CS341A Fall 2018

Project #1: Socket Programming

20170567 Jeon Sungwoo

1. Instruction to compile the programs

$make all: make server & client program by client.c and server.c

$make clean: delete the server & client program

```
CC = gcc

all: server client
clobber: clean
	rm -f *~ \
clean:
	rm -f server *.o
	rm -f client *.o

server: server.c
	$(CC) server.c -o server
client: client.c
	$(CC) client.c -o client

```

2. Self-test results of client & server

-Comparing the output between test server and my server with client

```
sungwoo@ubuntu:~$ ./client -h 143.248.53.25 -p 60000 -o 0 -k cake < a.txt > test
.txt
sungwoo@ubuntu:~$ ./client -h 127.0.0.1 -p 10000 -o 0 -k cake < a.txt > b.txt
sungwoo@ubuntu:~$ diff -i test.txt b.txt
sungwoo@ubuntu:~$
```

File Edit Selection Find View Goto Tools Project Preferences Help

a.txt

```
dsakfjhlkjvfkjhsdakjv
aslvkjabsvlkbs

asflkajsvlkasjbvlkfsbv
favjfvlkdjabvkljfvb


afvljkabfvklajbvfkljbfv
afvjklbvfkljbfvba

adfvjfvkljbfvjklbafv
```

File Edit Selection Find View Goto Tools Project Preferences Help

b.txt

```
fskohjrpmjfjmjrwfaunx
acpxktedsfpmbc

eufvocjcznkkwlbfpmfcfx
fkzlffpmdtedvuplfff


cffplkkfhvupcjlzhkvndff
ehvtonbfjmltfhvle

cdpzlffonjljxjupdapz
```

File Edit Selection Find View Goto Tools Project Preferences Help

test.txt

```
fskohjrpmjfjmjrwfaunx
acpxktedsfpmbc

eufvocjcznkkwlbfpmfcfx
fkzlffpmdtedvuplfff


cffplkkfhvupcjlzhkvndff
ehvtonbfjmltfhvle

cdpzlffonjljxjupdapz
```

-Comparing the encrypt and decrypt data with my server & client

```
sungwoo@ubuntu:~$ ./client -h 127.0.0.1 -p 10000 -o 0 -k cake < a.txt > b.txt
sungwoo@ubuntu:~$ ./client -h 127.0.0.1 -p 10000 -o 1 -k cake < b.txt > test.txt
sungwoo@ubuntu:~$ diff a.txt test.txt
sungwoo@ubuntu:~$
```

3. Structure of client

Packet: I make struct packet to send with protocol. In packet, there is 5 elements: operator, checksum, keyword, length, and the pointer of data array. For operator and length, I change them from host-order to network-order. And I allocate the memory for data in packet and struct packet.

```
14  // define protocol packet
15  struct packet {
16      unsigned short op;
17      unsigned short checksum;
18      char keyword[4];
19      unsigned long long length;
20      char* data;
21  };
```

And, I use <endian.h> to change the network-order

Open_clientfd: I reference this function from 2018 cs230 class17 ppt.

```
25  // establish a connection with a server
26  // from 2018 cs230 class17 ppt
27  int open_clientfd(char *hostname, char *port) {
28    int clientfd;
29    struct addrinfo hints, *listp, *p;
30
31    /* Get a list of potential server addresses */
32    memset(&hints, 0, sizeof(struct addrinfo));
33    hints.ai_socktype = SOCK_STREAM;  /* Open a connection */
34    hints.ai_flags = AI_NUMERICSERV;  /* …using numeric port arg. */
35    hints.ai_flags |= AI_ADDRCONFIG;  /* Recommended for connections */
36    getaddrinfo(hostname, port, &hints, &listp);
37      /* Walk the list for one that we can successfully connect to */
38      for (p = listp; p; p = p->ai_next) {
39          /* Create a socket descriptor */
40          if ((clientfd = socket(p->ai_family, p->ai_socktype,
41                                 p->ai_protocol)) < 0)
42              continue; /* Socket failed, try the next */
43
44          /* Connect to the server */
45          if (connect(clientfd, p->ai_addr, p->ai_addrlen) != -1)
46              break; /* Success */
47          close(clientfd); /* Connect failed, try another */
48      }
49
50      /* Clean up */
51      freeaddrinfo(listp);
52      if (!p) /* All connects failed */
53          return -1;
54      else    /* The last connect succeeded */
55          return clientfd;
56  }
```

