UCS 1511 - Network Lab

Exercise 02A - Simple Client Server using TCP
Excersie 02B - Echo Server Using TCP

Name:	Mahesh Bharadwaj K
Reg No:	185001089
Semester:	V
Date:	August 27, 2020

---

1 Simple Client Server using TCP

Aim:

To develop a socket program to establish a client server communication. The client sends data to server. The server replies to the client.

Algorithm

1. Server

1. Create a socket descriptor with **socket()** system call with AF_INET (IPV4 domain), SOCK_STREAM, default protocol and store as sockfd.
2. If sockfd is a negative number, socket creation failed, end program.
3. Create sockaddr_in object to assign IP address and Port number for socket. Set family to AF_INET, IP address to INADDR_ANY to accept connections from any client and port number required.
4. Bind newly created socket to addresss given in sockaddr_in.
5. If bind is non zero, bind failed, print error message and terminate.
6. Listen on the socked defined for as many clients as required. If **listen()** returns non zero value, print error message and terminate.
7. Accept connections from socket using **accept()** system call and store client socket descriptor in a separate variable.
8. Read message into buffer using **read()** system call.
9. Read the message to send to client into the buffer from the user.
10. Write the message from buffer onto client using **write()** system call.
11. Close connections on socket using **close()** and terminate program.

2. Client

1. Create a socket descriptor with **socket()** system call with AF_INET (IPV4 domain), SOCK_STREAM, default protocol and store as sockfd.
2. If sockfd is a negative number, socket creation failed, end program.
3. Create sockaddr_in object to assign IP address and Port number for socket. Set family to AF_INET, IP address to localhost(127.0.0.1) to connect to server and port number required.
4. Connect the client to server at address given in socket descriptor using **connect()** system call.

5. If connect() returns -1, connection failed; Print error message and terminate the program.
6. Read message from user into buffer variable and write into server socket using **write()** system call.
7. Read the response from server into buffer variable using **read()** system call and display received message from the user.
8. Close the connections on socket using **close()** and terminate program.

Program

1. Server Side

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.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 successfull!\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 acccept failed!\n");
                exit(0);
        }
        else
                printf("Accept Successfull!\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);
}
```

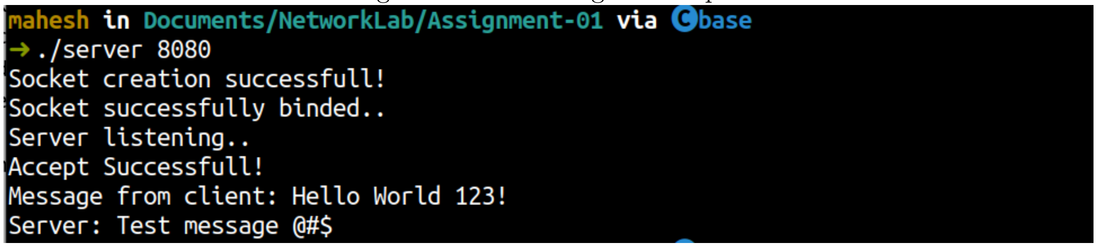
---

2. Client Side

```
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.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 varification
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd == -1)
        {
                fprintf(stderr,"Socket creation failed!\n");
                exit(0);
        }
        else
                printf("Socket creation successfull!\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);
}
```

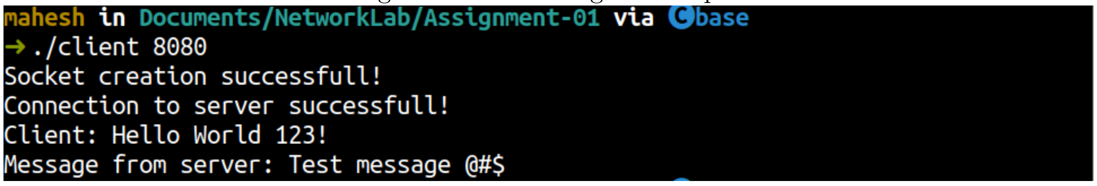
Output

Figure 1: Server Program Output



```
mahesh in Documents/NetworkLab/Assignment-01 via base
➜ ./server 8080
Socket creation successfull!
Socket successfully binded..
Server listening..
Accept Successfull!
Message from client: Hello World 123!
Server: Test message @#$
```

Figure 2: Client Program Output



```
mahesh in Documents/NetworkLab/Assignment-01 via base
➜ ./client 8080
Socket creation successfull!
Connection to server successfull!
Client: Hello World 123!
Message from server: Test message @#$
```

2 Echo server using TCP

Aim:

Develop a socket program to establish a client server communication. The client sends data to server. The server in turn sends the same message back to the client. Transmit multiple lines of text. In client side display the echoed message. In server side display the message which is echoed to client.

Algorithm

1. Server

1. Create a socket descriptor with **socket()** system call with AF_INET (IPV4 domain), SOCK_STREAM, default protocol and store as sockfd.
2. If sockfd is a negative number, socket creation failed, end program.
3. Create sockaddr_in object to assign IP address and Port number for socket. Set family to AF_INET, IP address to INADDR_ANY to accept connections from any client and port number required.
4. Bind newly created socket to addresss given in sockaddr_in.
5. If bind is non zero, bind failed, print error message and terminate.
6. Listen on the socked defined for as many clients as required. If **listen()** returns non zero value, print error message and terminate.
7. Accept connections from socket using **accept()** system call and store client socket descriptor in a separate variable.
8. Read message into buffer using **read()** system call.
9. Write the back the message from buffer onto client using **write()** system call.
10. Close connections on socket using **close()** and terminate program.

2. Client

1. Create a socket descriptor with **socket()** system call with AF_INET (IPV4 domain), SOCK_STREAM, default protocol and store as sockfd.
2. If sockfd is a negative number, socket creation failed, end program.
3. Create sockaddr_in object to assign IP address and Port number for socket. Set family to AF_INET, IP address to localhost(127.0.0.1) to connect to server and port number required.
4. Connect the client to server at address given in socket descriptor using **connect()** system call.
5. If connect() returns -1, connection failed; Print error message and terminate the program.
6. Read message from user into buffer variable and write into server socket using **write()** system call.
7. Read the echoed response from server into buffer variable using **read()** system call and display received message from the user.
8. Close the connections on socket using **close()** and terminate program.

Program

1. Server Side

