是在网络上进行文件传输的一套标准协议，它是 TCP/IP 网络结构的应用层协议。它是是一个客户机/服务器系统 (C/S)，支持任何类型的文件在多种操作系统之间完成文件交流。用户通过 FTP 可以把自己的文件传送给服务器，或者从其它的服务主机获得文件，往网站空间上放网站文件的时候，也经常采用 FTP 方式。

FTP 的操作流程是客户机采用 FTP 服务器授权的帐号执行登录过程，向服务器发出命令下载文件，上传文件，创建或改变服务器上的目录。许多 FTP 服务主机还支持匿名服务，用户输入账号："anonymous" 就可以登录服务器。

FTP 服务一般运行在 20 和 21 两个端口。端口 20 用于在客户端和服务器之间传输数据流，而端口 21 用于传输控制流，并且是命令通向 ftp 服务器的进口。当数据通过数据流传输时，控制流处于空闲状态。而当控制流空闲很长时间后，客户端的防火墙会将其会话置为超时，这样当大量数据通过防火墙时，会产生一些问题。此时，虽然文件可以成功的传输，但因为控制会话会被防火墙断开，传输会产生一些错误。

FTP 协议比 HTTP 协议要复杂，FTP 协议使用两个 TCP 连接，一个是命令链路，用来在 FTP 客户端与服务器之间传递命令；另一个是数据链路，用来上传或下载数据。FTP 协议有两种工作方式：PORT 方式和 PASV 方式，中文意思为主动式和被动式。

PORT（主动）方式的连接过程是：服务器开放一个端口，通知客户端连接，服务端接受连接，建立一条命令链路。客户端在命令链路上用 PORT 命令告诉服务器：“已打开了一个 1024+ 的随机端口，等待连接”。服务器从 20 端口向客户端的 1024+ 随机端口发送连接请求，建立一条数据链路来传送数据。客户端由于安装了防火墙会产生一些问题。

PASV（被动）方式的连接过程是：客户端向服务器的 FTP 端口（默认是 21）发送连接请求，服务器接受连接，建立一条命令链路。服务器在命令链路上用 PASV 命令通知客户端："已打开了一个 1024+ 的随机端口，等待连接"。客户端向服务器的 1024+ 端口发送连接请求，建立一条数据链路来传送数据。被动方式 FTP 中，命令连接和数据连接都由客户端发起，这样就可以解决从服务器到客户端的数据端口的入方向连接被防火墙过滤掉的问题。

FTP 协议的命令由表12-1表示。

表 12-1 FTP 协议的命令

命令	参数	描述
ABOR	无	中断数据连接程序
ACCT	account，账号	系统特权帐号
ALLO	bytes，字节数量	为服务器上的文件存储器分配字节
APPE	文件名	添加文件到服务器
CDUP	文件路径	改变服务器上的父目录
CWD	文件路径	改变服务器上的工作目录
DELE	文件名	删除服务器上的指定文件
HELP	命令名称	返回命令信息
LIST	命令名称	文件列表
MODE	模式代码	传输模式（S/B/C）
MKD	新目录名称	在服务器上建立指定目录
NLST	目录名称	列出指定目录内容
NOOP	无	服务器在线连接确认
PASS	账户密码	账户登录密码
PORT	IP 地址与端口	进入主动传输模式
PASV	无	要求进行被动传输模式
PWD	无	显示服务器当前工作目录
QUIT	无	退出登录 FTP 服务器
REIN	无	重新初始化登录状态连接
REST	文件偏移量	由特定偏移量重启文件传递
RETR	文件名	从服务器上下载文件
RMD	目录名	对旧路径重命名
STAT	目录名	在当前程序或目录上返回信息
STOR	文件名	储存（复制）文件到服务器上
SYST	无	返回服务器使用的操作系统
USER	用户账号	系统登录的用户名

FTP 协议中服务器的响应代码由表12-2表示。

FTP 主要特点是支持不同平台间的文件传输，支持断点续传功能，其主要缺点是密码和文件内容都使用明文传输，易被窃听。支持 FTP 服务的软件有 FileZilla 和 Xlight 等。

