|  |  |
| --- | --- |
|  | 🞂基于WinINet的ftpClient实现  上海应用稽术大学 面向对象程序设计 综合实训 课程设计报告 |
|  |  |
|  | 刘柏榕🞂上海应用技术大学 🞂 6/29/17 Thu |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# 概述

此应用程序的目的为，通过调用WinINet Application Interface实现一个较为完全的FTP客户端。

## 开发和编译环境

此Win32应用程序使用喷气大脑公司开发的CLion作为集成开发环境，mingw作为编译器。

# 需求分析

我们注意到有不少人类存在使用Windows环境下的命令行FTP客户端的需求，而Windows自带的ftp.exe的操作方法并不符合部分Unix惯用人类的习惯。因此，我们开发了这个应用程序，旨在为这部分人类提供操作转接，使用Unix中的惯用命令操作FTP客户端。

## 子系统列表

1. ftp作业员
   1. 连接到服务器
   2. 列出目录
   3. 写当前目录
   4. 读当前目录
   5. 下载指定文件
   6. 上传指定文件
   7. 创建目录
   8. 移除目录
   9. 解除文件链接
   10. 重命名文件
   11. 断开连接
   12. 附属通信子系统
2. ftp作业员终端
   1. 用户输入缓冲子系统
   2. 用户命令解析子系统
   3. 用户帮助子系统
   4. 附属通信子系统
3. 选单
4. 辅助子系统

# 子系统原始码和功能说明

1. ftp作业员

这个子系统负责管理ftp连接。

**class** ftpOpt{  
**private**:  
 HINTERNET hInternet;  
 HINTERNET hFtpSession;  
 **bool** ftpOpt\_ftpPassive = **true**;  
 **bool** ftpOpt\_connected = **false**;  
 string ftpOpt\_currHost = "";  
 string ftpOpt\_currUsr = "";  
 string ftpOpt\_currPass = "";  
 **int** ftpOpt\_currPort = 0;  
 string ftpOpt\_filename = "";  
 string ftpOpt\_currDir = "";  
 **int** ftpOpt\_transferMode = FTP\_TRANSFER\_TYPE\_UNKNOWN;  
 **bool** cdHappened = **false**;  
 /\*  
 \* transferMode can be:  
 \* FTP\_TRANSFER\_TYPE\_ASCII  
 \* FTP\_TRANSFER\_TYPE\_BINARY  
 \* FTP\_TRANSFER\_TYPE\_UNKNOWN = FTP\_TRANSFER\_TYPE\_BINARY  
 \*/  
 **int** updateCurrDir();  
**public**:  
 **int** connect(string url, string username = **nullptr**,  
 string password = **nullptr**, **int** port = 21,  
 **bool** ftpPassive = **true**, **bool** keepDir = **false**);  
 **int** ls();  
 **int** cd(string dir);  
 **int** ftpGet(string pRemote, string pLocal,  
 **bool** overwrite = **false**);  
 **int** ftpPut(string pLocal, string pRemote);  
 **int** mkdir(string dir);  
 **int** rm(string file);  
 **int** rmdir(string dir);  
 **int** rename(string oldFile, string newFile);  
 **int** disconnect();  
 string currDir();  
 string currHost();  
 string currUsr();  
 **bool** isConnected();  
};

* 1. 连接到服务器

这个子系统负责连接到指定服务器。

**int** ftpOpt::connect(string url, string username, string password, **int** port, **bool** ftpPassive, **bool** keepDir){  
  
 //Open Internet session  
 hInternet = InternetOpen(**nullptr**, INTERNET\_OPEN\_TYPE\_DIRECT,  
 **nullptr**, **nullptr**, 0);  
 **if**(hInternet == **nullptr**){  
 //On error goto  
 console(\_OOP\_FTPCLIENT\_WININET\_ERROR);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_ERROR;  
 }  
  
 //Attempt to connect  
 hFtpSession = InternetConnect(hInternet, url.c\_str(), (INTERNET\_PORT)port,  
 username.c\_str(), password.c\_str(),  
 INTERNET\_SERVICE\_FTP, ftpPassive?INTERNET\_FLAG\_PASSIVE:0, 0);  
 **if**(hFtpSession == **nullptr**){  
 console(\_OOP\_FTPCLIENT\_WININET\_ERROR);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_ERROR;  
 }  
  
 //OK  
 ftpOpt\_currHost = url;  
 ftpOpt\_currUsr = username;  
 ftpOpt\_currPass = password;  
 ftpOpt\_currPort = port;  
 ftpOpt\_ftpPassive = ftpPassive;  
 ftpOpt\_connected = **true**;  
 **if**(!keepDir){  
 updateCurrDir();  
 }  
 **return** 0;  
}

* 1. 列出目录

这个子系统负责列出当前目录下的所有对象。

**int** ftpOpt::ls(){  
  
 /\*  
 \* List all item in this directory.  
 \* It is VERY complex when we made it with wininet.  
 \*/  
  