Checksum: I also reference this function from http://locklessinc.com/articles/tcp_checksum/. However, I used the packet to send the data, and there is a pointer for data array; so, I add some code to calculate the checksum by pointer.

```
65      // calculate the checksum for packet without data
66      for(int i = 0; i<2; i++){
67          unsigned long long s = *b++;
68          sum += s;
69          if (sum < s) sum ++;
70          size -= 8;
71      }
72      // calculate the checksum for data in packet
73      b = (unsigned long long *) ((packet_T)buf)->data;
```

Main: First, I allocate the memory for packet, and dedicate each argument to packet. And then, I connect the client with server by open_clientfd function; and, calculate the data length and checksum. Lastly, I send the whole data by packet http://locklessinc.com/articles/tcp_checksum/to server; and, receive the encrypted/decrypted data, print it depend on checksum.

4. Structure of server

Packet: As I mentioned at structure of client, I make struct packet to send with protocol same way.

Checksum: I also reference this function from http://locklessinc.com/articles/tcp_checksum/. And, I add some code to calculate the checksum by pointer.

```
65      // calculate the checksum for packet without data
66      for(int i = 0; i<2; i++){
67          unsigned long long s = *b++;
68          sum += s;
69          if (sum < s) sum ++;
70          size -= 8;
71      }
72      // calculate the checksum for data in packet
73      b = (unsigned long long *) ((packet_T)buf)->data;
```

Open_listenfd: To create a listening descriptor, I also reference this function from 2018 cs230 class17 ppt.

```
94  // create a listening descriptor that can be used
95  // to accept connection requests from clients
96  // from 2018 cs230 class17 ppt
97  int open_listenfd(char *port)
98  {
99      struct addrinfo hints, *listp, *p;
100     int listenfd, optval=1;
101
102     /* Get a list of potential server addresses */
103     memset(&hints, 0, sizeof(struct addrinfo));
104     hints.ai_socktype = SOCK_STREAM;             /* Accept connect. */
105     hints.ai_flags = AI_PASSIVE | AI_ADDRCONFIG; /* …on any IP addr */
106     hints.ai_flags |= AI_NUMERICSERV;            /* …using port no. */
107     getaddrinfo(NULL, port, &hints, &listp);
108
109         /* Walk the list for one that we can bind to */
110     for (p = listp; p; p = p->ai_next) {
111         /* Create a socket descriptor */
112         if ((listenfd = socket(p->ai_family, p->ai_socktype,
113                                p->ai_protocol)) < 0)
114             continue;  /* Socket failed, try the next */
115
116         /* Eliminates "Address already in use" error from bind */
117         setsockopt(listenfd, SOL_SOCKET, SO_REUSEADDR,
118                    (const void *)&optval , sizeof(int));
119
120         /* Bind the descriptor to the address */
121         if (bind(listenfd, p->ai_addr, p->ai_addrlen) == 0)
122             break; /* Success */
123         close(listenfd); /* Bind failed, try the next */
124     }
125     /* Clean up */
126     freeaddrinfo(listp);
127     if (!p) /* No address worked */
128         return -1;
129
130     /* Make it a listening socket ready to accept conn. requests */
131     if (listen(listenfd, LISTENQ) < 0) {
132         close(listenfd);
133         return -1;
134     }
135     return listenfd;
136 }
```

Main: Like client, I allocate the memory for packet; and, dedicate the packet from client with connection between server and client by open_listenfd & accept function. And then, depend on operator, I encrypt/decrypt the data by ascii code; and, calculate the checksum again by changed data. Lastly, I write the data to the client from server.