12.3 FileZilla 安装 FTP 服务和参数配置

FileZilla 是一个免费开源的 FTP 软件，有客户端版本和服务器版本，软件性能优越。FileZilla 的官网是：https://filezilla-project.org/，用户可方便下载其客户端与服务端程序。客户端版本可以运行在 Linux，window，Mac OS 系统上，服务器版本可以运行在 Window 系统中。

FileZilla 服务端安装不复杂，运行 FileZilla_Server.exe 即可完成安装，安装过程可参考图12-2。安装成功后 FileZilla 服务端以服务方式在 Windows 平台运行。

Win8 平台需防火墙放开对 FileZilla 服务的封闭，以下是参考 FileZilla 官网的防火墙配置过程：

1. 打开控制面板中的防火墙项目，如图12-7。

表 12-2 FTP 协议的响应代码

响应代码	说明	响应代码	说明
110	新文件指示器上的重启标记	120	服务器准备就绪的时间（分钟数）
125	打开数据连接，开始传输	150	打开连接
200	成功	202	命令没有执行
211	系统状态回复	212	目录状态回复
213	文件状态回复	214	帮助信息回复
215	系统类型回复	220	服务就绪
221	退出网络	225	打开数据连接
226	结束数据连接	227	进入被动模式（IP 地址、ID 端口）
230	登录因特网	250	文件行为完成
331	要求密码	332	要求帐号
350	文件行为暂停	421	服务关闭
425	无法打开数据连接	426	结束连接
450	文件不可用	451	遇到本地错误
452	磁盘空间不足	500	无效命令
331	要求密码	332	要求帐号
350	文件行为暂停	421	服务关闭
425	无法打开数据连接	426	结束连接
452	磁盘空间不足	500	无效命令
501	错误参数	502	命令没有执行
503	错误指令序列	504	无效命令参数
530	未登录网络	532	存储文件需要帐号
550	文件不可用	551	不知道的页类型
552	超过存储分配	553	文件名不允许
530	未登录网络	532	存储文件需要帐号

2. 防火墙操作界面上选择允许应用功能通过防火墙，如图12-8。
3. 在允许的应用窗体选择更改设置，再选择允许其它应用，如图12-9。
4. 在允许的应用中定位文件到 FileZillaServer.exe 文件，如图12-10。
5. 在已经允许的 FileZillaServer.exe 程序状态上勾选专用及公用属性，如图12-11及，如图12-12。
6. 在 Win8 的开始界面找到命令提示符，右键调出以管理员身份运行，如图12-11。
7. 执行命令 netsh advfirewall set global StatefulFTP disable。

XP 平台防火墙的设置与 Win8 类似，用户也可查阅 FileZilla 官网相关文档。

FileZilla 的 FTP 服务需要执行配置，设置账户后客户端才能使用，在 FileZilla 服务程序目录下运行 FileZilla Server Interface.exe 程序连接本机的 FileZilla 服务，注意初次连接的 IP 要设成"127.0.0.1"，如图12-14。添加一个匿名账号的操作如图12-15。