 //Define temp container  
 HINTERNET hFind;  
 WIN32\_FIND\_DATA fd;  
 **char** currDir[\_OOP\_FTPCLIENT\_PATH\_SIZE];  
 strcpy(currDir, ftpOpt\_currDir.c\_str());  
  
 //Get file!  
 hFind = FtpFindFirstFile(hFtpSession, currDir, &fd, INTERNET\_FLAG\_RESYNCHRONIZE, 0);  
  
 **if**(hFind != NULL){  
 **do**{  
 console(fd.cFileName);  
 }**while**(InternetFindNextFile(hFind, &fd));  
 }**else**{  
 console(\_OOP\_FTPCLIENT\_WININET\_ERROR);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_ERROR;  
 }  
  
 /\*  
 \* After calling FtpFindFirstFile and until calling  
 \* InternetCloseHandle, the application cannot call  
 \* FtpFindFirstFile again on the given FTP session  
 \* handle. If a call is made to FtpFindFirstFile on  
 \* that handle, the function fails with  
 \* ERROR\_FTP\_TRANSFER\_IN\_PROGRESS. After the calling  
 \* application has finished using the HINTERNET  
 \* handle returned by FtpFindFirstFile, it must be  
 \* closed using the InternetCloseHandle function.  
 \*/  
  
 //Do some clean works (also means reconnect)  
 disconnect();  
 connect(ftpOpt\_currHost, ftpOpt\_currUsr, ftpOpt\_currPass, ftpOpt\_currPort, ftpOpt\_ftpPassive, **true**);  
 cd(ftpOpt\_currDir);  
 cdHappened = **false**;  
  
 **return** 0;  
}

* 1. 写当前目录

写入当前的目录。

**int** ftpOpt::cd(string dir){  
  
 /\*  
 \* Change the current working directory.  
 \* Seems easy than ls xd.  
 \*/  
  
 **char** charDir[\_OOP\_FTPCLIENT\_PATH\_SIZE];  
 strcpy(charDir, dir.c\_str());  
  
 **if**(!FtpSetCurrentDirectory(hFtpSession, charDir)){  
 console(\_OOP\_FTPCLIENT\_WININET\_ERROR);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_ERROR;  
 };  
  
 //We must get current directory anytime.  
 updateCurrDir();  
 **return** 0;  
}

* 1. 读当前目录

从服务器读取当前的目录。

**int** ftpOpt::updateCurrDir(){  
 /\*  
 \* Private function.  
 \* Used for update current directory.  
 \*/  
 **char** tmp[\_OOP\_FTPCLIENT\_PATH\_SIZE];  
 DWORD currDirSize = \_OOP\_FTPCLIENT\_PATH\_SIZE;  
  
 **if**(!FtpGetCurrentDirectory(hFtpSession, tmp, &currDirSize)){  
 console(\_OOP\_FTPCLIENT\_WININET\_ERROR);  
 ftpOpt\_currDir = "undefined";  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_ERROR;  
 }  
  
 ftpOpt\_currDir = tmp;  
 **return** 0;  
}

* 1. 下载指定文件

从服务器下载指定的文件。

**int** ftpOpt::ftpGet(string pRemote, string pLocal, **bool** overwrite){  
  
 /\*  
 \* Get files from remote server.  
 \*/  
  
 **char** pathRemote[\_OOP\_FTPCLIENT\_PATH\_SIZE];  
 **char** pathLocal[\_OOP\_FTPCLIENT\_PATH\_SIZE];  
 strcpy(pathRemote, pRemote.c\_str());  
 strcpy(pathLocal, pLocal.c\_str());  
 **if**(!FtpGetFile(  
 hFtpSession,  
 pathRemote,  
 pathLocal,  
 (BOOL)!overwrite,  
 FILE\_ATTRIBUTE\_ARCHIVE,  
 (DWORD)ftpOpt\_transferMode,  
 0)  
 )  
 {  
 console(\_OOP\_FTPCLIENT\_WININET\_ERROR);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_ERROR;  
 }  
 **return** 0;  
}

* 1. 上传指定文件

向服务器上传指定的文件。

**int** ftpOpt::ftpPut(string pLocal, string pRemote){  
  
 /\*  
 \* Put files to remote server.  
 \*/  
  
 **char** pathRemote[\_OOP\_FTPCLIENT\_PATH\_SIZE];  
 **char** pathLocal[\_OOP\_FTPCLIENT\_PATH\_SIZE];  
 strcpy(pathRemote, pRemote.c\_str());  
 strcpy(pathLocal, pLocal.c\_str());  
 **if**(!FtpPutFile(  
 hFtpSession,  
 pathLocal,  
 pathRemote,  
 (DWORD)ftpOpt\_transferMode,  
 0)  
 )  
 {  
 console(\_OOP\_FTPCLIENT\_WININET\_ERROR);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_ERROR;  
 }  
 **return** 0;  
}

