Ameer Khan
ID: 1476214
1/25/19

CMPE 156: Lab 2 Documentation

In order to compile the program, first go to src folder which has the makefile, client.c and server.c. First run make clean, then run make which will create an executable called client and server in the bin folder. In the src folder, run ../bin/server and the following arguments to run the file server.c in one terminal and in another run ../bin/client and the following arguments to run the file client.c in one terminal. In order to use this program, the server must be run first before running the client.

On the server side, the user has to insert one argument which is the port number. On the client side, the user has to insert two arguments. The first argument is the ip address and the second argument is the port number that matches the one entered on the server. Once the arguments have been entered on the client side, it shows the prompt "client $ ", where there is a single space before and after the dollar sign and on the server side, a message saying client is connected is printed. Now you can enter the valid commands. If valid, the client-side outputs the command and on the server side, it prints out the command name entered on the client side. If an invalid command is entered, both the client and server enter a message saying command not found and command invalid respectively. When the user enters the command exit, the client disconnects but the server does not. The server prints a message saying client has disconnected. Now I will talk about error handling done in the client and server programs.

The error checking done on the client side is to check the correct number of arguments input by the user, error checking for opening the socket, error checking for checking if the ip address is valid, error checking for connecting to the server, error checking for writing and reading from the server. The error checking done on the server side is to check the correct number of arguments input by the user, error checking for opening the socket, error checking for binding the socket to the port number, error checking to see if connection is valid with the client, error checking for reading in the socket from the client, error checking for reading in output of command received from client into a file and error checking for writing to the socket from the client.

Now I will explain how this program works. Essentially, there is a three-way handshake between the server and client. The server first binds to the socket and listens for client connections, then the client sends a connection request to the server using the ip address and port number. The server accepts the connection and connects which means the client now is able to write a request to the server. The server reads in the request and writes back to the client and the client reads it and prints out the output. This three-way handshake will continue as long as the client is connected to the server.

This three-way handshake is implemented in a while loop in both my client and server side. In my client file, I print out the client $ prompt and clear the buffer. I have two loops that send a command to the server and reads in the command from the server along with the necessary error handling. I also print out the output of the command onto the screen. I also have a loop so that when the user enters the command exit, the client terminates but the server does not. The server is now looking for a client connection again. In my server file, I clear the buffer and read in the output of the command written by the client into a file using a helpful function called popen. If the command is invalid, an error message pops up on both the client and server side. I then write the ouput of the command requested to the client. I also have a loop so that when the user enters the command exit, the server doesn’t terminate and just try’s to listen for another connection again and accepts if valid.

One shortcoming I had was when I printed out the output of the Unix commands on the client, the commands with longer responses did not print out fully so I might have a buffer overflow problem for some commands.