Elijah Carter
@02777616
10.18.2018
Dr. Li

TCP Programming Project for CSCI 450

GitHub Repository Link:

https://github.com/Elicarter20/TCProgrammingProject

To run program:

client
gcc ./src/client.c -o client
./client <server IP> <server port> <file path> <to format> <to name>

server
gcc server.c translator.c -o server
./server.exe <port>

Program was written in a Unix environment (Unix libraries for socket programming, etc..) in Cygwin for Windows.

Application Protocol

- *Types of messages*
  - There is only one type of message
    - 1 single byte representing a request or response
  - Syntax and semantics of each message type
    - 1 byte messages are read through a *read($socket, $c, 1)* command or written through a *write($socket, $c, 1)* command
  - Format of messages
    [data] - 1 byte
  - Meaning of each field in the format
    - [data]: a single ASCII character
  - Value range of each field
    - **0 to 255** for unsigned

- *Rules of sending messages*
  - Messages must follow the format described above or the server disconnects, and client program ends.
  - Messages are always 1 byte
  - Messages sending and receiving follows the algorithm laid out below
  - Both client and server send same message type (1 single byte)

Usage of Client and Server

Running the server hosts it on a continuous loop until a client connects to it. After a client disconnects(success case) or the server shuts down the port (error case), the server waits for another connection. The server must be brought down manually. The general algorithm for message sending is as follows:

1. Client sends a primer message to check server status (‘\x2’)
   Client waits for ACK from server (‘\x3’). If no ACK received, the end program.

   Server waits for primer message. If is received, then send ACK (‘\x3’)
   Else, close connection.

2. Once receives ACK for primer, client computes number of chars in the file.
   Reads the limit of digits in length from server in 1 byte from server.
   Once received, if input file is within limits, send ACK(‘\x3’).
   Else, send (‘\x4’) and end program.

   Server sends max digits in the length of a file (max == 4) in 1 byte
   Server waits for ACK(‘\x3’) from client that input file is within limits.
   If not, close connection

3. Client sends each char in the input file as 1 byte to server.
   After each byte, client waits for ACK from server (‘\x1’) before sending another
   If any byte/char is not ACKed, end program**.
   Once file is read, send FIN message (‘\x4’)

   Server waits for 1 byte from client, representing a char in the input file.
   Once byte is received, send ACK for byte (‘\x3’)
   If received FIN byte (‘\x4’), stop reading.

4. Client waits for ACK of file status. If ACK(‘\x5’), continue.
   If REJ(‘\x6’), ends program.

   Server runs file contents through check_file() function to validate its formatting.
   If validated, send ACK byte (‘\x5’). Else, send REJ byte (‘\x6’)

5. Client concatenates cmd line arguments for translation type and target file path
   Client sends arguments 1 byte at a time, ending with FIN msg (‘\x3’)
   Client waits for ACK (‘\x5’) or REJ (‘\x6’). If ACK, continue. Else, end program

   Server reads 1 byte at a time representing concatenated cmd line arguments
   Once read FIN msg (‘\x3’), validates the received arguments.
   If validated, sends ACK (‘\x5’). Else, send REJ (‘\x6’) and close connection.

6. Client waits for ACK (‘\x2’). If received, print (“Success”)
   If receive REJ (‘\x3’), end program.

   Server does translation and attempts to write to target file path.
   If write is successful, send ACK (‘\x2’) and closes connection.
   Else, send REJ (‘\x3’) and close connection

** if byte is not acknowledged, end communication and close socket (no resending)

Test Cases

See testcases folder in repository for examples. Basing test cases off 3 fields...

IF = input file **T** = translation type **RF** = target file

For all tests: server is open at **port 8090**

client connects to **localhost 127.0.0.1** at **port 8090**

Case	**Input**	**Expected Output**	**Result**
Client input file does not exist.	**IF** = *nonexistent file* **T** = “0” **RF** = “target.txt”	**Server:** Sent Rejection **Client:** Format Error	**Server:** Sent Rejection **Client:** Format Error
Client input file is empty.	**IF** = “empty.txt” **T** = “0” **RF** = “target.txt”	**Server:** Sent Rejection **Client:** Format Error	**Server:** Sent Rejection **Client:** Format Error
Client provides incorrect translation type format.	**IF** = “bothtypes” **T** = “00”,”a”,”4” **RF** = “target.txt”	**Server:** Sent Rejection **Client:** Format Error	**Server:** Sent Rejection **Client:** Format Error
File has only type 0, with “to format “ 0	**IF** = “alltypezero” **T** = “0” **RF** = “target.txt”	**Server:** Sent Confirmation and wrote: 00 02 0102 0304 00 03 091a bcde 0001 00 01 0000 00 02 003b 015b 00 01 0384 00 02 ffff 0000 **Client:** Success	**Server:** Sent Confirmation and wrote: 00020102030400030 91abcde0001000100 000002003b015b000 103840002ffff0000 **Client:** Success
File has only type 0, with “to format “ 1	**IF** = “alltypezero” **T** = “1” **RF** = “target.txt”	**Server:** Sent Confirmation and wrote: 01 303032 323538 2c 373732 01 303032 3539 2c 333437 01 303031 393030 01 303033 32333330 2c 3438333530 2c 31 01 303031 30 01 303032 3635353335 2c 30 **Client:** Success	013030323235382c3 73732013030333233 33302c34383335302 c3101303031300130 303235392c3334370 13030313930300130 303236353533352c3 0 **Client:** Success

