CSCI4430 Data Communication and Computer Networks 2016 - 2017 2nd term
Assignment 1 – Socket Programming Exercise

Deadline: 23:59 14th February 2017

1. Introduction

In this assignment you are required to implement a client program and a multi-thread server program. The client program should be able to connect to the server with the server hostname/IP address and port number. Your server should be able to handle multiple connections concurrently. It should create a separate thread to handle the connection from each client. After connecting to the server, a client can send multiple messages until a SIGQUIT signal (i.e. Ctrl-D) is detected. The server should maintain a global counter to count the number of messages received. For each message received, the server should print the message count, client IP address, client port number and the message. Note that you need to maintain the global counter in a critical session.

2. Input and Output Specification

Here is the example on the expected input of the client program when 2 clients are connected to the server. The first client is executing on a computer with a hostname "PC2" and an IP address 192.168.1.2. Meanwhile, the second client is executing on a computer with a hostname "PC3" and an IP address 192.168.1.3. Each client has 2 messages inputted through standard input.

Expected Input (Client Side)

Console, PC2 (IP Address: 192.168.1.2)

```
user@PC2:~$ ./client PC1 44300
The quick brown fox jumps over the lazy dog.
The quick brown dog jumps over the lazy cat.
```

Console, PC3 (IP Address: 192.168.1.3)

```
user@PC3:~$ ./client PC1 44300
Hello World!!
Hello Earth!!
```

Assuming the 4 messages inputted to the two client programs above are sent to the port 44300 of server with IP 192.168.1.1 and they are received in the following order.

1. The quick brown fox jumps over the lazy dog.
2. Hello World!!
3. Hello Earth!!
4. The quick brown dog jumps over the lazy cat.

The server should print the received messages to the standard output in the order as these messages are received with the message count, source IP and port number as shown below.

Expected Output (Server Side)

Console, PC1 (IP Address: 192.168.1.1)

```
user@PC2:~$./server 44300
Message 1, From 192.168.1.2, Port 44301: The quick brown fox jumps over the lazy dog.
Message 2, From 192.168.1.3, Port 44302: Hello World!!
Message 3, From 192.168.1.3, Port 44302: Hello Earth!!
Message 4, From 192.168.1.2, Port 44301: The quick brown dog jumps over the lazy cat.
```

3. Definition on Command-line Argument

The definitions of command-line arguments of the client and server programs are shown below:

3.1. Client Program

Usage	
./client <server_addr> <server_port>	
Argument	**Description**
server_addr	The IP address or hostname of the server
server_port	The listening port of the server

3.2. Server Program

Usage	
./server <port>	
Argument	**Description**
port	The listening port of the server

4. General rules and assumptions

- The testing platform is Linux workstation "linux9" to "linux12" of CSE department.
- The programming language used should be C or C++.
- Every connection between the client and the server should only be done through TCP protocol under an IPv4 network.
- The client program can only be connected to 1 server. However, a server can be connected to more than 1 client concurrently.
- A line of message is terminated by line return ("\n") character. Therefore, the last character of every message sent from a client to a server should be a line return ("\n") character.
- The length of a message ranges from 1 to 128 inclusively (Including line returning character "\n").
- The format of an IPv4 address is XXX.XXX.XXX.XXX, where XXX is an integer ranging from 0 to 255.
- It is assumed that every run-time argument entered into the client and server program is correct.
- It is assumed that every character in every message is printable.
- It is assumed that there will not be more than 50 clients concurrently connected a server.
- It is assumed that a client will send messages at a maximum rate of 1 message per second.

5. Submission

- Submit a ZIP file to the collection box at eLearning platform. The ZIP file should consist of:
  - All source codes you have written
  - A Makefile which can automate the compilation process