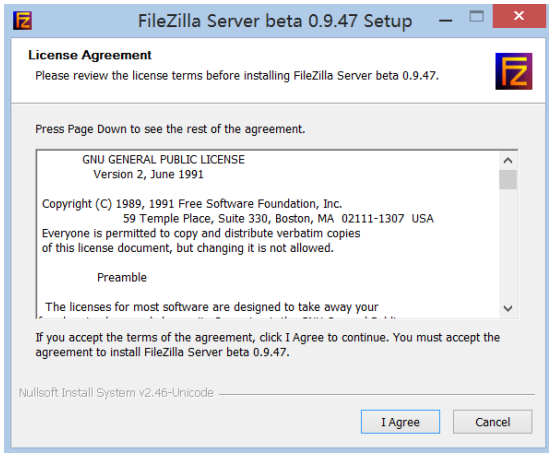

图 12-1 FileZilla 安装 –1

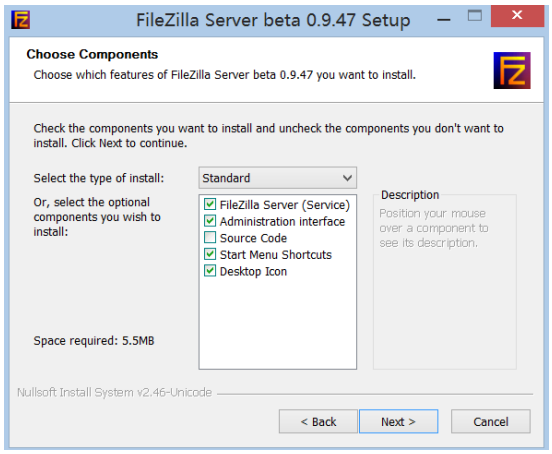

图 12-2 FileZilla 安装 –2

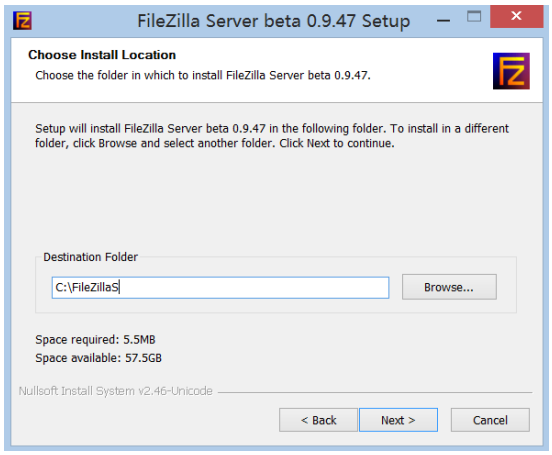

图 12-3 FileZilla 安装 –3

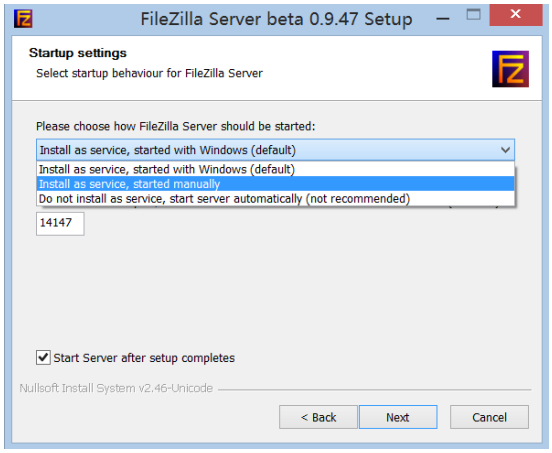

图 12-4 FileZilla 安装 –4

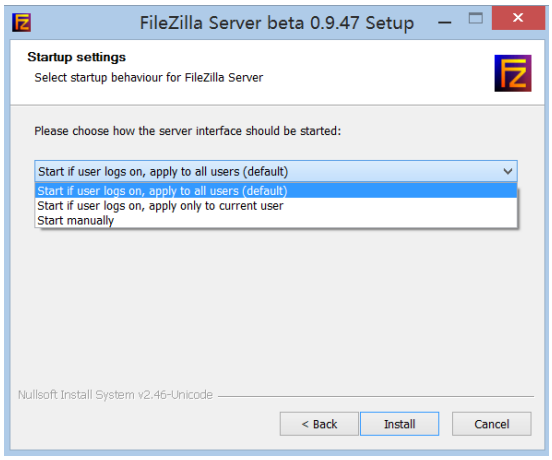

图 12-5 FileZilla 安装 –5

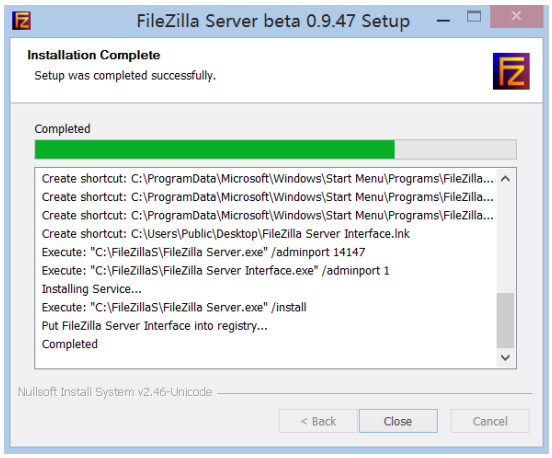

图 12-6 FileZilla 安装 –6

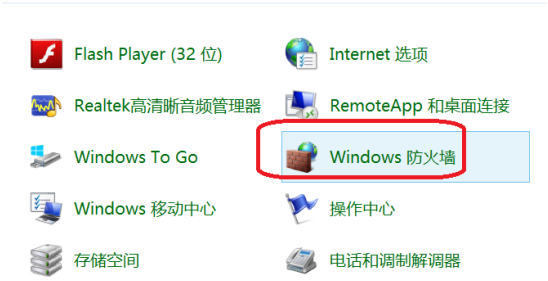

图 12-7 FileZilla 安装 –7

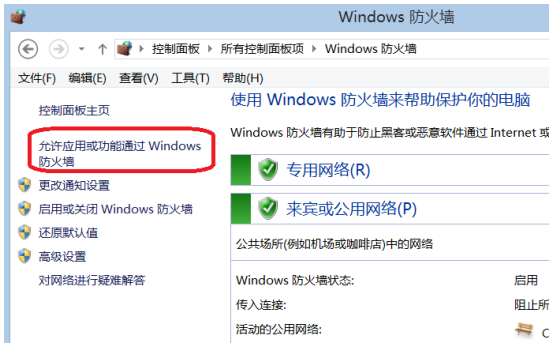

