**CSCI.477 Computer Networking**

**FTP Project – Homework #2 – Final submission**

**Points: 450 Due date: Refer Syllabus**

**Programs for Homework #2**

This is the final homework. Hence, the program must be tested thoroughly testing all ftp commands both valid and invalid cases such as wrong command, wrong number of arguments, invalid filename, etc.

This is a group project. Group will be announced in the class by the instructor at the beginning of the semester. **It is up to the members of the group to come together and implement this homework. Each member of the group must write part of the project documenting clearly who did what portion. At the same time, everyone must know the entire project code**. **There will be a close book exam during the final exam time. The performance on this project will affect your final points for this homework. Those piggyback will lose points. Not everyone in the group will get the same points. One person doing the work and others piggybacking is not allowed.**

For homework#2 of the ftp project, you have to modify both clientftp.c and serverftp.c files from hw#1 by adding appropriate code to implement the ftp commands given below:

**List of FTP Commands to Implement in Homework#2**

**Command Name Syntax**

1. user user username
2. pass pass password
3. quit quit
4. mkdir (or mkd) mkdir directory-name
5. rmdir (or rmd) rmdir directory-name
6. cd (or cwd) cd directory-name
7. dele dele filename
8. pwd pwd
9. ls ls
10. stat (or status) stat
11. help help
12. send send filename
13. recv recv filename

The send command is also known as put command to send file from client ftp to server ftp.

The recv command is also known as get command to get or receive a file from the server to the client.

**NOTE:** Only ASCII mode (text file transfer mode) has to be implemented. In other words, binary mode file transfer is not required to be implemented. The non-text such as binary executable program file transfer mode need not be implemented. The stat command reply message should indicate that the file transfer mode is ASCII (Do not system system call or unix command stat/status to implement ftp status command. Simply send a text stating transfer mode is ASCII from the server ftp).

For format and list of the reply messages sent by the server ftp to client ftp is given in the official RFC ftp document. The link for the document is given in the ftp project Assignment section. Read it and find the correct reply to each ftp command the server must send.

**Editing the Program Files**

You can edit the .c program files either on the UNIX system or on the PC. To edit a file on the UNIX system, use the vi editor. Do not edit any file while the script command is active. It will produce lots of junk output. You will lose points, if you do it. To edit the file on the PC, use Notepad and then transfer it to the UNIX system using ftp that comes with the server like you did in homework#1. If you are editing on the UNIX server, then transfer it to the PC using ftp.

**Backing up Files**

Irrespective of where you edit the files, transfer the files and keep an up-to-date copy on your PC. It is preferred to have two copies, one on the PC and one on a flash drive. This type of backup will help, in case you lose the file or corrupt it or the UNIX server goes down.

**Testing Programs**

This is the final homework. Hence, the program must be tested thoroughly testing all ftp commands. Both valid and invalid cases such as wrong command, wrong number of arguments, invalid filename, etc., must be tested.

First login to UNIX server and change the directory to server. Then issue:

(1) script command to capture session output into a file like you did your homework#1.

NOTE: You can specify an output file name as argument to the script command like:

script serverftpHw2output

(2) pwd

(3) ls –l command to see all the files in the server directory

(4) cat command to display content of file to be sent to the client ftp

(5) Delete any file that has been received from client when you did testing earlier.

(6) run serverftp program.

After serverftp ends (terminates), issue

(7) ls -l command to see files in the directory

(8) cat command to display content of file received from client

(9) exit to stop the script

Login second time to UNIX server, opening a second window and change the directory to client. Then, issue:

1. script command to capture session output.

NOTE: You can specify an output file name as argument to the script command like:

script clientftpHw2output

(2) pwd

(3) ls –l command to see all the files in the server directory

(4) cat command to display content of file to be sent to the server

(5) Delete any file that has been received from client when you did testing earlier.

(6) run clientftp program.

After clientftp ends (terminates), issue

(7) ls -l command to see files in the directory

(8) cat command to display content of file received from server

(9) exit to stop the script

You must test all ftp commands mentioned above in a logical order. Finally, issue the quit command.

You must test all valid and invalid cases like invalid ftp command, invalid arguments for each ftp command.

During testing the following commands, you must issue ls –l command before and after issuing the following ftp commands to ensure that the command worked correctly:

1. mkdir, (2) rmdir, (3) dele, (4) send, (5) recv

to ensure command worked properly.

