CSCI-351 Programming Assignment 1

Due Date: Monday September 29, at 11:59:00pm

Late projects will not be accepted and will result in a 0

Overview

For this project you will write an RFC959 compliant FTP client. The only allowed deviation from the RFC is that your client only needs to provide *file-structure* transfers (directory transfers need not be supported). Your program can provide either a text based and/or graphical user interface.

In this assignment you will write a FTP client, using Java or c#, that will provide the following commands to the user:

Command	Arguments	Description
ascii		Set ASCII transfer type
binary		Set binary transfer type
cd	*path*	Change remote working directory
cdup		Change remote working directory to parent directory (i.e., cd ..)
debug		Toggle debugging mode. The format of the debugging output is not specified. Your output should be readable and give a clear understanding of the actions taken by your client.
dir		List the contents of the remote directory
get	*filename*	Retrieve a file from the remote system
help		Provide help for each command implemented by the client
passive		Toggle passive/active transfer mode
pwd		Print the working directory on the server
quit		Close the connection to the server and terminate the program
user	*user_name*	Specify the user name to be used for the connection. Note that you should query the user for a password. This command will rarely be used due to the prompting of a username at login.

Your program will be named ftp and it will be run from the command line. The program will be written to invoked as shown below:

java FTP server
or
mono FTP server

Where server will be replaced by the network name of the server you wish to connect to, and FTP is the name of your program.

Your program should read commands from the user, execute them, and display messages that indicate their success/progress or failure. You must implement each of the commands in the table above to get full credit. You may, if you wish, implement additional commands. Note any files transferred to/from the server will be placed in/copied from the current working directory.

You may not use existing FTP libraries to complete this project. You must write all of the networking code yourself. In other words, both java and .net provide an extensive FTP set of classes. You MAY NOT use these.

If you decide to add additional functionality (via the command line), it must still be possible to invoke your client as shown above. Your program must check for incorrect usage, and display appropriate messages if the program is invoked incorrectly, or cannot start for some reason. If you add additional command line functionality, you must include a *-help* option that will display command line usage information.

I have written a simple text based stub that you can use in your program if you wish. You can obtain a copy of the program in Java (http://www.cs.rit.edu/~jsb/2141/DataComm/Ftp.java), or the c# version (http://www.cs.rit.edu/~jsb/2141/DataComm/Ftp.cs). This stub is only for you to start with - your code should NOT look like this when you are done!
A sample run of the program is shown below:

```
jsb@localhost > ftp.exe ftp.mozilla.org
Trying 63.245.215.46...
Connected to ftp.mozilla.org.
220-
220-    ftp.mozilla.org / archive.mozilla.org - files are in /pub/mozilla.org
220-
220-    releases.mozilla.org now points to our CDN distribution network and no longe
```

System Information

An FTP server is running on ftp.mozilla.org on the default port. When it asks for a username you can log in with either *anonymous* or *ftp*. You will supply your email address as the password.

Take note that the command in the table above are the commands you need to accept from the user, they are NOT the commands you need to send to the server (in most cases). You will need to review the RFC to figure out the mapping.

It is up to you to figure out how to determine when there is no more data to receive from the server. Examine the response from the server carefully

A note on Passive Mode and data transfers: This will be covered in class, and thus has been excluded from this writeup. Please do not email the instructor asking for clarification on this point.

Style

Write clean, modular, well-structured, and well-documented code. Here are some general guidelines:

1. Make sure that your name appears somewhere in each source file.
2. Each class and every method in the class should have a header that describes what the class/method does.
3. You do not have to document every single line of code you write. Provide enough documentation so someone who is not familiar with your program can figure it out
4. Names of classes, method, constants, and variables should be indicative of their purpose.
5. All literal constants other than trivial ones (e.g. 0 and 1) should be defined symbolically.
6. Define functions as necessary. Keep your code modular and easy to follow.

Submitting

Place all your source files in a single directory and zip the files up (in zip format). Submit the archive to myCourses, in the project 1 folder.

You may include a README file in this directory, if you wish, but no other files are acceptable.

Improperly submitted projects will results in a minimum of one letter grade deduction. All Projects will be tested on the CS Linux machines. You should ensure that your project builds and runs on these machines.

Grading

For full credit, you must mimic the real FTP program as close as possible. While you do not need to implement the same functionality (no upload support, for example), you do need to support active and passive mode, binary and ASCII modes, and all commands in the writeup need to function. Failure to implement any of these commands will result in between 5-10 point off.