图 12-8 FileZilla 安装 –8

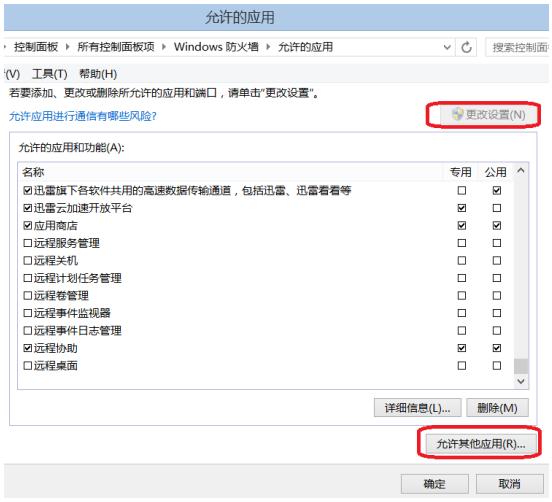

图 12-9 FileZilla 安装 –9

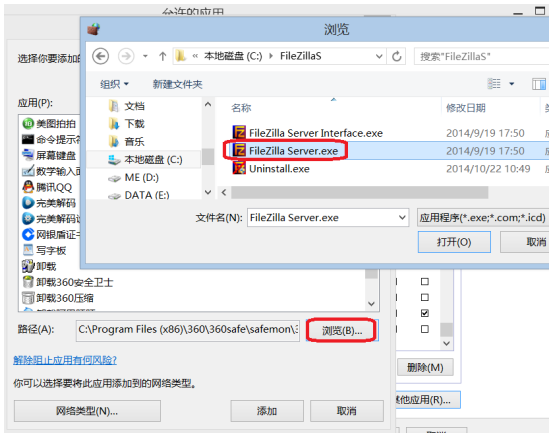

图 12-10 FileZilla 安装 –10

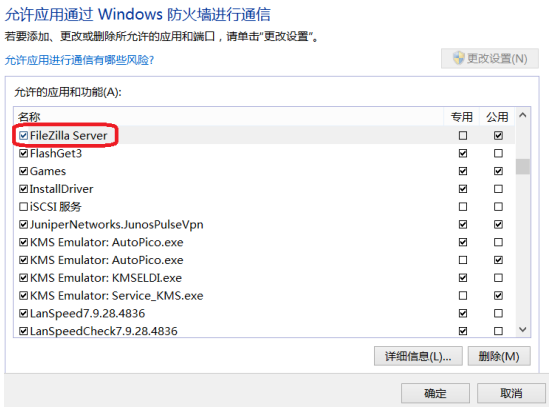

图 12-11 FileZilla 安装 –11

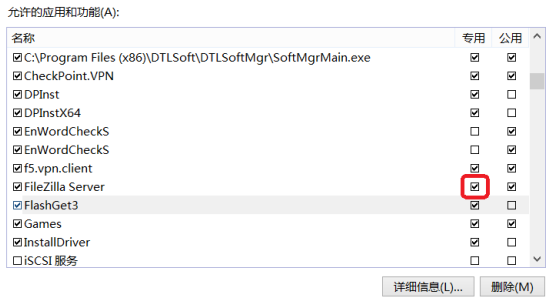

图 12-12 FileZilla 安装 –12

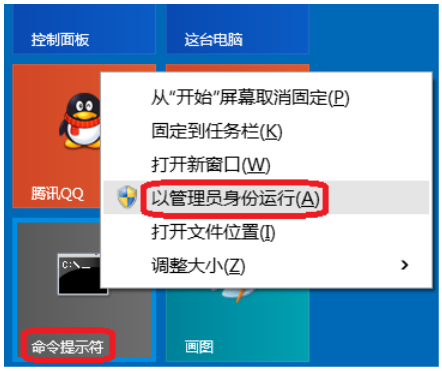

图 12-13 FileZilla 安装 –13

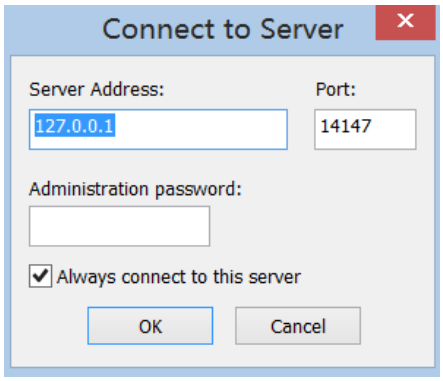

图 12-14 FileZilla 安装 --服务配置

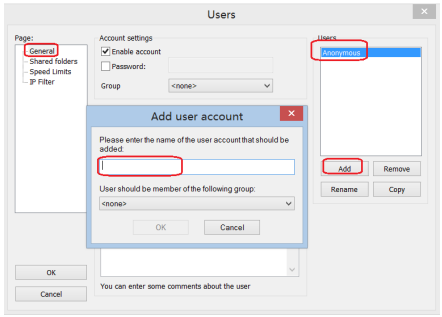

