Programming Assignment 2 - Writing an ftp client

Computer Networks
Fall 2014
Due: Nov. 11, 2014

Your assignment is to write an interactive ftp client program which implements a subset of the ftp command set. Your program should be able to communicate with ftp servers throughout the Internet. You may use our Ubuntu machine to debug your code. The username and the password is the same as your login username and password. Right now, 172.26.91.120 machine has FTP server.

You are responsible for implementing the following commands:

- quit
- pwd
- put filename
- get filename
- ls
- passive
- active

The program has one command-line argument: the name of a remote server. The client should open a control connection with the server when it starts up. Your code should accommodate both capital and small letter commands.

Remarks

1. The client program can be structured as a loop:

```
while( ) {
   read a command from the keyboard;
   identify the command name;
   carry out the command;
}
```

2. The default mode for transfers is active. The commands passive and active set the mode for subsequent transfers.

3. Reply messages from the server should be displayed on the client terminal. In most cases, that is all that needs to be done. In a few cases, your program may need to read the message. One example occurs when opening a connection. The client sends a user name to the server with the USER command. If the server responds with a 230 message (login successful), the client may begin sending requests immediately. However, if the server responds with a 331 message (password required), the client must send a password with the PASS command to complete the login before sending any requests. In response to the PASS command, the server will send a 230 message (login successful) or a 530 message (login failed).

4. If you're in any doubt as to how a particular command should work, you can use wireshark to log a session using a working ftp client program. The rfc 959 discussed in class also contains detailed information about how each client-server interaction should proceed.

5. All the messages sent from the client to the server should end with a carriage character. You can basically append \x0d and \x0a to the end of the string, e.g. `"USER pcao\x0d\x0a"`. For string operations in C, please refer to http://www.cplusplus.com/reference/clibrary/cstring/ for details.