* 1. 创建目录

在服务器上创建目录。

**int** ftpOpt::mkdir(string dir){  
  
 /\*  
 \* Make directory on the remote server.  
 \*/  
  
 **char** dirName[\_OOP\_FTPCLIENT\_PATH\_SIZE];  
 strcpy(dirName, dir.c\_str());  
 **if**(!FtpCreateDirectory(hFtpSession, dirName)){  
 console(\_OOP\_FTPCLIENT\_WININET\_ERROR);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_ERROR;  
 }  
 **return** 0;  
}

* 1. 移除目录

从服务器上移除目录。

**int** ftpOpt::rmdir(string dir){  
  
 /\*  
 \* Remove directory from remote server.  
 \*/  
  
 **char** dirName[\_OOP\_FTPCLIENT\_PATH\_SIZE];  
 strcpy(dirName, dir.c\_str());  
 **if**(!FtpRemoveDirectory(hFtpSession, dirName)){  
 console(\_OOP\_FTPCLIENT\_WININET\_ERROR);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_ERROR;  
 }  
 **return** 0;  
}

* 1. 解除文件链接

解除一个文件的链接。

**int** ftpOpt::rm(string file){  
  
 /\*  
 \* Remove files from remote server.  
 \*/  
  
 **char** fileName[\_OOP\_FTPCLIENT\_PATH\_SIZE];  
 strcpy(fileName, file.c\_str());  
 **if**(!FtpDeleteFile(hFtpSession, fileName)){  
 console(\_OOP\_FTPCLIENT\_WININET\_ERROR);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_ERROR;  
 }  
 **return** 0;  
}

* 1. 重命名文件

重新命名一个文件。

**int** ftpOpt::rename(string oldFile, string newFile){  
  
 /\*  
 \* Rename files on the remote server.  
 \*/  
  
 **char** oldFileName[\_OOP\_FTPCLIENT\_PATH\_SIZE];  
 **char** newFileName[\_OOP\_FTPCLIENT\_PATH\_SIZE];  
 strcpy(oldFileName, oldFile.c\_str());  
 strcpy(newFileName, newFile.c\_str());  
 **if**(!FtpRenameFile(hFtpSession, oldFileName, newFileName)){  
 console(\_OOP\_FTPCLIENT\_WININET\_ERROR);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_ERROR;  
 }  
 **return** 0;  
}

* 1. 断开连接

从服务器断开连接。

**int** ftpOpt::disconnect(){  
  
 /\*  
 \* Disconnect from the server.  
 \*/  
  
 **if**(!(**bool**)InternetCloseHandle(hFtpSession)){  
 console(\_OOP\_FTPCLIENT\_WININET\_ERROR);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_ERROR;  
 }  
 **if**(!(**bool**)InternetCloseHandle(hInternet)){  
 console(\_OOP\_FTPCLIENT\_WININET\_ERROR);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_ERROR;  
 };  
 ftpOpt\_connected = **false**;  
 **return** 0;  
}

* 1. 附属通信子系统

杂项子系统。

string ftpOpt::currDir(){  
 **return** ftpOpt\_currDir;  
}  
  
string ftpOpt::currHost(){  
 **return** ftpOpt\_currHost;  
}  
  
string ftpOpt::currUsr(){  
 **return** ftpOpt\_currUsr;  
}  
  
**bool** ftpOpt::isConnected(){  
 **return** ftpOpt\_connected;  
}

1. ftp作业员终端
2. **class** ftpOptTerminal{  
   **private**:  
    /\*  
    \* vaild command:  
    \* exit  
    \* connect, link  
    \* dc, disconnect  
    \* cd  
    \* ls, dir, ll  
    \* download, get, pull  
    \* upload, put, push  
    \* mkdir  
    \* rm  
    \* rmdir  
    \* rename, mv  
    \* help, ?, man  
    \*/  
    string command;  
    ftpOpt object;  
    **bool** firstLine = **true**;  
   **public**:  
    **int** init(ftpOpt obj);  
    //print dir, return command  
    string wait();  
    //parse usr command  
    /////// Warning: argument unique is not being check. ////////  
    /////// New argument will override old one. ////////  
    **int** parse(string cmdStr);  
     
    //echo help  
    **int** help(string command);  
   };
   1. 用户输入缓冲子系统

接收用户输入。

string ftpOptTerminal::wait(){  
 **if**(!firstLine){  
 cout << endl;  
 }**else**{  
 firstLine = **false**;  
 }  
 **if**(object.isConnected()){  
 console(object.currDir() + ">", **false**);  
 }**else**{  
 console("ftp>", **false**);  
 }  
 **char** str[\_OOP\_FTPCLIENT\_TERMINAL\_SIZE];  
 cin.getline(str,\_OOP\_FTPCLIENT\_TERMINAL\_SIZE);  
 string strObj = str;  
 **return** strObj;  
}

