CS341 Project #0 – Socket Programming

20170171 Jihwan Kim

Project #0 - Socket Programming Document

In this document, I will explain some key points of my implementation.

1. Common (both client.c and server.c)
   - uint16_t unpack(const char *ptr)

     It converts continuous two bytes to byte-ordered unsigned short. Just simple shift operation.

   - uint32_t read_n(int sockfd, char *out, uint32_t length)

     Try to read length bytes from socket #sockfd. Return 0 if there is an error while read or socket was closed, return 0, else amount actually read (must be same as length). Data read from socket will be saved in buf

   - uint16_t checksum(char op, char shift, uint32_t length, uint32_t buf_size, const char *buf)

     Calculate a checksum from given parameters. length is whole length of message, which must be same as buf_size + 8. buf_size is a length of user's input. It unpacks bytes to unsigned short and perform roll-over after every addition.

   - uint32_t build_packet(char **out, char op, char shift, uint32_t payload_length, const char *payload)

     From given op, shift, payload, make a packet of our protocol, and returns the size of whole message. Generated message will be allocated dynamically in char[payload_length + 8], and out will point address of generated message.

   - stderr and VERBOSE

     If something is wrong, both client and server prints error message in stderr. After print stderr, client will be quit, server will be closed socket connection unless it fails opening socket at initial. If then, server will be closed. Also, my implementation supports one more extra argument in command line, -v to enable verbose mode to show some log. Logs are printed in stderr, to prevent logs are printed in stdout, which will be compared with reference data.

2. Only in client

Client checks some arguments before establish socket connection. Client will be terminated before socket connection when required arguments (host, port, op, shift) are missing, port is lower than 10, or port is greater than 65535, or op is not neither 0 nor 1. It preprocesses shift into [0, 25] by modular operations. If shift is negative, then will be added by 26 to fit them into [0, 25]. It is mandatory because shift field will be considered as uint8_t in server side, we must convert it to get proper result

3. Only in server

   Server only checks validity of argument '-p'. If port is lower than 0 or greater than 65535, it will be rejected.

   - while(1) loop

     To listen socket forever without external signals, accept() will be called in while(1) loop forever.

   - fork()

     If connection established and file descriptor is opened, it forks and parent continues to while(1) loop, and child call process function which actually handles the protocol. After communication is over, it closes the file descriptor and is exited.

   - int process(int connfd)

     Read and parse protocol message, and sent back result string to client. It returns a length of "string length" when request is valid, else -1. Caesar encryption/decryption is performed by str_crypt(char *, int, size) function. If it's decryption, negate it to perform same logic as encryption.

   - int str_crypt(char *buf, int key, int size)

     Perform caesar cryptographic process in buf which has 'size' length. It shifts 'key' amount in buf, and returns how many bytes are processed. Return value always same as size. For performance, when key is zero, just return it.

4. Others

   - Makefile

     It compiles with -W -Wall -Werror options. 'client', 'server', 'all', 'clean' is avail options.