CSC4420 Program 2 Readme

Developer: Mirza Besic

File Descriptions:

1. EchoServer.c -> Server file.
2. EchoClient.c -> Client file.
3. NP.c / NP.h -> Contains useful functions for networking.
4. Makefile -> File used to correctly build the program with the right flags

Makefile Instructions (**compiled on Ubuntu 16.04**):

The makefile is provided in the zip folder. Simply navigate in terminal to the folder location where you unzipped the program files and type in make. This should create a files named **tcpServer, tcpClient** which are the executable programs.

Example: (CSC4420P2 is the folder where the files are unzipped)

```
username@username:~/Desktop/CSC4420P2$ make
```

How to run program:

After the makefile has been ran, files named **tcpServer, tcpClient** will appear in the folder. First run the server, in order to run this file type in ./**tcpServer <port>** into the terminal console.

Example:

```
username@username:~/Desktop/CSC4420P2$ ./tcpServer <port>
```

Next run the client, in order to run this file type in ./**tcpClient <port>** into the terminal console.

Example:

```
username@username:~/Desktop/CSC4420P2$ ./tcpClient <ip address> <port>
```

Design Notes

1. All the commands work ls, !ls, put and get.
    - ls
    - !ls
    - put <input> <output>
    - get <input> <output>
2. Converted to concurrent model using threads instead of processes
3. Checksum works
4. Using thread mutex lock to block out critical region for areas where writing happens
5. Transfer or files work using Readn / Writen and all files complete all the way

Problems, bugs and deficiencies:

General Issues:
The main issue I've had the entire week with is that when I use Readn on either Client or Server, it hangs unless I close the socket. It's infuriating, absolutely humbling at the same time. So basically the behavior is as follows, if I use read instead of Readn then the program runs fine and is able to complete the read without issues. However, then the files don't complete all the way and always send only about half way as read doesn't repeat. When I swap out read with Readn, it hangs and doesn't release the buffer unless I close the socket. I've tried a lot of things but I don't understand the problem fundamentally so I felt like I was flailing around in the dark. I chose having to just reconnect to the server each time as the lesser problem, if the file doesn't complete then it's completely useless.

Known Bugs / Deficiencies:
The client disconnects after each interaction with the server. This is a design flaw that I was not able to overcome. While the server stays open, the client closes after each transaction and has to be restarted with ./tcpClient <ip address> <port>.

Was not able to test on Paris, it didn't work earlier and I have been gone until today. I wanted to test before but I haven't been able to get the read/write to work right until today because of the problem described above. I have tested thoroughly on my local machine, very, very, very thoroughly. Probably ran it over five thousand times easily throughout the past month while testing various steps along the way.

References:

https://stackoverflow.com/questions/1068849/how-do-i-determine-the-number-of-digits-of-an-integer-in-c
http://man7.org/linux/man-pages/man2/shutdown.2.html
https://github.com/arnavsharma93/File-transfer-protocol/blob/master/checksum.c
https://stackoverflow.com/questions/1543466/how-do-i-change-a-tcp-socket-to-be-non-blocking
https://stackoverflow.com/questions/13554150/implementing-the-ls-al-command-in-c

Tokenizer from Project 1
Strip white space from Project 1

Comer, Douglas E., and David L. Stevens. Interworking with TCP/IP Client-Server Programming and applications. Vol. 3. New Jersey: Prentice Hall, 2001. Print. Linux / POSIX Sockets Version. p. 143-160

Stevens, Richard W. UNIX Network Programming. Vol. 1. New Jersey: Prentice Hall, 1990. Print. Prentice Hall Software Ser. p. 83, 465, 521

Stevens, Richard W. Advanced Programming in the UNIX Environment. Indianapolis: Addison-Wesley, 1993. Print. Addison_Wesley Professional Computing Ser. P. 47, 121, 407, 487 - 505