* 1. 用户命令解析子系统

解析用户命令。

**int** ftpOptTerminal::parse(string cmdStr){  
 vector<string> strArray = split(cmdStr, ' ');  
 **if** (strArray[0] == "exit") {  
 **return** \_OOP\_FTPCLIENT\_TERMINAL\_EXIT;  
 }  
 **if** (isElementOf(strArray[0], {"connect", "link"})) {  
 //default argument init  
 string url = "ftp.example.com";  
 **int** port = 21;  
 string username;  
 string password;  
 **bool** ftpPassive = **true**;  
 **for** (**int** i=1;i<strArray.size();i++) {  
 **if** (strArray[i].substr(0, 1) == "-" && i + 1 < strArray.size() && strArray[i+1].substr(0, 1) != "-") {  
 //if it is an argument  
 **if** (isElementOf(strArray[i].substr(1),  
 {"username", "user", "usr", "u"})) {  
 username = strArray[i + 1];  
 } **else if** (isElementOf(strArray[i].substr(1),  
 {"password", "pass", "pw", "p"})) {  
 password = strArray[i + 1];  
 } **else if** (isElementOf(strArray[i].substr(1),  
 {"port", "pt", "v", ":", ";"})) {  
 port = toNumber(strArray[i + 1]);  
 } **else** {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 i++;  
 } **else if** (strArray[i].substr(0, 1) == "-") {  
 **if** (isElementOf(strArray[i].substr(1),  
 {"active"})) {  
 //or it is an option  
 ftpPassive = **false**;  
 } **else** {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 } **else if** (strArray[i].substr(0, 1) != "-"){  
 //it can only be an url  
 **if** (url == "ftp.example.com") {  
 url = strArray[i];  
 } **else** {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 } **else** {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 }  
 console("Sit back and relax when we are attempt to connect to:");  
 console(url);  
 //call this func  
 object.connect(url, username, password, port, ftpPassive);  
 **return** 0;  
 }  
 **if** (isElementOf(strArray[0], {"dc", "disconnect"})) {  
 //Accept no argument  
 **if** (strArray.size() > 1) {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 //call this func here  
 object.disconnect();  
 **return** 0;  
 }  
 **if** (isElementOf(strArray[0], {"cd"})) {  
 //Accept {1,1} string argument  
 **if** (strArray.size() != 2) {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 //default argument init  
 string target\_dir;  
 **if** (strArray[1].substr(0,1) == "-") {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 //it can only be target dir  
 target\_dir = strArray[1];  
 //call this func here  
 object.cd(target\_dir);  
 **return** 0;  
 }  
 **if** (isElementOf(strArray[0], {"ls", "dir", "ll"})) {  
 //Accept no argument  
 **if** (strArray.size() > 1) {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
  
