# CSS 432
# Final Project: Network Application

**Instructor: Athirai Aravazhi Irissappane**
**Due date: See the syllabus**

---

## 1. Purpose

This is an individual project to implement a network application: an ftp client program based on the Internet Engineering Task Force RFC 959 protocol.

## 2. File Transfer Protocol

The file transfer protocol (ftp) is defined in a 1985 IETF [plain text] document, RFC 959. It is highly recommended that you read this document, and understand the FTP functions (described in Section 4.1), especially:

- `USER`
- `PASS`
- `SYST`
- `PASV`
- `LIST`
- `RETR`
- `STOR`
- `QUIT`

The CSS432 lecture slides for Chapter 9 on FTP may also be useful.

## 3. Statement of Work

### 3.1 Implementation

Design, implement and test an ftp client program that is based on RFC 959 but limited to the commands shown in the table below.

The program should be invoked with **`ftp`** ***`[hostname]`***. If invoked with the optional argument, it will establish a *control* connection to the remote *`hostname`* server using the default port 21 and be able to transfer files over a *data* connection in passive mode. If no argument is given, the client program should not establish any connection until it receives an **`open`** ***`hostname port`*** command (See below).

| **ftp client interface** | **RFC959 command(s)** | **Description** |
|---|---|---|
| `open` *`hostname port`* | N/A | Establish a TCP connection to *`hostname`* on the specified *`port`* |
| `name:` *`user_name`* | `USER` *`user_name`* | Send a *`user_name`* identifier to the server |
| `password:` *`user_password`* | `PASS` *`user_password`* `SYST` | Send the *`user_password`* to the server |
| `cd` *`subdir`* | `CWD` *`subdir`* | Change the server's current working directory to *`subdir`* |

| | | |
|---|---|---|
| `ls` | `PASV`<br>`LIST` | Ask the server to send a list of its current directory contents (through the *data* connection) |
| ~~`get` *`filename`*~~ | `PASV`<br>`RETR` *`filename`* | Get a copy of *`filename`* from the current remote directory on the server (through the *data* connection) |
| `put` *`filename`* | `PASV`<br>`STOR` *`filename`* | Store a copy of *`filename`* into the current remote directory on the server (through the *data* connection) |
| `close` | `QUIT` | Close the connection *but do not quit ftp* |
| `quit` | `QUIT` (if not closed) | Close the connection *and quit ftp* |

Note the server responds to a `PASV` command by sending back a 6-tuple of numbers on the *control* TCP connection (via port 21), in which

- the first four elements represent the **IP address** (using commas instead of periods)
- the last two elements represent the **port number** (which can be computed by mulitplying the 5th element by 256 and adding the 6th element)

specifying where the server will listen for a new *data* TCP connection (via port 20) from the client.

### 3.2 Verification

To test your client program, you should conduct the following verification:

Run your client program on a Linux Lab (uw1-320-lab.uwb.edu) machine and connect to the `ftp.tripod.com` server. Your client program must be able to execute the following ftp operations and commands:

- authentication (using the `css432` *user_name* and *password*)
- `cd`
- `ls`
- `get`
- `put`
- `close`
- `quit`

The *password* for `css432` can be found in the file `/CSSDIV/classes/432/ftp/passwd` on the uw1-320-lab network.

In the sample session below, **prompts** from the ftp client are shown in bold; **user input** is shown in bold blue.

The session starts with the user launching a local executable version of `ftp` - referenced by explicitly including the local directory, **`./ftp`** - rather than the default `ftp` program on the Linux Lab network - which would be invoked if the local directory were not referenced.

```
[css432@uw1-320-20]$ ./ftp ftp.tripod.com
220 Welcome to Tripod FTP.
Name (ftp.tripod.com:css432): css432w17
331 Username set to css432w17. Now enter your password.
Password: *********
230- ============================================================
230-                       IMPORTANT NOTICE
230- ============================================================
230-
230- Powerful building tools. Traffic-generating, money-making
230- programs. It's all waiting for you at the new Tripod.
230-
230-   httpd://www.tripod.com/
```

```
230-
230- ============================================================
230-
230- Tripod offers a cgi-bin to all members! For more
230- information, visit our CGI tutorial.
230-
230-   httpd://www.tripod.lycos.com/build/module_library/tutorial/
230-
230- ============================================================
230-
230- Got a great idea for a website?  Then don't
230- wait, get your own web address today!
230-
230-   httpd://tripod.domains.lycos.com/
230-
230- ============================================================
230-
230- Visit Tripod's Script Library, with free, easy-to-use,
230- customizable CGI and JavaScripts available to all members.
230-
230-   httpd://build.tripod.com/tools/script_library
230-
230- ============================================================
230-
230- We heard you loud and clear! You love your site, but
230- you don't like the ads.  Remove those pesky popups
230- forever with Tripod Plus!
230-
230-   httpd://www.tripod.lycos.com/host/plus.html
230-
230- ============================================================
230 User 'css432w17' logged on.
215 UNIX Type: L8
ftp> ls
227 Entering Passive Mode (209,202,240,80,206,212)
150 Opening ASCII mode data connection for LIST.
drwxr-xr-x   1 css432w17   Tripod          0 Feb 10 21:22 cgi-bin
-rw-r--r--   1 css432w17   Tripod      26169 Feb 10 18:28 ttcp.cpp
-rw-r--r--   1 css432w17   Tripod       8236 Feb 10 21:22 index.htm
drwxr-xr-x   1 css432w7   Tripod          0 Feb 10 18:32 project
226 Transfer complete.
ftp> get ttcp.cpp
227 Entering Passive Mode (209,202,240,80,206,221)
200 Type set to 'I' (IMAGE aka BINARY).
150 Opening BINARY mode data connection for 'ttcp.cpp'.
226 Transfer complete.  (26169 bytes sent.)
ftp> cd project
250 Directory set to '/project'.
ftp> ls
227 Entering Passive Mode (209,202,240,80,206,229)
150 Opening ASCII mode data connection for LIST.
226 Transfer complete.
ftp> put
(local-file) data
(remote-file) NewFile.txt
227 Entering Passive Mode (209,202,240,80,206,248)
200 Type set to 'I' (IMAGE aka BINARY).
150 Opening BINARY mode data connection for 'NewFile.txt'.
226 Transfer complete.  (26169 bytes sent.)
ftp> ls
227 Entering Passive Mode (209,202,240,80,206,252)
150 Opening ASCII mode data connection for LIST.
-rw-r--r--   1 css432   Tripod      26169 Sep 16 18:33 NewFile.txt
226 Transfer complete.
ftp> close
```

```
221 Goodbye...
ftp> quit
[css432@uw1-320-20]$
```