```c
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.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 successfull!\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 acccept failed!\n");
                exit(0);
        }
        else
                printf("Accept Successfull!\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

        printf("Message echoed to client: %s\n", buff);
        write(new_fd, buff, sizeof(buff));
        close(sockfd);
}
```

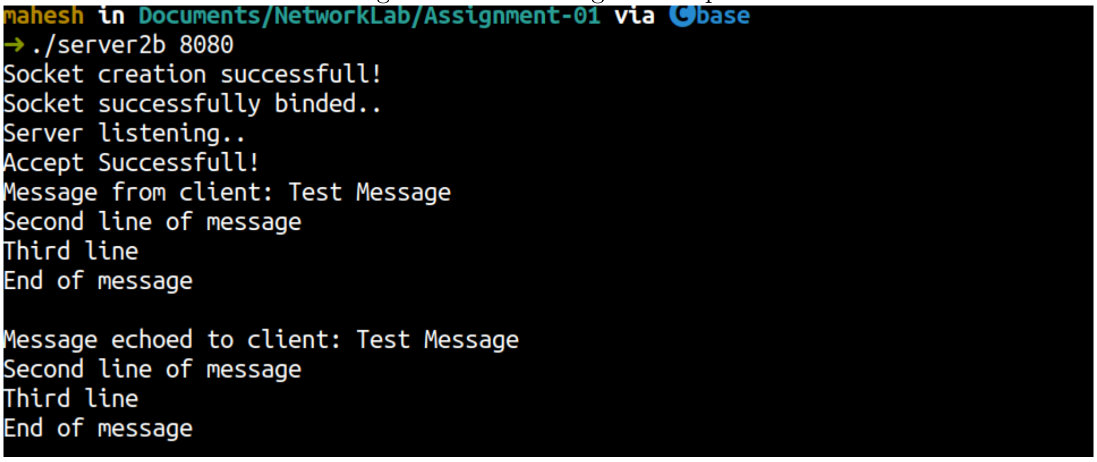
---

2. Client Side

```
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.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 varification
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd == -1)
        {
                fprintf(stderr,"Socket creation failed!\n");
                exit(0);
        }
        else
                printf("Socket creation successfull!\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("%[^;]", buff);
        buff[strlen(buff)] = 0;
        write(sockfd, buff, sizeof(buff));
        read(sockfd, buff, sizeof(buff));

        printf("Echoed message: %s\n", buff);
        close(sockfd);
}
```

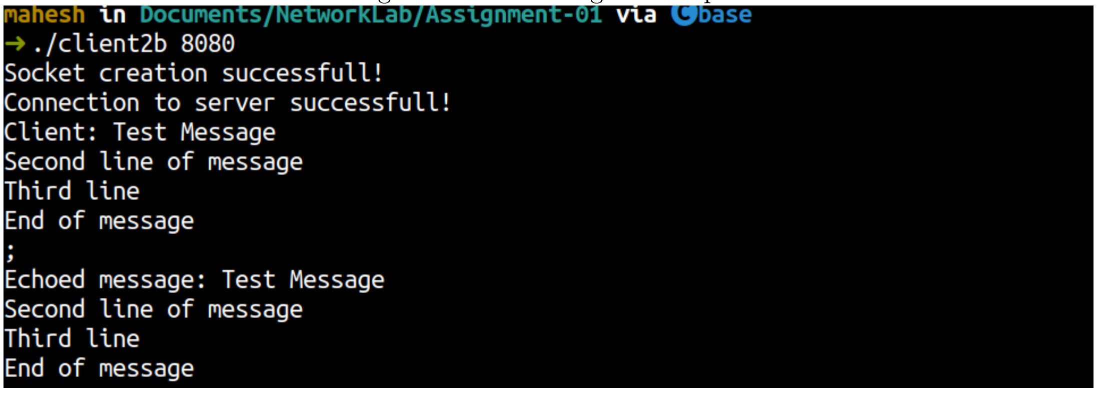
Output:

Figure 3: Server Program Output



```
mahesh in Documents/NetworkLab/Assignment-01 via base
→./server2b 8080
Socket creation successfull!
Socket successfully binded..
Server listening..
Accept Successfull!
Message from client: Test Message
Second line of message
Third line
End of message

Message echoed to client: Test Message
Second line of message
Third line
End of message
```

Figure 4: Client Program Output



```
mahesh in Documents/NetworkLab/Assignment-01 via base
→./client2b 8080
Socket creation successfull!
Connection to server successfull!
Client: Test Message
Second line of message
Third line
End of message
;
Echoed message: Test Message
Second line of message
Third line
End of message
```