 //call this func here  
 object.ls();  
 **return** 0;  
 }  
 **if** (isElementOf(strArray[0], {"download", "get", "pull"})) {  
 //default argument init  
 string fileRemote;  
 string fileLocal;  
 **bool** overwrite = **false**;  
 **for** (**int** i = 1; i < strArray.size(); i++) {  
 **if** (strArray[i].substr(0, 1) == "-" && i + 1 < strArray.size() && strArray[i + 1].substr(0, 1) != "-") {  
 //if it is an argument  
 **if** (isElementOf(strArray[i].substr(1),  
 {"local", "l"})) {  
 fileLocal = strArray[i + 1];  
 } **else if** (isElementOf(strArray[i].substr(1),  
 {"remote", "r"})) {  
 fileRemote = strArray[i + 1];  
 } **else** {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 i++;  
 } **else if** (strArray[i].substr(0, 1) == "-") {  
 **if** (isElementOf(strArray[i].substr(1),  
 {"overwrite", "o"})) {  
 //or it is an option  
 overwrite = **true**;  
 } **else** {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 } **else if** (strArray[i].substr(0, 1) != "-") {  
 //it can only be file remote and file local  
 **if** (fileRemote == "") {  
 //if fileRemote is not defined  
 fileRemote = strArray[i];  
 } **else if** (fileLocal == "") {  
 //else if fileLocal is not defined  
 fileLocal = strArray[i];  
 } **else** {  
 //wtf is this  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 } **else** {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 }  
 //call this func here  
 object.ftpGet(fileRemote, fileLocal, overwrite);  
 **return** 0;  
 }  
 **if** (isElementOf(strArray[0], {"upload", "put", "push"})) {  
 //default argument init  
 string fileLocal;  
 string fileRemote;  
 **for** (**int** i = 1; i < strArray.size(); i++) {  
 **if** (strArray[i].substr(0, 1) == "-" && i + 1 < strArray.size() && strArray[i + 1].substr(0, 1) != "-") {  
 //if it is an argument  
 **if** (isElementOf(strArray[i].substr(1),  
 {"local", "l"})) {  
 fileLocal = strArray[i + 1];  
 } **else if** (isElementOf(strArray[i].substr(1),  
 {"remote", "r"})) {  
 fileRemote = strArray[i + 1];  
 } **else** {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 i++;  
 } **else if** (strArray[i].substr(0, 1) != "-") {  
 //it can only be file local and file remote  
 **if** (fileLocal == "") {  
 //if fileLocal is not defined  
 fileLocal = strArray[i];  
 } **else if** (fileRemote == "") {  
 //else if fileRemote is not defined  
 fileRemote = strArray[i];  
 } **else** {  
 //wtf is this  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 } **else** {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 }  
 //call this func here  
 object.ftpPut(fileLocal, fileRemote);  
 **return** 0;  
 }  
 **if** (isElementOf(strArray[0], {"mkdir"})) {  
 //Accept {1,1} string argument  
 **if** (strArray.size() != 2) {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 //default argument init  
 string name;  
 **if** (strArray[1].substr(0,1) == "-") {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 //it can only be dir name  
 name = strArray[1];  
 //call this func here  
 object.mkdir(name);  
 **return** 0;  
 }  
 **if** (isElementOf(strArray[0], {"rm"})) {  
 //Accept {1,1} string argument  
 **if** (strArray.size() != 2) {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 //default argument init  
 string name;  
 **if** (strArray[1].substr(0,1) == "-") {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 //it can only be filename  
 name = strArray[1];  
 //call this func here  
 object.rm(name);  
 **return** 0;  
 }  
 **if** (isElementOf(strArray[0], {"rmdir"})) {  
 //Accept {1,1} string argument  
 **if** (strArray.size() != 2) {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 //default argument init  
 string name;  
 **if** (strArray[1].substr(0,1) == "-") {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 //it can only be dir name  
 name = strArray[1];  
 //call this func here  
 object.rmdir(name);  
 **return** 0;  
 }  
 **if** (isElementOf(strArray[0], {"rename", "mv"})) {  
 //default argument init  
 string oldName;  
 string newName;  
 **for** (**int** i = 1; i < strArray.size(); i++) {  
 **if** (strArray[i].substr(0, 1) == "-" && i + 1 < strArray.size() && strArray[i + 1].substr(0, 1) != "-") {  
 //if it is an argument  
 **if** (isElementOf(strArray[i].substr(1),  
 {"old", "o"})) {  
 oldName = strArray[i + 1];  
 } **else if** (isElementOf(strArray[i].substr(1),  
 {"new", "n"})) {  
 newName = strArray[i + 1];  
 } **else** {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 i++;  
 } **else if** (strArray[i].substr(0, 1) != "-") {  
 //it can only be old name and new name  
 **if** (oldName == "") {  
 //if old name is not defined  
 oldName = strArray[i];  
 } **else if** (newName == "") {  
 //else if new name is not defined  
 newName = strArray[i];  
 } **else** {  
 //wtf is this  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 } **else** {  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT;  
 }  
 }  
 //call this func here  
 object.rename(oldName, newName);  
 **return** 0;  
 }  
 **if**(isElementOf(strArray[0], {"man", "?", "help", "？"})){  
 //default argument init  
 string helpStr;  
 **for** (**int** i = 1; i < strArray.size(); i++){  
 helpStr = strArray[i];  
 }  
 //call this func here  
 help(helpStr);  
 **return** 0;  
 }  
 console(\_OOP\_FTPCLIENT\_FTPOPT\_BAD\_COMMAND);  
 **return** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_COMMAND;  
}

* 1. 用户帮助子系统

为用户提供帮助。

**int** ftpOptTerminal::help(string command){  
 **if**(isElementOf(command, {"connect", "link"})){  
 console(  
 "Usage:\n"  
 " connect|link [-port|:|v <port> -username|u <user> -password|p <pass>]\n"  
 "Start a session with a remote ftp server.\n"  
 );  
 }**else if**(isElementOf(command, {"dc", "disconnect"})){  
 console(  
 "Usage:\n"  
 " dc|disconnect\n"  
 "Ends session with the current remote computer.\n"  
 );  
 }**else if**(isElementOf(command, {"cd"})){  
 console(  
 "Usage:\n"  
 " cd <dir>\n"  
 "Sets the current working location to a specified location.\n"  
 );  
 }**else if**(isElementOf(command, {"ls", "dir", "ll"})){  
 console(  
 "Usage:\n"  
 " ls|dir\n"  
 "Gets the items and child items in the current locations.\n"  
 );  
 }**else if**(isElementOf(command, {"download", "get", "pull"})){  
 console(  
 "Usage:\n"  
 " get|pull [-r|remote] <remote\_file> [-l|local] <local\_file> -[o|overwrite]\n"  
 "Gets a specified file from current server.\n"  
 );  
 }**else if**(isElementOf(command, {"upload", "put", "push"})){  
 console(  
 "Usage:\n"  
 " put|push [-l|local] <local\_file> [-r|remote] <remote\_file>\n"  
 "Push a specified file to current server.\n"  
 );  
 }**else if**(isElementOf(command, {"mkdir"})){  
 console(  
 "Usage:\n"  
 " mkdir <dirName>\n"  
 "Creates a new directory on the remote server.\n"  
 );  
 }**else if**(isElementOf(command, {"rmdir"})){  
 console(  
 "Usage:\n"  
 " rmdir <dirName>\n"  
 "Deletes the specified directory on the remote server.\n"  
 );  
 }**else if**(isElementOf(command, {"rm"})){  
 console(  
 "Usage:\n"  
 " rm <fileName>\n"  
 "Deletes the specified file on the remote server.\n"  
 );  
 }**else if**(isElementOf(command, {"rename", "mv"})){  
 console(  
 "Usage:\n"  
 " rename|mv [-o|old] <old\_name> [-n|new] <new\_name>\n"  
 "Rename a file on the remote server.\n"  
 );  
 }**else if**(isElementOf(command, {"man"})){  
 console(  
 "All vaild commands:\n"  
 "man\n"  
 "exit\n"  
 "connect|link\n"  
 "dc|disconnect\n"  
 "cd\n"  
 "ls|ll|dir\n"  
 "download|get|pull\n"  
 "upload|put|push\n"  
 "mkdir\n"  
 "rmdir\n"  
 "rm\n"  
 "rename|mv\n"  
 );  
 }**else if**(command == ""){  
 console(  
 "Which manual page do you want?\n"  
 "Type \"man man\" for all available commands\n"  
 );  
 }**else**{  
 console("Sorry, we have no manual for " + command + ".");  
 }  
}

