FTP Client Class

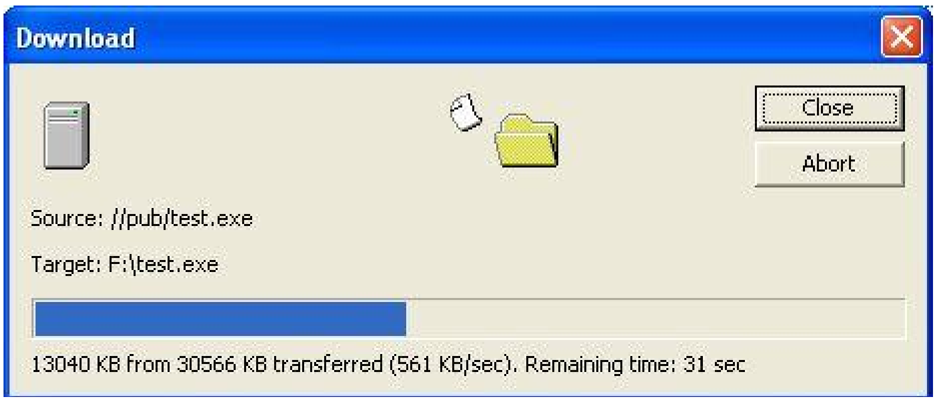
otom, 9 Dec 2012

4.85 (79 votes) Rate:

A non-MFC class to encapsulate the FTP protocol.

