CS341 Project #0 Report

20170436 Chanwoo Yun

- Client

At first, client program makes socket by socket(). Then, it parses arguments by checking argv[i] is one of '-h', '-p', '-o', '-s'. If argv[i] is one of them, argv[i+1] has information about server address, port, op, and shift.

If shift value is bigger than 26 or minus value, client needs to adjust the value between 0~26.

Second, the client tries to connect with server by using information have been parsed.

Then, it gets input from stdin and saves it in buf, 10000000-10 times of until it gets EOF. After that, it makes protocol. The length of protocol is strlen(buf)+8, because sizeof(protocol) is 8. At that time, checksum value is 0. Then, saves protocol and input string together in packet.

Next, calculate checksum value by using checksum2() function. Then, re-save protocol containing checksum value in 'packet'. Then, write it to server.

Next, read data from server. At first, repeat reading until read more than 8 bytes, which is length of protocol. Then, client can know the length of packet server sent. And, repeat reading until read 'protocol.length' bytes. If all bytes read, make protocol.checksum 0, and check the result of checksum2() is same with the value in protocol. If two values are same, then print 'packet+8' because there is a protocol in first 8 bytes.

Finally, repeat these procedures until there are no remainder inputs in stdin.

- Server

At first, client program makes socket by socket(). Then, it parses arguments by checking argv[i] is '-p'. If argv[i] is '-p', argv[i+1] has information about server port. And, bind socket to address 'INADDR_ANY', which means the address of itself. And, listen.

After client is connected, make child process to deal with the client. Parent process closes connection to the client and wait of other connection.

Child process read data from client. At first, repeat reading until read more than 8 bytes, which is length of protocol. Then, server can know the length of packet client sent. And, repeat reading until read 'protocol.length' bytes. If all bytes read, make protocol.checksum 0, and check the result of checksum2() is same with the value in protocol. If two values are same, then encode or decode the input. After decoding, re-check the checksum (make sure of the value of checksum is 0 while checking) and write it to the client.

Finally, repeat these procedures until there are no remainder data from client.

During the project, I ckecked the code in these sites.

To know the way how to use socket function.

https://luckyyowu.tistory.com/81

https://luckyyowu.tistory.com/88

Use checksum2() function in this site.

https://locklessinc.com/articles/tcp_checksum/