* 1. 附属通信子系统

提供辅助通讯。

**int** ftpOptTerminal::init(ftpOpt obj){  
 object = obj;  
}

1. 选单

**class** menu{  
**private**:  
 vector<vector<string>> menuSto;  
**public**:  
 **int** print(); //print all objMenu item to screen  
 **int** create(string arg1); //create a objMenu item  
};

**int** menu::print(){  
 **int** maxLength = 0;  
 **for**(**int** i=0;i<menuSto.size();i++){  
 **if**(menuSto[i][0].length() > maxLength){  
 maxLength = menuSto[i][0].length();  
 }  
 }  
 **for**(**int** i=0;i<menuSto.size();i++){  
 cout << i+1 << setfill(' ') << setw(maxLength+4-count(i+1))  
 << menuSto[i][0] << endl;  
 }  
 **return** 0;  
}  
  
**int** menu::create(string arg1){  
 menuSto.push\_back({arg1,"true"});  
 **return** \_OOP\_FTPCLIENT\_UNDEFINED\_ERROR;  
}

1. 辅助子系统

**const int** \_OOP\_FTPCLIENT\_PATH\_SIZE = 4096;  
**const int** \_OOP\_FTPCLIENT\_TERMINAL\_SIZE = 1024;  
  
**const int** \_OOP\_FTPCLIENT\_TERMINAL\_EXIT = 5461654;  
  
//on error CONSTs:  
**const int** \_OOP\_FTPCLIENT\_UNDEFINED\_ERROR = 1577267;  
**const int** \_OOP\_FTPCLIENT\_WININET\_ERROR = 25443;  
**const int** \_OOP\_FTPCLIENT\_FTPOPT\_ERROR = 416541;  
**const int** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_COMMAND = 6548645;  
**const int** \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT = 5616513;  
**const int** \_OOP\_FTPCLIENT\_FTPOPT\_FILEEXIST = 8765456;

**int** count(**int** arg1); //count digital of an integer  
  
**int** console(**int** arg1);  
**int** console(string arg1, **bool** endLine = **true**);  
  
string parse\_wininet\_errno(**int** arg1);  
  
vector<string> split(**const** string& s, **char** seperator);  
  
**bool** isElementOf(string arg1, vector<string> arg2);  
  
**int** toNumber(string str);

**int** count(**int** arg1){  
 **if** (arg1 == 0) {  
 **return** 1;  
 }  
 **return** (**int**)log10((**double**)arg1)+1;  
}  
  