Download demo project - 98 Kb

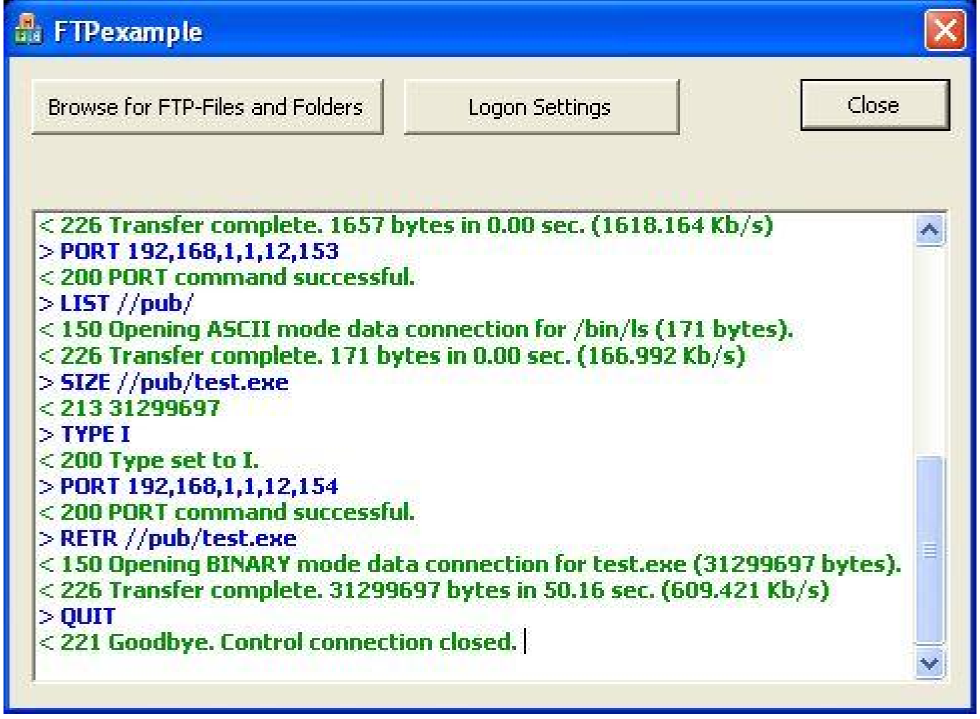
Download source - 51 Kb



Source: //pub/test.exe

Target: F:\test.exe

13040 KB from 30566 KB transferred (561 KB/sec). Remaining time: 31 sec



```
< 226 Transfer complete. 1657 bytes in 0.00 sec. (1618.164 Kb/s)
> PORT 192,168,1,1,12,153
< 200 PORT command successful.
> LIST //pub/
< 150 Opening ASCII mode data connection for /bin/ls (171 bytes).
< 226 Transfer complete. 171 bytes in 0.00 sec. (166.992 Kb/s)
> SIZE //pub/test.exe
< 213 31299697
> TYPE I
< 200 Type set to I.
> PORT 192,168,1,1,12,154
< 200 PORT command successful.
> RETR //pub/test.exe
< 150 Opening BINARY mode data connection for test.exe (31299697 bytes).
< 226 Transfer complete. 31299697 bytes in 50.16 sec. (609.421 Kb/s)
> QUIT
< 221 Goodbye. Control connection closed.
```

Introduction

`CFTPClient` is a class to encapsulate the FTP protocol. I have tried to implement it as platform independent. For the purpose of communication, I have used the classes `CBlockingSocket`, `CSockAddr`, ... from David J. Kruglinski's "Inside Visual C++". These classes are only small wrappers for the sockets-API. Further, I have used a smart pointer-implementation from Scott Meyers "Effective C++, More Effective C++, Effective STL". The implementation of the logon-sequence (with Firewall support) was published in an article on CodeGuru by Phil Anderson. The code for the parsing of different FTP LIST responses is taken from D. J. Bernstein's (parsing code). I only wrapped the C code in a class. I haven't tested the code on other platforms, but I think with little modifications it would compile and run smoothly.

The main features are:

- not based on MFC-sockets,
- not using other MFC-stuffs like `CString` (uses STL),
- supports Firewalls,
- supports resuming,
- supports file eXchange Protocol (FXP) - uses FTP to transfer data directly from one remote server to another (servers must support this feature),
- testet under Windows with Visual Studio 2008,
- testet under Linux (Suse 11.4) with Qt,
- smart pointer implementation can be easily replaced with boost::shared_ptr or std::shared_ptr by defining USE_BOOST_SMART_PTR or USE_STD_SMART_PTR,
- parser which parses the output of the LIST command can be replaced by implementing the interface "IFileListParser",
- can be easily extended.

- `CFTPClient`
  The heart of the application. It accepts a `CLogonInfo` object. Handles the complete communication with the FTP-server like:
  - get directory listing,
  - download/upload files,
  - delete directories/files,
  - walk through directory-tree,
  - passive mode,
  - ...
- `CLogonInfo`
  A simple data structure for logon information, such as host, username, password, firewall, ...
- `CFTPClient::IFileListParser`
  Interface for defining a parser class which can be set in the `CFTPClient` class for parsing the output of the LIST command.
- `CFTPClient::ITransferNotification`
  Implementations of this interface can be used in the Download and Upload methods for controlling the byte streams which are be downloaded/uploaded. This can be used for example to download a file only into memory instead of a local file (see class `COutputStream`).
- `CFTPClient::CNotification`
  The base class for notification mechanism. The class which derives from `CFTPClient::CNotifaction` can be attached to the `CFTPClient` class as an observer. The `CFTPClient` object notifies all the attached observers about the various actions (see example application):

