CSC 255: Project 2

Socket Programming Assignment 2
C implementation of Client-Server Communication

Submitted on: 28 November 2018

Ramandeep Chumber

Index 2

Introduction

Sockets are low level endpoint which are used to process information across a network. In this programming assignment we are using C implementation of client-server communication. The client reads data from keyboard and sends the data to the server. The server receives the data and displays it on the screen.

Process

Take a look at the c programs I write for server and client below. After looking at the results we will look at code description.

```c
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <stdio.h>
#include<string.h>

void main()
{
    char str[100];

    //define and create the server socket
    int server_socket, comm_fd;
    struct sockaddr_in serveraddr;
    server_socket = socket(AF_INET, SOCK_STREAM, 0);
    if(server_socket == -1)
      {
       perror("Socket not created\n");
      }

    bzero( &serveraddr, sizeof(serveraddr));
    serveraddr.sin_family = AF_INET;
    serveraddr.sin_addr.s_addr = htons(INADDR_ANY);
    serveraddr.sin_port = htons(22000);

    //bind the socket to our specified IP and port
   int bsocket = bind(server_socket, (struct sockaddr *) &serveraddr, sizeof(serveraddr));
    if(bsocket == -1)
      {
       perror("Socket not bound\n");
      }

    int ulisten = listen(server_socket, 10);
     if(ulisten == -1)
      {
       perror("Unable to listen to the client\n");
      }

    comm_fd = accept(server_socket, (struct sockaddr*) NULL, NULL);

     if(comm_fd == -1)

      {
       perror("Unable to accept connection.\n");
      }

    // read and print the client msg
    while(1)
    {
        bzero( str, 100);
        read(comm_fd,str,100);
        printf("%s",str);

    }
    // close the socket
    close(server_socket);
}
```

Figure 1: server.c program for server.

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <string.h>

int main(int argc,char *argv[])
{
    // create a socket
    int network_socket, n;
    char sendline[100];

    // specify an address for the socket
    struct sockaddr_in serveraddr;
    network_socket=socket(AF_INET,SOCK_STREAM,0);

     //check error with socket
    if(network_socket == -1)
     {
     perror("Socket not created\n");
     }

    // IP address
    struct hostent *h;
    struct sockaddr_in sin;
    char domain[512];
    sin.sin_addr.s_addr=gethostid();
    h = gethostbyaddr((char *)&sin.sin_addr.s_addr,
    sizeof(struct in_addr), AF_INET);
    if (h==(struct hostent *)0)
       {
         printf("gethostbyaddr failed\n");
       }

    //specify an address for the socket

    int result;
    bzero(&serveraddr,sizeof serveraddr);
    serveraddr.sin_family=AF_INET;
    serveraddr.sin_port=htons(22000);
    result = connect(network_socket,(struct sockaddr *)&serveraddr,sizeof(serveraddr));

    // check for error with the connection
    if (result == -1) {
       printf("There was some error in the connection to the server \n\n");
     }

    //read data from keyboard and send it to server
    while(1)
    {
        bzero(sendline, 100);
        fgets(sendline,100,stdin); /*stdin = 0 , for standard input */
       n = write(network_socket,sendline,strlen(sendline)+1);
     if (n<0)
       {
        perror("Error in writing\n");
       }

    }
    // close the socket
    close(network_socket);
}
```

Figure 2: client.c program for the client

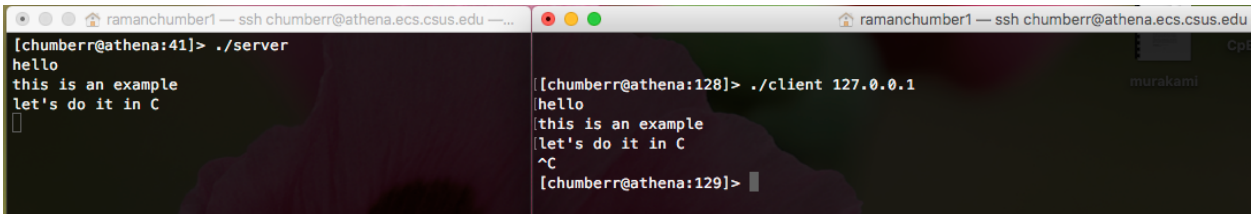

Figure 3: the message typed on client, displayed on server side.

Code description

server.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <stdio.h>
#include<string.h>

void main()
{
  char str[100];

  //define and create the server socket
  int server_socket, comm_fd;
  struct sockaddr_in serveraddr;
  server_socket = socket(AF_INET, SOCK_STREAM, 0);
  if(server_socket == -1)
   {
    perror("Socket not created\n");
   }

  bzero( &serveraddr, sizeof(serveraddr));
  serveraddr.sin_family = AF_INET;
  serveraddr.sin_addr.s_addr = htons(INADDR_ANY);
  serveraddr.sin_port = htons(22000);

  //bind the socket to our specified IP and port
 int bsocket = bind(server_socket, (struct sockaddr *) &serveraddr, sizeof(serveraddr));
  if(bsocket == -1)
   {
    perror("Socket not bound\n");
   }

  int ulisten = listen(server_socket, 10);
   if(ulisten == -1)
   {
    perror("Unable to listen to the client\n");
   }
```

```
    comm_fd = accept(server_socket, (struct sockaddr*) NULL, NULL);

     if(comm_fd == -1)

     {
      perror("Unable to accept connection.\n");
     }

    // read and print the client msg
    while(1)
    {
       bzero( str, 100);
       read(comm_fd,str,100);
       printf("%s",str);

    }
    // close the socket
    close(server_socket);
}
```

