CNT 4713 – Net Centric Programming Project 1

FTP Client and FTP Server

Instructor: Francisco R. Ortega

Due Date: See Moodle.

Additional Files and Links: See Moodle.

1. Introduction

This project consists in building an FTP client and FTP server with limited (yet correct) RFC #765 specifications that will allow it to work with your project but also standard client and servers. To obtain full grade, it is important that you complied with the requirements. (RFC: https://www.ietf.org/rfc/rfc959.txt)

1.1. Login to School

School's server: CNT4713.cs.fiu.edu
Login: Your CS account

School's FTP server: CNT4713.cs.fiu.edu (however, you must be on campus to use it or login via another server, for example: ocelot.aul.fiu.edu)
FTP Server Login: classftp   PASSWORD: micarock520

In the school's server you can run Python3 (and Python2), as well as C and C++ using either gcc or clang. It is recommended that you test your final version here. When working on the server, you will need to use different ports to avoid problems with your classmates. These ports will be provided to you in this link, with your initials and ports to be used: goo.gl/E39eRW. Finally, remember that the local loopback in any computer is 127.0.0.1.

1.2. PORTS

When using the school's server, the FTP server works on port 21. However, your FTP server must work in a different port, to avoid conflict with this server and your classmates' servers. The data ports will also be different if working on the server. If you work on your computer, these do not apply to you but you may want to keep the consistency. When you deliver the final submission, they must be using the ports assigned to you.

The default ports must be found in a configuration file (for the server). However, you will notice in the requirements, that you can override this via the command line. Please note that in the school's server, you may not use port 1024 or below.

1.3. About FTP Clients and Servers

The following section provides generic notes about FTP clients and servers.

1.3.1. FTP Clients

As we described in the class, we have two modes in FTP. Active mode leaves the responsibility to the client to open the data channels and pass the PORT command. In passive mode, the server opens the data channels passing the PASV mode. The latter is more common and more convenient. The FTP client code provided to you in class shows active mode. You are required to implement only one more (both will received extra credit).

Additional Notes:

- You can see all the ftp messages by typing debug (which turns on/off debug mode). See : http://users.cs.cf.ac.uk/Dave.Marshall/Internet/node108.html
- You can also turn on/off passive mode.
- Passive mode and Active Mode

1.3.2. About FTP Servers.

We have an FTP server installed for you test your client in cnt4713.cs.fiu.edu. You may want to install an FTP server in your computer. I do recommend using a virtual machine with linux. Lots of people use Ubuntu (which is the one installed cnt4713.cs.fiu.edu), Debian (Ubuntu is Debian derived – In this particular case, I prefer Debian testing branch), and CentOs, among others. You can use a virtual machine in your computer (Virtual Box is free).

You will notice that the FTP server running on CNT4713 is proftpd by running the command in the shell (of cnt4713.cs.fiu.edu) "ps -aux | grep ftp" (without the quotes). If you check the manual (man proftpd), you will gather information useful to you including files that it uses. For example, */etc/proftpd/proftpd.conf* contains the information of the ftp server, which is useful to use it as an example for your FTP server configuration file. You can find additional information about proftpd in http://www.proftpd.org.

1.4. FTP Commands that must be supported

USER
PASS
CWD
CDUP
LOGOUT
RETR
STOR
STOU
APPE
TYPE (supporting at least ASCII and Image – image is binary)
RNFR
RNTO
ABOR (Only required if doing multi-threading ftp client for extra credit)
DELE
RMD
MKD
PWD
LIST
NOOP

PORT (or PASV or Both)

1.5. Languages and Environment.

You may use either Python (Version 3), C++ (suggest to use C++11 or higher using clang), or C (std=c99). This will be graded in the school's sever (CNT4713.cs.fiu.edu). Remember that you may leave processes running on the background (even if you logout) by putting the ampersand (&) next to it. You may also try (this may vary if the server allows you to do) to run it as a daemon (goo.gl/Y4jcdF).