File has only type 0, with “to format “ 2	**IF** = “alltypezero” **T** = “2” **RF** = “target.txt”	**Server:** Sent Confirmation and wrote: 00 02 0102 0304 00 03 091a bcde 0001 0001 0000 00 02 003b 015b 00 01 0384 00 02 ffff 0000 **Client:** Success	**Server:** Sent Confirmation and wrote: 00020102030400030 91abcde0001000100 000002003b015b000 103840002ffff0000 **Client:** Success
File has only type 0, with “to format “ 3	**IF** = “alltypezero” **T** = “3” **RF** = “target.txt”	**Server:** Sent Confirmation and wrote: 01 303032 323538 2c 373732 01 303032 3539 2c 333437 01 303031 393030 01 303033 32333330 2c 3438333530 2c 31 01 303031 30 01 303032 3635353335 2c 30 **Client:** Success	**Server:** Sent Confirmation and wrote: 013030323235382c3 73732013030333233 33302c34383335302 c3101303031300130 303235392c3334370 13030313930300130 303236353533352c3 0 **Client:** Success
File has only type 1, with “to format “ 0	**IF** = “alltypeone” **T** = “0” **RF** = “target.txt”	**Server:** Sent Confirmation and wrote: 01 303032 323538 2c 373732 01 303032 3539 2c 333437 01 303031 393030 01 303033 32333330 2c 3438333530 2c 31 01 303031 30 01 303032 3635353335 2c 30 **Client:** Success	**Server:** Sent Confirmation and wrote: 013030323235382c3 73732013030323539 2c333437013030313 93030013030333233 33302c34383335302 c3101303031300130 303236353533352c3 0 **Client:** Success
File has only type 1, with “to format “ 1	**IF** = “alltypeone” **T** = “1” **RF** = “target.txt”	**Server:** Sent Confirmation and wrote: 01 303032 323538 2c 373732 01 303032 3539 2c	**Server:** Sent Confirmation and wrote: 013030323235382c3 73732013030323539

		333437 01 303031 393030 01 303033 32333330 2c 3438333530 2c 31 01 303031 30 01 303032 3635353335 2c 30 **Client:** Success	2c333437013030313 93030013030333233 33302c34383335302 c3101303031300130 303236353533352c3 0 **Client:** Success
File has only type 1, with “to format “ 2	**IF** = “alltypeone” **T** = “2” **RF** = “target.txt”	**Server:** Sent Confirmation and wrote: 00 02 0102 0304 00 03 091a bcde 0001 0001 0000 00 02 003b 015b 00 01 0384 00 02 ffff 0000 **Client:** Success	**Server:** Sent Confirmation and wrote: 00020102030400020 03b015b0001038400 03091abcde0001000 100000002ffff0000 **Client:** Success
File has only type 1, with “to format “ 3	**IF** = “alltypeone” **T** = “3” **RF** = “target.txt”	**Server:** Sent Confirmation and wrote: 00 02 0102 0304 00 03 091a bcde 0001 0001 0000 00 02 003b 015b 00 01 0384 00 02 ffff 0000 **Client:** Success	**Server:** Sent Confirmation and wrote: 00020102030400020 03b015b0001038400 03091abcde0001000 100000002ffff0000 **Client:** Success
File has both type 0 and 1, with “to format” 0	**IF** = “bothtypes” **T** = “0” **RF** = “target.txt”	**Server:** Sent Confirmation and wrote: 00 02 0102 0304 00 03 091a bcde 0001 0001 0000 00 02 003b 015b 00 01 0384 00 02 ffff 0000 01 303032 323538 2c 373732 01 303032 3539 2c 333437 01 303031 393030 01 303033 32333330 2c 3438333530 2c 31	**Server:** Sent Confirmation and wrote: 00020102030400030 91abcde0001000100 000002003b015b000 103840002ffff000001 3030323235382c373 7320130303235392c 33343701303031393 03001303033323333 302c34383335302c3 10130303130013030 3236353533352c30 **Client:** Success