**int** console(**int** arg1) {  
 DWORD err = GetLastError();  
 //https://support.microsoft.com/en-us/help/193625/info-wininet-error-codes-12001-through-12156  
 //https://msdn.microsoft.com/en-us/library/windows/desktop/ms681381(v=vs.85).aspx  
 //00006 0x00000006 ERROR\_INVALID\_HANDLE  
 //12002 0x00002ee2 ERROR\_INTERNET\_TIMEOUT  
 //12007 0x00002ee7 ERROR\_INTERNET\_NAME\_NOT\_RESOLVED  
 //12014 0x00002eee ERROR\_INTERNET\_INCORRECT\_PASSWORD  
 //12015 0x00002eef ERROR\_INTERNET\_LOGIN\_FAILURE  
 //12029 0x00002efd ERROR\_INTERNET\_CANNOT\_CONNECT  
 //12031 0x00002eff ERROR\_INTERNET\_CONNECTION\_RESET  
 //12110 0x00002f4e ERROR\_FTP\_TRANSFER\_IN\_PROGRESS  
 **if** (arg1 == \_OOP\_FTPCLIENT\_WININET\_ERROR) {  
 cout << "Error " << "0x" << hex  
 << setw(8) << setfill('0') << err << " " << parse\_wininet\_errno((**int**) err) << endl;  
 **if** (err == 6) {  
 console("You were not connected.");  
 console("You must connect to a remote host to continue.");  
 } **else if** (err == 12002) {  
 console("Connection timed out.");  
 console("You may try it again later.");  
 } **else if** (err == 12007) {  
 console("The given remote host was not resolved.");  
 } **else if** (err == 12014) {  
 console("Incorrect password.");  
 console("Failed to authorize your session.");  
 } **else if** (err == 12015) {  
 console("Failed to authorize your session.");  
 } **else if** (err == 12029) {  
 console("Your request has been rejected by remote host.");  
 console("Bad argument OR network communicate issue.");  
 } **else if** (err == 12031) {  
 console("Bad argument OR network communicate issue.");  
 } **else if** (err == 12110) {  
 console("Please wait until transfer terminated");  
 } **else** {  
 console("Unknown error. You may contact the author of this application");  
 }  
 } **else if** (arg1 == \_OOP\_FTPCLIENT\_UNDEFINED\_ERROR) {  
 cout << "Error " << "0x" << hex  
 << setw(8) << setfill('0') << arg1 << endl;  
 } **else if** (arg1 == \_OOP\_FTPCLIENT\_FTPOPT\_ERROR) {  
 cout << "Error " << "0x" << hex  
 << setw(8) << setfill('0') << arg1 << endl;  
 } **else if** (arg1 == \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_ARGUMENT) {  
 cout << "Error " << "0x" << hex  
 << setw(8) << setfill('0') << arg1 << endl;  
 console("Bad Argument.");  
 console("Type 'help', 'man' or '?' for user manual.");  
 } **else if** (arg1 == \_OOP\_FTPCLIENT\_FTPOPT\_BAD\_COMMAND) {  
 cout << "Error " << "0x" << hex  
 << setw(8) << setfill('0') << arg1 << endl;  
 console("Bad Command.");  
 console("Type 'help', 'man' or '?' for user manual.");  
 } **else** {  
 cout << arg1 << endl;  
 }  
}  
  
**int** console(string arg1, **bool** endLine){  
 **if**(endLine){  
 cout << arg1 << endl;  
 }**else**{  
 cout << arg1;  
 }  
 **return** 0;  
}  
  
string parse\_wininet\_errno(**int** arg1) {  
 **for**(**int** i=0;i<59;i++){  
 **if**(wininet\_errno[i] == arg1){  
 **return** wininet\_errstr[i];  
 }  
 }  
 **return** "STOP\_UNKNOWN";  
}  
  
vector<string> split(**const** string& s, **char** seperator){  
 vector<string> output;  
  
 string::size\_type prev\_pos = 0, pos = 0;  
  
 **while**((pos = s.find(seperator, pos)) != string::npos)  
 {  
 string substring( s.substr(prev\_pos, pos-prev\_pos) );  
  
 output.push\_back(substring);  
  
 prev\_pos = ++pos;  
 }  
  
 output.push\_back(s.substr(prev\_pos, pos-prev\_pos)); // Last word  
  
 **return** output;  
}  
  
**bool** isElementOf(string arg1, vector<string> arg2){  
 **for**(**int** i=0;i<arg2.size();i++){  
 **if**(arg2[i] == arg1){  
 **return true**;  
 }  
 }  
 **return false**;  
}  
  
**int** toNumber(string str){  
 **return** (**int**)strtol(str.c\_str(), **nullptr**, 0);  
}

//  
// Created by Aozak on 2017/6/22.  
//  
  
#ifndef OOP\_ENDTERM\_FTPCLIENT\_WININET\_ERRNO\_H  
#define OOP\_ENDTERM\_FTPCLIENT\_WININET\_ERRNO\_H  
  
**int** wininet\_errno[] = {  
 12001,12002,12003,12004,12005,12006,12007,12008,  
 12009,12010,12011,12012,12013,12014,12015,12016,  
 12017,12018,12019,12020,12021,12022,12023,12024,  
 12025,12026,12027,12028,12029,12030,12031,12032,  
 12033,12036,12037,12038,12039,12040,12041,12042,  
 12043,12110,12111,12130,12131,12132,12133,12134,  
 12135,12136,12137,12138,12150,12151,12152,12153,  
 12154,12155,12156  
};  
  