2. FTP Client

A simple implementation (provided by a previous student) is provided to you – using Python 2 (see moodle). This is provided for your convenience but you can write the code from scratch. Note that if you use the code provided – it is provided as is. Therefore you are responsible to make sure that it works and that it conforms to the requirements. Follow the instructions from class to complete the FTP client and the additional information provided in this document. You must follow RFC's specifications and implement the commands in §1.3. It is important that you understand the behavior of the FTP client in the school's linux server (similar to linux or mac) to understand the behavior that is expected from the client. This is critical for your grading. Matching the behavior will provide the highest change of getting full grade on your assignment. This is part of the assignment. You will notice that some commands are not listed in §1.4 but they still need to be implemented – the reason is that some FTP clients commands are aliases to actual RFC commands: for example, ls is the same as List. It is part of your assignment to discover the functionality of the console client to replicate the behavior. Another example is that you may start FTP without a connection. Once you are in the "FTP>" prompt, you can type open (to open a connection), and then it will ask for username and password. If it is incorrect, the user may type again USER. It is your responsibility that the client, with the commands given works with a standard FTP server (either in active or passive mode, or both if you are doing the extra credit). Please also remember that your FTP client is not case sensitive for commands to make it easier but it is case sensitive for the parameters (e.g., filenames).

An additional important requirement is that while you implement the RFC server commands, you must use the common commands used in a ftp client (already mentioned in the previous paragraph). One example is get (or GET as ftp clients commands are not case sensitive). See appendix A.1. for the list of available commands but note that not all of them are needed. Part of your assignment is to investigate which ones you need. For example, binary and ascii changes the mode but the RFC commands are different. Other commands are local (and important), such as lcd, ?, and help.

The program must be called ftp_client and it must be executed as binary (for C or C++) as ftp_client or Python3 ftp_client.

2.1. FTP Client Usage

-h hostname
-u username
-w password
-fp ftp server port
-P passive
-A active
-D debug mode on/off (it actives debug for ftp) ; -d on (on) ; -d off
-V verbose (this may provide additional messages you may want to output); This is designed for your additional output.
-dpr data port range; -dpr 50000-51000
-c configuarion file
-t run test file; -t test2.txt
-T run default test file
-L log_file_name (for log messages)
-ALL log all output to log file (and still display it) log_file_name; -LALL fileall.txt
-ONLY log all output to log file.
-version prints version number of this client.
-info prints student information and additional information about this ftp client.

2.2. FTP Client's Configuration File

The ftp client configuration file will be text based, with the following style.

```
#this is a comment
data_port_max             51000
data_port_min             50000
default_ftp_port          21
default_mode              Passive
default_debug_mode        off
default_verbose_mode      off
default_test_file         test1.txt
default_log_file          ftpclient.log
```

2.3. FTP Client's Test File

The first line of the test file will allow the same parameters as provided by the usage (above)
Each additional line will contain a valid ftp client command.
This will help you to run a system test using multiple clients if needed. One way to run multiple clients is by running in the background (long-running-command &). Other options are possible (creating another python program that runs multiple threads running the test, among others). You can use # for comments.

3. FTP Server

The ftp sever is critical for this assignment. It must be multi-threaded (more about this later). The assignment requires that your server support the RFC commands shown in §1.4. This means any standard client should be able to communicate with your client, not only your client. The following list provides the required components:

- A data file (text) that contains the list of users with: username, password, type (admin/user/notallowed/locked).
    - Your FTP server will allow logins from different users.
    - Admin users can view all the folders.
    - User can only view its folder.
    - Notallowed will get a rejected login (same as locked but a different meaning)
    - Your FTP server will host a private directory for each user.
    - Your FTP will authenticate the users using this data file, which may be already loaded when the ftp server starts.
    - If a user fails to login more than N times within T minutes, the account is locked.
- A configuration file (text) that provides the following information
    - Path for root FTP directory
        - Each user will be under this directory, for example, /user1, /user2,.. Users cannot access the root directory or anyone else's directory. Only the admin can see all of them.
        - Name of the user data file and its location
        - How many times (N) is a user able to try to login with the wrong password for how many (T) times. N should be called
    - Path to default user data file.
    - Specify if this FTP server provides support for Active, Passive, or Both (Mode=Active, Passive, Both). Remember that that client makes this decision. Also remember that you are only required to support one (both for extra credit).
    - Specify data port ranges (for passive mode).
    - Specify port number where this FTP server runs.
    - Specify is the FTP server is enable or disabled.
    - Specify maximum amount of connections before dropping connections with a message. (You also need to define the message in this file)
    - Include a welcome message in this file.
    - Path to log file
- FTP Server must be able to respond to the commands in §1.4
- FTP Server must handle each connection in a separate thread.
- FTP Server has more freedom that then FTP client. However, everything must be well documented and it is expected work with linux ftp client as well.
- FTP Server will have a service port that is only to be used by the admin user. This connection is to simply start and stop the ftp server.
    - FTP Server must response to outside signal to terminate gracefully. For this, you will need to have ftp_server_stop, ftp_server_start (mini python