  Hide Shrink ▲ Copy Code

```
void TestFTP()
{
   nsFTP::CFTPClient ftpClient;
   nsFTP::CLogonInfo logonInfo(_T("localhost"), 21, _T("anonymous"),
                                           _T("<a
href="mailto:anonymous@user.com">anonymous@user.com"));

   // connect to server
   ftpClient.Login(logonInfo);

   // get directory listing
   nsFTP::TFTPFileStatusShPtrVec list;
   ftpClient.List(_T("/"), list);

   // iterate listing
   for( nsFTP::TFTPFileStatusShPtrVec::iterator it=list.begin();
                                            it!=list.end(); ++it )
       TRACE(_T("\n%s"), (*it)->Name().c_str());

   // do file operations
   ftpClient.DownloadFile(_T("/pub/test.txt"), _T("c:\\temp\\test.txt"));

   ftpClient.UploadFile(_T("c:\\temp\\test.txt"), _T("/upload/test.txt"));
href="mailto:anonymous@user.com">anonymous@user.com"));

   // connect to server
   ftpClientSource.Login(logonInfoSource);
   ftpClientTarget.Login(logonInfoTarget);


   // do file operations
   nsFTP::CFTPClient::TransferFile(ftpClientSource, _T("/file.txt"),
                                   ftpClientTarget, _T("/newFile.txt"));


   // disconnect
   ftpClientTarget.Logout();
   ftpClientSource.Logout();
}


void TestDownloadAsciiFileIntoTextBuffer()
{
   nsFTP::CFTPClient ftpClientSource;
   nsFTP::CLogonInfo logonInfoSource(_T("sourceftpserver"), 21, _T("anonymous"),
                                                        _T("<a
href="mailto:anonymous@user.com">anonymous@user.com</a>"));

   // connect to server
   ftpClientSource.Login(logonInfoSource);

   nsFTP::COutputStream outputStream(_T("\r\n"), _T("Example"));

   // do file operations
   ftpClientSource.DownloadFile(_T("/file.txt"), outputStream,
                                nsFTP::CRepresentation(nsFTP::CType::ASCII()));

   tstring output = outputStream.GetBuffer();

   // disconnect
   ftpClientSource.Logout();
}
```

History

- 2004-10-25 - First public release.
- 2005-12-04 - Version 1.1
    - Some interfaces changed (e.g. `CNotification`).
    - Bug in `OpenPassiveDataConnection` removed: `SendCommand` was called before data connection was established.
    - Bugs in `GetSingleResponseLine` removed:
        - Infinite loop if the response line doesn't end with CRLF.
        - Return value of `std:string`->find must be checked against `npos`.

    - code modified so it also runs under Linux
    - support for file eXchange Protocol (FXP)

What will be done next

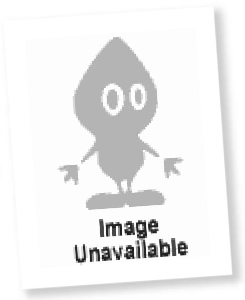
- Example application with Linux GNU-C++.
- New features for FTP client class (for example: copy and delete recursively).
- Unit tests.

License

Share

TWITTER FACEBOOK

About the Author



otom
Software Developer (Senior)
Germany

Follow this Member

No Biography provided

You may also be interested in...

FTP Wanderer - FTP Client using WININET

Get Started Turbo-Charging Your Applications

Comments and Discussions

Add a Comment or Question

Search Comments

First Prev Next

Visual Studio 2017
Member 13581165 18-Dec-17 2:26

Good Job
dennislx 3-Nov-17 11:13

Need crc32 hash support
karaulov 23-Aug-17 1:09

Compiling ERROR
Member 13224876 27-May-17 2:34

can't login every catch
28 9 7-May-17 22:10

Problems compiling this for Pocket PC
Jaime Stuardo - Chile 13-Apr-17 10:36

deprecated stuff
Waaagh 29-Mar-17 3:31

Does not compile; "Library" is garbage
Member 13027086 1-Mar-17 1:03

Re: Does not compile; "Library" is garbage
Member 13020069 16-Mar-17 4:58

Re: Does not compile; "Library" is garbage
TimyFreakkk 4-Apr-17 21:09

Easiest way to get the UP/DOWN Progress in bytes or whatever
redanium 18-Jul-16 3:58

nice lib
louisejackie 16-May-16 11:39

Upload does not work. Do not spend your time on it

upload zip file bugs
lion_117 23-Aug-14 11:24

Re: upload zip file bugs
ztana 24-Dec-14 17:02

Abort
member 6268473 28-Mar-14 6:06

Problem to upload a file by FTP
Member 10675976 18-Mar-14 19:14

Re: Problem to upload a file by FTP
Ashish Tyagi 40 26-Mar-14 17:10

Re: Problem to upload a file by FTP
Member 10675976 26-Mar-14 17:15

Re: Problem to upload a file by FTP
Member 10675976 26-Mar-14 19:19

Re: Problem to upload a file by FTP
Ashish Tyagi 40 27-Mar-14 0:01

Refresh 1 2 3 4 5 6 7 Next »

General News Suggestion Question Bug Answer Joke Praise Rant Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.