图 12-15 FileZilla 安装 --添加匿名账号

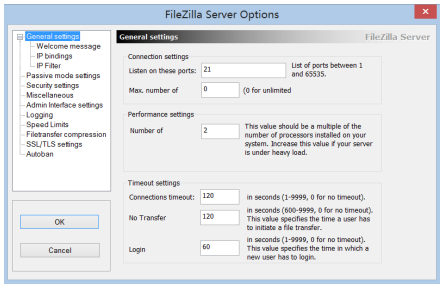

图 12-16 FileZilla 安装 --基本设置

12.4 .NET 平台 FTP 客户端编程

由于 FTP 协议比较复杂，直接基于 Socket 对象开发 FTP 应用难度高，示例程序是基于.NET 平台的 FTP 相关类生成应用。使用.NET 平台类编写简单的 FTP 应用，界面可参考图12-17。

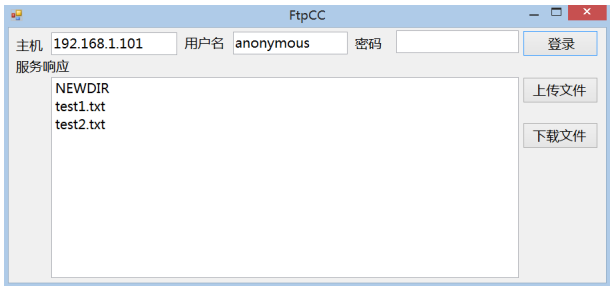

图 12-17 ftp 客户端界面

表 12-3 FTP 操作相关类

类名	描述
FtpStatusCode	将 FTP 协议中的状态码的文本形式
FtpWebRequest	客户端向服务器发送的请求内容如登录，上传命令等功能
FtpWebResponse	客户端接收服务器响应的各种信息
WebRequestMethods.Ftp	代表 FTP 协议中相应的命令，例如 Download-File 代表 RETR 命令

