CEng 436 Data Communications and Networking Homework 2

March 30, 2004

1 Instructions

- This assignment is a programming assignment.
- You are expected to use C/C++ programing language and *native socket library* in your programs.
- All your programs will be tested in a Linux environment.
- You are not allowed to use any high level networking library written for C/C++.
- You are not allowed to use any other programming language.
- You might use Windows socket libraries in development. However you must submit Linux/Unix libraries.
- You may not share with your peers any material about your programs. However you are encouraged to share ideas.
- Submit your source code files, a Makefile specified below and a documentation file specified below.
- Use latex format provided for the documentation file.
- Due date is 12 April 2004 Monday,till midnight.
- Late submission is allowed unit 15 April 2004 Thursday, till midnight. Each day has 10% penalty.
- Submit a softcopy of your homework using https://submit.ceng.metu.edu.tr/.

2 Problem

In this homework you are expected to write a server and a client program for *file transfer*. The architecture should have basic features of FTP protocol, however it will not be an exact implementation of the protocol.

The client would input user commands from a command prompt and send server relevant commands for file transfer and other operations. User commands will be specified, however you will design the protocol between client and server.

3 User commands

put source-file [**destination-file**] transfers the source file to server. If the destination file is specified the file would be renamed.

get source-file [**destination-file**] transfers the source file to client. If the destination file is specified the file would be renamed.

as changes the transfer mode to ASCII.

bi changes the transfer mode to Binary.

nl [**char**] changes the new line character of client to specified value if specified. "*" means blank char. If the character is not specified the new line is character is set to "\n".

bye terminates the client.

4 Specifications

- Any file owerwrite should be prompted to user with possible confirmation answers Y or N. The full path should be printed in the confirmation question.
- The newline character of server might be specified with command line argument -newline.
- The port that the server would listen is specified with command line argument -port.
- If newline character is not specified "\n" character is used. "*" stands for blank character.
- In binary transfer you should transfer the file in bytes. You can use buffering.
- In ascii transfer you should transfer the file in lines. The lines are determined by new line character specified to both parties.

Example

```
$myftpserver -port=6667 -newline=c &
$cat utku.txt
welcome to
the matrix neo
$myftpclient 0 6667
>>>>as
Mode ASCII
>>>>nl a
NewLine a
>>>>put utku.txt deneme.txt
27 bytes transfered in 0 sec. (mode ASCII)
>>>>bye
Come back
$cat deneme.txt
welcome to
the mctrix neo
$
```

Note that the client determines the newline positions to server at transfer and the server inserts its own newline character to these positions.

In real world client and server newline characters are not specified but fixed according the the architecture on which the program works. This technique eliminates the problems due to different newline conventions in Unix and Windows.

- Your implementation would have a lock mechanism on files. If a client wants to read a file that is being written, it should wait until the writing client finishes its work. Similarly if a client want to write to a file that is being read, it should wait until the reading client finishes its work.
- Note that multiple clients may read a file simultaneously.
- Your server should handle up to 10 clients.
- Your clients and servers would be graded on different physical machines so do not make any single processor or single memory assumption.

5 Execution Specifications

You should submit a makefile. *make server* command should build the server with executable name "server", *make client* command should build the client with executable name ”client”. *make all* command should build both. *make clean* command should remove all executables.

Your client should execute as

```
client {hostname} {portno}
```

Example:

```
client buggsbunny.ceng.metu.edu.tr 5567
```

Your server should execute as

```
server {-port=int} [-newline=char]
```

Examples:

```
server -port=5567
server -port=5567 -newline=/
server -newline=9 -port=23
```

6 Documentation

You must submit a short, brief documentation file including issues of

- How you implemented handling multiple clients.
- How you implemented data exchange between handlers of multiple clients.
- How you implemented lock mechanism on files.
- Limitations and bugs of your program, if any.
- Extra features of your program if any. (Extra features will be rewarded if they are significant).