programs or bash script) that communicate with the server via the service port.

- The FTP Server usage should include (at least)
    - -port Port Number where this ftp will run. Default will be found in configuration file (-port 1000)
    - -configuration configuration file path
    - -max connections
    - -dpr data por range (-dpr 5000-5100) (this is for passive mode only)
    - –userdb user file path
- The default locations for configurations for the ftp are located in the same place that the ftp server is running under a /conf folder. The default ftp root directory is located under /ftproot. Finally, the default location for log files should be /logs.

4. Tests

We will perform at least one system test. Unit testing is encouraged but not required. You must provide test files with two sunny day and rainy day cases for each situation that you believe needs to be tested. For sure, you must cover the commands in §1.4 and the aliases you implemented for the FTP client. The test file will run using your ftp client commands to test with your server. I will use my own test file to test your system. I will provide a sample test file at least a week before the project is due. Everything has to be automated. In addition, you are encouraged to test your system using a test file with the ftp client. In this case, it is fairly easy. You create a file with the commands and then you run ftp < input.test.text or you may want to create a bash script. This is not required but encourage, as I will be testing it like that as well to test your server.

5. Documentation

You are required to provide a readme.me file containing all-important information including how to compile it (if it applies), how to run it, and all the instructions required. In addition, a well-written report including details about this assignment and the learning experience is important. Please see §5.1 for specific information.

5.1. Report Sections and Information.

- Report Name
- Complete name and PantherID
- Introduction: Describe the problem that you are trying to solve and additional study you needed to do. For example, the requirements may be that you needed to read another book or a web site. Provide an insight on the final product of the lab in this section as well.
- Problem Statement: Describe what is the problem you needed to solve
- Methodology: Described how you solved the problem. This must include important information such as socket information and so forth. This may also include the configuration and anything else you created.
- Results: Provide some sample output and provide discussion if needed.

- Analysis: Describe your learning experience, problems you encountered, and anything else that may be useful.
- References: Provide references including sites, repos, books, and any other information you have used for this lab and this report. Use correct citation when doing it.
- Additional Notes:
    - Please be sure to have a well formatter report. Always use "Justify Text" when writing and also create the headings with numbering of them, such as 1. Introduction. You may need to create sub-sections at times.
    - Be clear and concise. Don't add code that you didn't write unless extremely important and don't add your entire code.
- You must hand this report physically in class and uploaded as PDF with your code. Word documents, text files, or other formats will be ignored.
- This report is equally important as your code.

6. Grading Criteria

The following grading criteria will be used:

- FTP Client: 15%
- FTP Server: 50%
- Multi-Threading for FTP Server: 10%
- Test-Cases: 15%
- Report, Documentation, and Others: 10%
- Extra Credit:
    - Support both Passive and Active in FTP Client and Server. 20 points
    - Multi-Thread ftp client: 10 points
    - Additional implementation beyond the requirements: up to 25 additional points (based on my judgment).

A.1. FTP Client commands in Unix

!	dir	mdelete	qc	site
$	disconnect	mdir	sendport	size
account	exit	mget	put	status
append	form	mkdir	pwd	struct
ascii	get	mls	quit	system
bell	glob	mode	quote	sunique
binary	hash	modtime	recv	tenex
bye	help	mput	reget	tick
case	idle	newer	rstatus	trace
cd	image	nmap	rhelp	type
cdup	ipany	nlist	rename	user
chmod	ipv4	ntrans	reset	umask

```
close          ipv6           open           restart        verbose
cr             lcd            prompt         rmdir          ?
delete         ls             passive        runique
debug          macdef         proxy          send
```