.NET 平台的 FTP 相关类由表12-3表示。以下是示例代码。

```
string ftpServerIP;
string ftpUserID;
string ftpPWD;
FtpWebRequest reqFTP;
private void Connect(String path)//连接 ftp
{
  // 根据 uri 创建 FtpWebRequest 对象
  reqFTP = (FtpWebRequest)FtpWebRequest.Create(new Uri(path));
  // 指定数据传输类型
  reqFTP.UseBinary = true;
  // ftp 用户名和密码
  reqFTP.Credentials = new NetworkCredential(ftpUserID, ftpPWD);
  string uri = "ftp://"+ ftpServerIP + "/"+path;
}
//从 ftp 服务器上获得文件列表
private string[] GetFileList(string path)
{
  string[] downloadFiles;
  StringBuilder result = new StringBuilder();
  try
  {
    Connect(path);
    reqFTP.Method = WebRequestMethods.Ftp.ListDirectory;
    FtpWebResponse response = (FtpWebResponse)reqFTP.GetResponse();
    StreamReader reader = new StreamReader(response.GetResponseStream(),
      System.Text.Encoding.Default);//中文文件名
    string line = reader.ReadLine();
    while (line != null)
    {
      result.Append(line);
```

```
            result.Append("\n");
            line = reader.ReadLine();
        }
        // to remove the trailing '\n'
        result.Remove(result.ToString().LastIndexOf('\n'), 1);
        reader.Close();
        response.Close();
        return result.ToString().Split('\n');
    }
    catch (Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.Message);
        downloadFiles = null;
        return downloadFiles;
    }
}
//从 ftp 服务器下载文件的功能
public bool Download(string filePath, string fileName, out string errorinfo)
{
    try
    {
        String onlyFileName = Path.GetFileName(fileName);
        string newFileName = filePath + "\\"+ onlyFileName;
        if (File.Exists(newFileName))
        {
            errorinfo = string.Format(" 本地文件 {0} 已存在, 无法下载", newFileName);
            return false;
        }
        string url = "ftp://"+ ftpServerIP + "/"+ fileName;
        Connect(url);//连接
        reqFTP.Credentials = new NetworkCredential(ftpUserID, ftpPWD);
        FtpWebResponse response = (FtpWebResponse)reqFTP.GetResponse();
        Stream ftpStream = response.GetResponseStream();
        long cl = response.ContentLength;
        int bufferSize = 2048;
        int readCount;
        byte[] buffer = new byte[bufferSize];
        readCount = ftpStream.Read(buffer, 0, bufferSize);
        FileStream outputStream = new FileStream(newFileName, FileMode.Create);
        while (readCount > 0)
```

```
        {
            outputStream.Write(buffer, 0, readCount);
            readCount = ftpStream.Read(buffer, 0, bufferSize);
        }
        ftpStream.Close();
        outputStream.Close();
        response.Close();
        errorinfo = "";
        return true;
    }
    catch (Exception ex)
    {
        errorinfo = string.Format(" 因 {0}, 无法下载", ex.Message);
        return false;
    }
}
//删除文件
public void DeleteFileName(string fileName)
{
    try
    {
        FileInfo fileInf = new FileInfo(fileName);
        string uri = "ftp://"+ ftpServerIP + "/"+ fileInf.Name;
        Connect(uri);//连接
        // 默认为 true，连接不会被关闭
        // 在一个命令之后被执行
        reqFTP.KeepAlive = false;
        // 指定执行什么命令
        reqFTP.Method = WebRequestMethods.Ftp.DeleteFile;
        FtpWebResponse response = (FtpWebResponse)reqFTP.GetResponse();
        response.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, " 删除错误");
    }
}
private void button1_Click(object sender, EventArgs e)
{
    ftpServerIP = textBox1.Text;
```

```
    ftpUserID = textBox2.Text;
    ftpPWD = textBox3.Text;
    string[] theResponse;
    theResponse=GetFileList("ftp://"+ ftpServerIP + "/");
    textBox4.Text = "";
    for (int i = 0; i < theResponse.Length;i++ )
    {
      textBox4.AppendText(theResponse[i]+"\r\n");
    }
  }
```

12.5 实验作业

1. 查看 WebRequestMethods.Ftp 类说明，理解 Ftp 的协议命令。
2. 设计完成文件的上传与下载功能。