UCS1511 - Networks Lab **AY:** *2022-23*	**Name:** *Krithika Swaminathan* **Roll No.:** *205001057*

Assignment 2 – Client Server Communication

Date: **26/08/2022**

Exercise 2a: Echo Client Server

Aim:

To develop a socket program to establish client-server communication. The client sends data to server. The server in turn sends the message back to the client. Send multiple lines of text.

Algorithm:

Server:

1. Start
2. Create a socket using socket() system call.
3. Bind() is used to bind the socket with a specified address defined by sockaddr_in pointer, with the address, family, port set accordingly.
4. Bzero() is used to clear the address pointer initially.
5. The listen() call is used to make the created socket listen for incoming connections. The maximum no. of connections that can be accepted is specified here.
6. The accept() call is used to make the server accept any connection requests. The parameter sockaddr_in accept() holds the requesting clients address, with which the messages are addressed to.
7. Read() is used to read data from the client into a temporary buffer.
8. The same message is sent back to the client using the write() system call.
9. Then, the received message is displayed.
10. Stop

Client:

1. Start
2. Create a socket using socket() system call.

3. The socket descriptor is noted using which a connect() call is made to connect to the server, whose address is to be specified as a pointer of sockaddr_in in the sockaddr field of connect().
4. The sockaddr_in pointer holds the address, set by user, some port, ip of the server machine, respectively.
5. Once the server accepts the call, the client requests input from the user, sends it to the server with the write() call. The server returns the same message and it is read using the read() system call.
6. Stop

Code:

```
/* C program to establish client-server communication using sockets */
//Server
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>
#include <arpa/inet.h>

#define MAX 150

#define SA struct sockaddr
int main(int argc, char ** argv)
{

   int PORT = atoi(argv[1]);
   int sockfd, new_fd, len;
   struct sockaddr_in servaddr, cli;
   char buff[MAX];
   int n;
   // socket create and verification
   sockfd = socket(AF_INET, SOCK_STREAM, 0);
   if (sockfd == -1)
   {
          fprintf(stderr, "Socket creation failed!\n");
          exit(0);
```

```
}
else
	printf("Socket creation successful!\n");

bzero(&servaddr, sizeof(servaddr));
// assign IP, PORT
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(PORT);

// Binding newly created socket to given IP and verification
if ((bind(sockfd, (SA *)&servaddr, sizeof(servaddr))) != 0)
{
	fprintf(stderr,"Socket bind failed!\n");
	exit(0);
}
else
	printf("Socket successfully binded..\n");

// Now server is ready to listen and verification
if ((listen(sockfd, 5)) != 0)
{
	fprintf(stderr,"Listen failed!\n");
	exit(0);
}
else
	printf("Server listening..\n");

len = sizeof(cli);
// Accept the data packet from client and verification
new_fd = accept(sockfd, (SA *)&cli, &len);
if (new_fd < 0)
{
	fprintf(stderr,"Server accept failed!\n");
	exit(0);
}
else
	printf("Accept Successful!\n");

bzero(buff, MAX);
// read the message from client and copy it in buffer
read(new_fd, buff, sizeof(buff));
// print buffer which contains the client contents
```

```
    printf("Message from client: %s\n", buff);
    // After chatting close the socket

    bzero(buff, MAX);
    printf("Server: ");
    scanf("%[^\n]", buff);
    write(new_fd, buff, sizeof(buff));
    close(sockfd);
}

//Client
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
#include <arpa/inet.h>

#define MAX 150
#define SA struct sockaddr
int main(int argc, char ** argv)
{

    int PORT = atoi(argv[1]);
    int sockfd, connfd;
    struct sockaddr_in servaddr, cli;
    char buff[MAX];
    int n; // socket create and verification
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1)
    {
            fprintf(stderr,"Socket creation failed!\n");
            exit(0);
    }
    else
            printf("Socket creation successful!\n");

    bzero(&servaddr, sizeof(servaddr));
    // assign IP, PORT
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);
```

```
    // connect the client socket to server socket
    if (connect(sockfd, (SA *)&servaddr, sizeof(servaddr)) != 0)
    {
            fprintf(stderr,"Connection failed!\n");
            exit(0);
    }
    else
            printf("Connection to server successfull!\n");

    printf("Client: ");
    scanf("%[^\n]", buff);
    buff[strlen(buff)] = 0;
    write(sockfd, buff, sizeof(buff));
    read(sockfd, buff, sizeof(buff));

    printf("Message from server: %s\n", buff);
    close(sockfd);
}
```

UCS1511 - Networks Lab *AY: 2022-23*	Name: *Krithika Swaminathan* Roll No.: *205001057*

Output:

Server side:

```
root@spl21:~/kri# gcc -o server server2a.c
root@spl21:~/kri# ./server
Segmentation fault (core dumped)
root@spl21:~/kri# ./server 8080
Socket creation successfull!
Socket successfully binded..
Server listening..
Accept Successfull!
Message from client: Hello
Server: Bye
root@spl21:~/kri#
```

Client side:

```
root@spl21:~/kri# ./client 8080
Socket creation successfull!
Connection to server successfull!
Client: Hello
Message from server: Bye
root@spl21:~/kri#
```

Learning outcomes:

- The process of establishing a connection between a client and a server was understood.
- A client-server connection was established and implemented.
- Sockets were used to establish a client-server connection.
- File transfer was implemented using TCP.