**IMPORTANT NOTE**: You must test all invalid cases also such as:

1. Invalid file name including non-existing file names
2. Invalid command and misspelled command names
3. Non-implemented command
4. Invalid ftp user
5. Invalid password for the user
6. File operations without logging in. In other words, no file operation ftp commands can be executed by the ftp server without user first logging in. For example, send, recv, mkdir, rmdir, ls, pwd, dele, cd commands an others. The ftp commands that can be issued by a client without logging in are stat, help, and user command to login.

All ftp commands must be tested.

**IMPORTANT NOTE**: Do NOT modified sendMessage and receiveMessage methods given as part of HW#1.

**Capturing Client and Server Output**

In order to capture the output of a program, issue “script” command before running the program. After running the program, that is, after issuing the quit ftp command, issue “exit” command to terminate the script command. The script command will capture everything that appears on the screen into a file from the time it was issued until exit command is issued. On the client ftp side, copy this file into hw2\_clientftp\_output file. On the server ftp side, copy this file into hw2\_serverftp\_output file.

Make sure that you capture the output of both the client and server ftp programs in two separate files.

Finally, transfer the output files and the program files to your PC.

**Items to be Submitted for Homework #2**

Submit the following on or before the due date stated in the syllabus:

1. serverftp.c program
2. clientftp.c program
3. output of the server ftp
4. output of client ftp

**Submit the homework#2 files in electronic form** by emailing the above four files including the captured output of the server and client ftp programs. Files must be attached in the email. Do not submit zip files and on network drive like OneDrive.

Modify homework#2 to implement the following file transfer ftp commands:

1. send (also known as put) command to send file from client ftp to server ftp
2. recv (also known as get) command to get or receive a file from the server to the client.

Only ASCII mode has to be implemented. In other words, binary mode file transfer is not required to be implemented.

**Note on Establishing Data Connection**

The content of the file to be transferred is sent over the data connection. In order to transfer file, a data connection must be established before starting sending file content. The ftp server and the client reverse their role in establishing the data connection for file transfer. In other words, for data connection for both send and recv commands, the client listens for connection request to come from the server for both send and recv commands and the server issues connect request for both send and recv command.

For data connection, use the port number which is one more than the control connection port. That is, if the control connection port is 2000, then the data connection port will be 2001. Define a constant as follows:

#define DATA\_CONNECTION\_PORT 2001

You have to change 2001 to the port which is one more than you control connection port.

To program this, copy the clntConnect function from clientftp.c program to the serverftp.c program. Also, copy svcInitServer function from serverftp.c program to clientftp.c program. Change the port to DATA\_CONNECTION\_PORT. Modify the copied functions to use DATA\_CONNECTION\_PORT.

In order for the client ftp to listen for data connection request from server, call the svcInitServer function after establishing the control connection and before getting into the while loop to read ftp command. Do not close the data connection listen socket until exiting from main. For every file transfer, the client will issue the accept socket function call. The data connection socket will be closed at the end of file transfer by the client.

In order for the server to establish the data connection, after receiving the send or get file transfer command, the server will call clntConnect function. After establishing the connection, it will do the file transfer. After transferring the file content, the server will close the data connection socket. In other words, the data connection socket is opened and closed for each file transfer.

More details will be given in the class including pseudo code for file transfer.

**Testing Server and Client FTP Programs**

In order to test the client and the server program for file transfer, create two files, one for send command and one for get command before running server ftp program and client ftp program. Do not edit the files on the server after starting the server and client programs. The size of the files should be 300 to 500 characters (bytes). Limit the size for sendMessage over data connection to 100 octets (bytes) for file transfer so that 3 to 5 message transfers will occur. You cannot send all the file content in one sendMessage call by reading all the file content once and or creating small files; you will lose points if you do.

**Capturing Client and Server Output**

Follow the steps given earlier to capture the out of send and recv commands. In order to capture the output of a program, issue “script” command before running the program. After running the program, that is, after issuing the quit ftp command, issue “exit” command to terminate the script command. The script command will capture everything that appears on the screen from the time it was issue until exit into a file. On the client ftp side, copy this file into hw3\_clientftp\_output file. On the server ftp side, copy this file into hw3\_serverftp\_output file.

Make sure you capture the output of both the client and server ftp programs.