We are using the following libraries for definitions of socket functions that we are going to use in our program later. It will include all the sockets functionality and the APIs that we are using.

```
1.  #include <sys/types.h>
2.  #include <sys/socket.h>
3.  #include <netdb.h>
```

We need **<netdb.h>** for the structures that we are using for addresses.
Following File Descriptors are used

```
int server_socket, comm_fd;
```

We need structures to hold IP Address and Port Numbers.

```
struct sockaddr_in serveraddr;
```

The server will listen to the client. The above function creates a socket with **AF_INET** (IP Addressing) and of type **SOCK_STREAM**. We need integer to hold the socket. We call it **server_socket**. And we store the results in the integer. Below is how to call socket function:

```
server_socket = socket(AF_INET, SOCK_STREAM, 0);  server_socket = socket(AF_INET, SOCK_STREAM, 0);
```

We need to define few fields on structure. Followings are some of them we are using in the code:

```
1.      serveraddr.sin_family = AF_INET;
2.      serveraddr.sin_addr.s_addr = htons(INADDR_ANY);
3.      serveraddr.sin_port = htons(22000);
```

If client send message and the server will accept, and it can be read from **comm_fd**, and whatever we write using **comm_fd** will also be sent to the other device.

```
1.  comm_fd = accept(server_socket, (struct sockaddr*) NULL, NULL);
```

client.c

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <string.h>

int main(int argc,char *argv[])
{
   // create a socket
   int network_socket, n;
   char sendline[100];

   // specify an address for the socket
   struct sockaddr_in serveraddr;
   network_socket=socket(AF_INET,SOCK_STREAM,0);

    //check error with socket
   if(network_socket == -1)
    {
    perror("Socket not created\n");
    }

   // IP address
   struct hostent *h;
   struct sockaddr_in sin;
   char domain[512];
   sin.sin_addr.s_addr=gethostid();
   h = gethostbyaddr((char *)&sin.sin_addr.s_addr,
   sizeof(struct in_addr), AF_INET);
   if (h==(struct hostent *)0)
     {
       printf("gethostbyaddr failed\n");
     }

   //specify an address for the socket

   int result;
   bzero(&serveraddr,sizeof serveraddr);
   serveraddr.sin_family=AF_INET;
   serveraddr.sin_port=htons(22000);
   result = connect(network_socket,(struct sockaddr *)&serveraddr,sizeof(serveraddr));

   // check for error with the connection
```

```
    if (result == -1) {
      printf("There was some error in the connection to the server \n\n");
     }

    //read data from keyboard and send it to server
    while(1)
    {
       bzero(sendline, 100);
       fgets(sendline,100,stdin); /*stdin = 0 , for standard input */
      n = write(network_socket,sendline,strlen(sendline)+1);
     if (n<0)
      {
       perror("Error in writing\n");
      }

    }
    // close the socket
    close(network_socket);
}
```

The Client then does following forever:

1. Clear **sendline**
2. read string from **stdin** in **sendline** (stdin is 0 for standard input)
3. write **sendline** in **sockfd**
4. Display **recvline**

in the following code:

```
1.  while(1)
2.      {
3.          bzero( sendline, 100);
4.
5.          fgets(sendline,100,stdin); /*stdin = 0 , for standard input */
6.
7.          write(network_socket,sendline,strlen(sendline)+1);
8.      }
```

Conclusion

This lab helped me in understanding the details about sockets, ports and socket programming over TCP in C. I tried to do error checks as much as possible in both client and server program. I think it would help to make program much more secure. Error handling is important in cases where the communication between server and client can go wrong at many places as per my understanding. For example, if the connection is not established between server and client and you are wondering where the problem is occuring. Error handling is important especially in case of buffer overflow. The clinician told me to fix my code at two places. My previous client.c program had IP address of the client in the program. So, I used “gethostbyaddr” function to fix it. I was able to fix the code by using “gethostbyname” function as well. I really enjoyed working on this project.