CSP334: Computer Networks
Lab Assignment No 5
Socket Programming #1

Date of First Review of your implementation : 4th October 2018
Date of Final submission : 11:59 hrs, 14th OCtober 2018
Autumn Semester 2018

Instructions:

1. *Please use only the C as the language of programming.*
2. *Please submit/upload on LMS, the following files for each of your programs: (1) the client and the server source files each (2) the client and the serve executable files each (3) a brief Readme file that shows the usage of the program.*
3. *Please appropriately comment your program and name all the identifiers suitable, to enable enhanced readability of the code.*
4. *Neat coding is necessary as illustrated in the class, that will carry specific marks for each program that you write.*
5. *In the first review (during the lab hours) on the stipulated date, it is expected that a few (if not all) of the assignments in this list are compledted and the execution shown and explained to the teacher in the lab.*

Problems

1. As discussed in class, the socket address is defined as follows:

```
struct  in_addr {u_long s_addr; }
/*32-bits netID or hostID in NBO */

struct  sockaddr_in  {
        short   sin_family;  /*2 bytes      AF_INET*/
        u_short sin_port;        /* 2 byte port no */
        struct  in_addr sin_addr; /* 4 bytes */
         char           sin_zero[8];     /* 8 bytes      unused */
         /*used for padding  the length of the structure to 16 bytes */
}
```

   Write a brief note on the meaning of each component of both the structures.

2. Write a program with two functions: (1) First function takes an IPA as argument from the user on the command line (i.e. in the form of a string) and populates the `structure sockaddr_in` with the IPA in the NBO; using the function int `inet_aton(const char *cp, struct in_addr *inp)` (2) Second, reads the IPA from an already populated structure and prints the IPA in the string format using the counterpart function `int inet_ntoa(const char *cp, struct in_addr *inp)`

3. As discussed in class, the Intel machines are little-endian machines whereas the NBO is big-endian. Using the logic discussed in the class, write a program that detects and prints whether a machine is a little-endian or a big-endian. Write another program that would - convert the same to network byte order and given an IPA and port number in network byte order, would convert the same to the host byte order.

4. Write a tcp client program that does not use `bind()` function, but relies upon the kernel to provide an IPA and port number. Since the IPA and port number are provided by the kernel, write a function which obtains the local socket name using the call viz.`int getsockname(int sockfd, struct sockaddr *localaddr, socklet_t *addrlen)` and displays the allocated IPA and port numbers. Look up the *man* pages of *getsockname* to learn how to use this function.

5. Write a program that takes multiple command line arguments (which are machine names) and uses appropriate function to print their IP addresses. Write its counterpart that takes multiple command line arguments (which are IP addresses) and uses appropriate function to print their hostnames. Include an additional option which determines which function to invoke and which argument to take.

6. Modify the TCP and UDP dautime service client illustrated in the lab to display the hostname and the port number of the server, also along with the time of the day. Recollect that he daytime client uses the datyime service running on port no 13 on the server to fetch the time of the day from the server and display the same on the standard output on the client side.

7. Now modify the client in the problem number 6 that requests an iterative TCP server on port no 99XX (XX are the last two digits of your IIT enrollement number) to send it the time of the day. The server upon receiving the client request responds with the time of the day, which the client outputs on the standard output, along with the hostname and the port number of the server. Now start the *tcpdump* with appropriate arguments (refer $man tcpdump) to capture packets from or to a remote host. After you have started the TCP server and the TCP client terminate tcpdump, examine its output and make appropriate inferences based on the output.

8. Repeat the exercise in Problem no 7, but now with UDP as the underlying Transport protocol.

9. Modify the server program in problem 7 to make it work as a concurrent server.