**NOTE: please do not overwrite any files in the `css432w17@ftp.tripod.com` home directory.**

Put any of your files in the `project` subdirectory in order to test your ftp client's `put` command. Your ftp client program may indicate a password on your computer display, although the above example hides the `css432w17` account password with *********.

### 3.3 telnet and the unix ftp

Typing `telnet ftp.tripod.com 21` may help you understand what the server returns to you in response to each command you have typed. Experimenting with the standard distribution ftp on the Linux Lab machines may also aid your understanding of the protocol, and/or help you justify your implementation in terms of execution performance and functionality.

## 4. What to Turn in

Each student must submit the following materials in PDF to CollectIt. In addition, he or she must also turn in all his/her source code to CollectIt. Archive all your source code using the Unix tar command as follows and thereafter upload just one PDF file and one .tar file through CollectIt.

```
If an individual student submits a copy:
   Your first name initial + your last name
   Example: if your name is Mickey Mouse,
            the file name is:     MMouse.tar

To archive all files in the current directory, type:
tar -cvf - * > MMouse.tar
```

Your final project will be graded with the following criteria:

| **Criteria** | **Weight** |
|---|---|
| **Documentation** of your implementations including explanations and illustrations in details | **29pts(29%)** |
| (1) client's interface in terms of OPEN, PASS/SYST, and QUIT | 5pts |
| (2) client's get function in terms of PAV, RETR, and local file options( `O_CREAT \| O_WRONLY` and `S_IRUSR \| S_IWUSR \|S_IRGRP \| S_IROTH`) | 5pts |
| (3) client's cd function (i.e., CWD) | 4pts |
| (4) client's ls function in terms of PASV, LIST, and file output to stdout | 5pts |
| (5) client's put function in terms of PASV, STOR, and local file option (O_RDONLY) | 5pts |
| (6) limitations (how much you complied with RFC 959) | 5pts |
| **Source code** that adheres good modularization, coding style, and an appropriate amount of comments. | **29pts(29%)** |
| (1) the correctness of client's interface (USER/PASS/SYST/QUIT) including authentication | 5pts |
| (2) the correctness of client's get (PASV/RETR) function | 5pts |
| (3) the correctness of client's cd (CWD )function | 4pt |
| (4) the correctness of client's ls (PASV/LIST) function | 5pts |
| (5) the correctness of client's put (PASV/STOR) function | 5pts |
| (6) comments/coding style | 5pts |
| **Execution output** such as snapshots of your display/windows. Type `import -window root X.jpeg; lpr -Puw1-320-p1 X.jpeg` on a uw1-320 linux machine. You don't have to print out all data. But the output must include all execution snapshots of the following execution. | **30pts(30%)** |

| | |
|---|---|
| (1) client's connection establishment | 6pts |
| (2) client's get function | 6pts |
| (3) client's cd function | 6pts |
| (4) client's ls function stdout | 6pts |
| (5) client's put function | 6pts |
| **Performance evaluation** that compares the difference in elapsed time between your ftp client program and the Unix ftp, when executing a GET command. | **4pts(4%)** |
| **Discussions** should be given for possible functional extensions of your own program and/or the Unix ftp. You should also mention about the difference in performance between your own program and the Unix ftp. | **8ts(8%)** |
| **Total** | **100pts(100%)** |
| **Extra credits** will be given if you have implemented addtional ftp commands or functionality. | **5pts(5%)** |

## 5. FAQ

This FAQ page may answer your questions. Click here