string wininet\_errstr[] = {  
 "STOP\_INTERNET\_OUT\_OF\_HANDLES", // 12001,  
 "STOP\_INTERNET\_TIMEOUT", // 12002,  
 "STOP\_INTERNET\_EXTENDED\_ERROR", // 12003,  
 "STOP\_INTERNET\_INTERNAL\_ERROR", // 12004,  
 "STOP\_INTERNET\_INVALID\_URL", // 12005,  
 "STOP\_INTERNET\_UNRECOGNIZED\_SCHEME", // 12006,  
 "STOP\_INTERNET\_NAME\_NOT\_RESOLVED", // 12007,  
 "STOP\_INTERNET\_PROTOCOL\_NOT\_FOUND", // 12008,  
 "STOP\_INTERNET\_INVALID\_OPTION", // 12009,  
 "STOP\_INTERNET\_BAD\_OPTION\_LENGTH", // 12010,  
 "STOP\_INTERNET\_OPTION\_NOT\_SETTABLE", // 12011,  
 "STOP\_INTERNET\_SHUTDOWN", // 12012,  
 "STOP\_INTERNET\_INCORRECT\_USER\_NAME", // 12013,  
 "STOP\_INTERNET\_INCORRECT\_PASSWORD", // 12014,  
 "STOP\_INTERNET\_LOGIN\_FAILURE", // 12015,  
 "STOP\_INTERNET\_INVALID\_OPERATION", // 12016,  
 "STOP\_INTERNET\_OPERATION\_CANCELLED", // 12017,  
 "STOP\_INTERNET\_INCORRECT\_HANDLE\_TYPE", // 12018,  
 "STOP\_INTERNET\_INCORRECT\_HANDLE\_STATE", // 12019,  
 "STOP\_INTERNET\_NOT\_PROXY\_REQUEST", // 12020,  
 "STOP\_INTERNET\_REGISTRY\_VALUE\_NOT\_FOUND", // 12021,  
 "STOP\_INTERNET\_BAD\_REGISTRY\_PARAMETER", // 12022,  
 "STOP\_INTERNET\_NO\_DIRECT\_ACCESS", // 12023,  
 "STOP\_INTERNET\_NO\_CONTEXT", // 12024,  
 "STOP\_INTERNET\_NO\_CALLBACK", // 12025,  
 "STOP\_INTERNET\_REQUEST\_PENDING", // 12026,  
 "STOP\_INTERNET\_INCORRECT\_FORMAT", // 12027,  
 "STOP\_INTERNET\_ITEM\_NOT\_FOUND", // 12028,  
 "STOP\_INTERNET\_CANNOT\_CONNECT", // 12029,  
 "STOP\_INTERNET\_CONNECTION\_ABORTED", // 12030,  
 "STOP\_INTERNET\_CONNECTION\_RESET", // 12031,  
 "STOP\_INTERNET\_FORCE\_RETRY", // 12032,  
 "STOP\_INTERNET\_INVALID\_PROXY\_REQUEST", // 12033,  
 "STOP\_INTERNET\_HANDLE\_EXISTS", // 12036,  
 "STOP\_INTERNET\_SEC\_CERT\_DATE\_INVALID", // 12037,  
 "STOP\_INTERNET\_SEC\_CERT\_CN\_INVALID", // 12038,  
 "STOP\_INTERNET\_HTTP\_TO\_HTTPS\_ON\_REDIR", // 12039,  
 "STOP\_INTERNET\_HTTPS\_TO\_HTTP\_ON\_REDIR", // 12040,  
 "STOP\_INTERNET\_MIXED\_SECURITY", // 12041,  
 "STOP\_INTERNET\_CHG\_POST\_IS\_NON\_SECURE", // 12042,  
 "STOP\_INTERNET\_POST\_IS\_NON\_SECURE", // 12043,  
 "STOP\_FTP\_TRANSFER\_IN\_PROGRESS", // 12110,  
 "STOP\_FTP\_DROPPED", // 12111,  
 "STOP\_GOPHER\_PROTOCOL\_ERROR", // 12130,  
 "STOP\_GOPHER\_NOT\_FILE", // 12131,  
 "STOP\_GOPHER\_DATA\_ERROR", // 12132,  
 "STOP\_GOPHER\_END\_OF\_DATA", // 12133,  
 "STOP\_GOPHER\_INVALID\_LOCATOR", // 12134,  
 "STOP\_GOPHER\_INCORRECT\_LOCATOR\_TYPE", // 12135,  
 "STOP\_GOPHER\_NOT\_GOPHER\_PLUS", // 12136,  
 "STOP\_GOPHER\_ATTRIBUTE\_NOT\_FOUND", // 12137,  
 "STOP\_GOPHER\_UNKNOWN\_LOCATOR", // 12138,  
 "STOP\_HTTP\_HEADER\_NOT\_FOUND", // 12150,  
 "STOP\_HTTP\_DOWNLEVEL\_SERVER", // 12151,  
 "STOP\_HTTP\_INVALID\_SERVER\_RESPONSE", // 12152,  
 "STOP\_HTTP\_INVALID\_HEADER", // 12153,  
 "STOP\_HTTP\_INVALID\_QUERY\_REQUEST", // 12154,  
 "STOP\_HTTP\_HEADER\_ALREADY\_EXISTS", // 12155,  
 "STOP\_HTTP\_REDIRECT\_FAILED" // 12156  
};  
#endif //OOP\_ENDTERM\_FTPCLIENT\_WININET\_ERRNO\_H

# 结束语

此应用程序还需要实现以下功能：

搜索前往雍和宫的步行路线。

# 参考文献

WinINet Reference

https://msdn.microsoft.com/en-us/library/windows/desktop/aa385483(v=vs.85).aspx