Socket programming in C

Sachin Patil

March 30, 2018

1 Q & A

- In the client, show that ip address and the port number of the client is assigned implicitly within the connect system call.

  > Refer line 44 to line 49 in file $\text{tcp}_{\text{client.c}}$, where "client" socket addressing is defined. A client should run fine without those lines. Also note that a client will behave exactly similar with or without those lines. i.e. Client will choose `INADDR_ANY` (Special IP address which allows program to run without explicitly knowing IP address of a machine. Basically it chooses a default IP address of lowest interface available which is `lo` (loopback) in most cases)

  For client to connect, it does not need to know the port number of a machine it runs on. However defining `.sin_port = 0` (line 46) allows kernel to choose an ephemeral port available. Hence without specifying an IP address or a port client should connect without any problem.

- Find out the port number and ip address corresponding to your client.

  > The server always knows everything about client. But for server code to print client information, we have to create a "client socket" to hold information of a client to print it after the connection is (accept)ted by server. Refer lines 35 in file $\text{tcp}_{\text{server.c}}$ where a `struct` to hold client's information is defined. This information is later printed using line 51, just after `accept()`. It is important to print client info only after `accept()`, if one tries to print client's info before `accept()` it will print "garbage". Try shifting line 51 before line 48(where is connection is accepted)

- Is there any way of forcing your own port number for the client?

  > Yes, try replacing value of `.sin_port` from 0 to `htons(9000)` in file $\text{tcp}_{\text{client.c}}$ Now the client will open port 9000 for server to connect. In this way we are forcing the client to communicate over 9000 with the server.

2 Usage

1. Download code for both client and server using

```
git clone https://github.com/psachin/udemy-c-socket-programming.git \
    -b network-programming-assignment-1
```

2. To compile, run

```
 make
```

Typical output of `make` will be,

```
$ make
gcc                                tcp_client.c   -o tcp_client
gcc                                tcp_server.c   -o tcp_server
```

3. If you do any changes in code, simple run `make` again. That should re-compile the binary

4. Runing a server

```
./tcp_server
```

Typical output

```
$ ./tcp_server
Server listening on 0.0.0.0:9002
```

5. Running a client

```
./tcp_client
```

Typical output of a client when it successfully connects with the server

```
$ ./tcp_client
Server responded with: You have reached the server
```

Typical output of a server when client is successfully

```
$ ./tcp_server
Server listening on 0.0.0.0:9002
Client addr: 127.0.0.1:45321
```