		01 303031 30 01 303032 3635353335 2c 30 **Client:** Success	
File has both type 0 and 1, with “to format” 1	**IF** = “bothtypes” **T** = “1” **RF** = “target.txt”	**Server:** Sent Confirmation and wrote: 01 303032 323538 2c 373732 01 303032 3539 2c 333437 01 303031 393030 01 303033 32333330 2c 3438333530 2c 31 01 303031 30 01 303032 3635353335 2c 30 01 303032 323538 2c 373732 01 303032 3539 2c 333437 01 303031 393030 01 303033 32333330 2c 3438333530 2c 31 01 303031 30 01 303032 3635353335 2c 30 **Client:** Success	**Server:** Sent Confirmation and wrote: 013030323235382c3 73732013030333233 33302c34383335302 c3101303031300130 303235392c3334370 13030313930300130 303236353533352c3 0013030323235382c 37373201303032353 92c33343701303031 39303001303033323 333302c3438333530 2c310130303130013 0303236353533352c 30 **Client:** Success
File has both type 0 and 1, with “to format” 2	**IF** = “bothtypes” **T** = “2” **RF** = “target.txt”	**Server:** Sent Confirmation and wrote: 00 02 0102 0304 00 03 091a bcde 0001 0001 0000 00 02 003b 015b 00 01 0384 00 02 ffff 0000 00 02 0102 0304 00 03 091a bcde 0001 0001 0000 00 02 003b 015b 00 01 0384 00 02 ffff 0000 **Client:** Success	**Server:** Sent Confirmation and wrote: 00020102030400030 91abcde0001000100 000002003b015b000 103840002ffff000000 02010203040002003 b015b000103840003 091abcde000100010 0000002ffff0000 **Client:** Success

File has both type 0 and 1, with “to format” 3	**IF** = “bothtypes” **T** = “3” **RF** = “target.txt”	**Server:** Sent Confirmation and wrote: 01 303032 323538 2c 373732 01 303032 3539 2c 333437 01 303031 393030 01 303033 32333330 2c 3438333530 2c 31 01 303031 30 01 303032 3635353335 2c 30 00 02 0102 0304 00 03 091a bcde 0001 0001 0000 00 02 003b 015b 00 01 0384 00 02 ffff 0000 **Client:** Success	**Server:** Sent Confirmation and wrote: 013030323235382c373732013030333233333302c34383335302c3101303031300130303235392c3334370130303139303001303030236353533352c300002010203040002003b015b000103840003091abcde0001000100000002ffff0000 **Client:** Success
File has type 0 with errors	**IF** = “typezeroerror” **T** = “0” **RF** = “target.txt”	**Server:** Sent Rejection **Client:** Format Error	**Server:** Sent Rejection **Client:** Format Error
File has type 1 with errors	**IF** = “typeoneerror” **T** = “0” **RF** = “target.txt”	**Server:** Sent Rejection **Client:** Format Error	**Server:** Sent Rejection **Client:** Format Error
File has type 0 and 1 with errors in type 0	**IF** = “botherrtypezero” **T** = “0” **RF** = “target.txt”	**Server:** Sent Rejection **Client:** Format Error	**Server:** Sent Rejection **Client:** Format Error
File has type 0 and 1 with errors in type 1	**IF** = “botherrtypeone” **T** = “0” **RF** = “target.txt”	**Server:** Sent Rejection **Client:** Format Error	**Server:** Sent Rejection **Client:** Format Error
Two clients sending to server one by one	Did not write script to test, but performed manually.	**Server:** Sent 2 successive or Confirmations or Rejection **Client:** Format Error or Success x2	**Server:** Sent 2 successive or Confirmations or Rejection **Client:** Format Error or Success x2
Ten clients sending to	Did not write script to test, but performed	**Server:** Sent 10 successive or	**Server:** Sent 10 successive or

server one by one	manually.	Confirmations or Rejection **Client:** Format Error or Success x10	Confirmations or Rejection **Client:** Format Error or Success x10

Code Bases Used

integer to binary converter
https://www.quora.com/Is-there-a-function-in-C-that-converts-an-integer-into-bits

starting code for server and client
https://www.geeksforgeeks.org/socket-programming-cc/

References Used

google search and stack overflow
for string operations, message sending, etc..

Known Problems

None known, besides all the edge cases of file formatting. Likewise, placing more limits on file size would be better. Also, to create a more elaborate message format.