COSC402 Lab 1: Core Sockets API Functions

Lab Setup

We will use MacOS for 402 labs. The source package for the first four labs is put in the following course pickup directory: `/home/cshome/coursework/402/pickup`. Copy the "402lab" folder to your local machine, and follow the instructions given in the *README* file to compile the source package.

Programming

Copy the *Lab1* folder from the pickup directory into the "402lab" directory on your local machine. In this lab you are asked to implement client and server programs for copying a text file from server to the client based on the skeleton code provided in *tcp_easy_ipv4_client.c* and *tcp_easy_ipv4_server.c*. You can refer to the sample code provided in lecture notes when implementing these two programs. If you are unfamiliar with some API functions, please use the *man* command in the terminal to check the manual of these functions. For how to use the *man* command, please refer to the following wiki page: `https://en.wikipedia.org/wiki/Man_page`.

- In *tcp_easy_ipv4_client.c*, you need to implement **tcp_easy_connect()** and **dump_socket_to_stdout()**.
- In *tcp_easy_ipv4_server.c*, you need to implement **tcp_easy_listen()** and **server_loop()**.

Testing

Open a terminal and change into the *Lab1* directory. Use the **make** command to compile the source codes. Once the executable files have been generated with no errors, do the following tests.

Test 1: local machine

1. Create a text file named test.txt, add any text you want in this text file, and put it in the *Lab1* directory.
2. Open one terminal window, change into the *Lab1* directory, and type the following command to run the server program: **./tcp_easy_ipv4_server 127.0.0.1 65530 test.txt**
3. Open another terminal window, change into the *Lab1* directory, and type the following command to run the client program: **./tcp_easy_ipv4_client 127.0.0.1 65530**
4. Check the outputs in the two terminal windows. If the client terminal window successfully displays the text in the test.txt file, your implementation is correct.

Test 2: network

1. Open one terminal window, type **ifconfig** to get the IPv4 address of your local machine.
2. Repeat step 2 in Test 1 by replacing the IP address with the IPv4 address of your local machine.
3. Copy the source package (402lab) to another machine (your laptop with Internet connection or a machine in the lab). Recompile the two programs you developed. Repeat step 3 in Test 1 by replacing the IP address with the IPv4 address you get in Step 1.
4. Check whether the developed programs work in a network environment.

Test 3: send a big file

The bigtext.txt file in *Lab1* has a size of 285MB. Send this file from sever to a client in this test.

1. Open one terminal window, change into the *Lab1* directory, and type the following command to run the server program: **./tcp_easy_ipv4_server 127.0.0.1 65530 bigtext.txt**
2. Open the second terminal window, type the following command to check the status of the server process: **netstat -a | grep 65530**
3. Open the third terminal window, change into the *Lab1* directory, and type the following command to run the client program: **./tcp_easy_ipv4_client 127.0.0.1 65530**
4. In the second terminal window type the following command to check the status of the client and server processes: **netstat -a | grep 65530**
5. Open two terminal windows and run the client program at the same time. What happens? Can you explain why this happens?