SSN COLLEGE OF ENGINEERING, KALAVAKKAM

(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1511 – COMPUTER NETWORKS LAB

Lab Exercise 2a – Echo client server

Aim:

Develop a socket program to establish a 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. Create a socket using socket() system call.
2. bind() is used to bind the socket with a specified address defined by sockaddr_in pointer, with the address, family, port set accordingly, bzero() is used to clear the address pointer initially.
3. listen() to make the created socket listen for incoming connections, maximum no of connections that can be accepted is specified here.
4. accept() call 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.
5. read() is used to read data from client into a temporary buffer.
6. Sends the same message back to the client using write() system call.
7. Then the received message is displayed.

Client:

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

Code:

q1-server.c

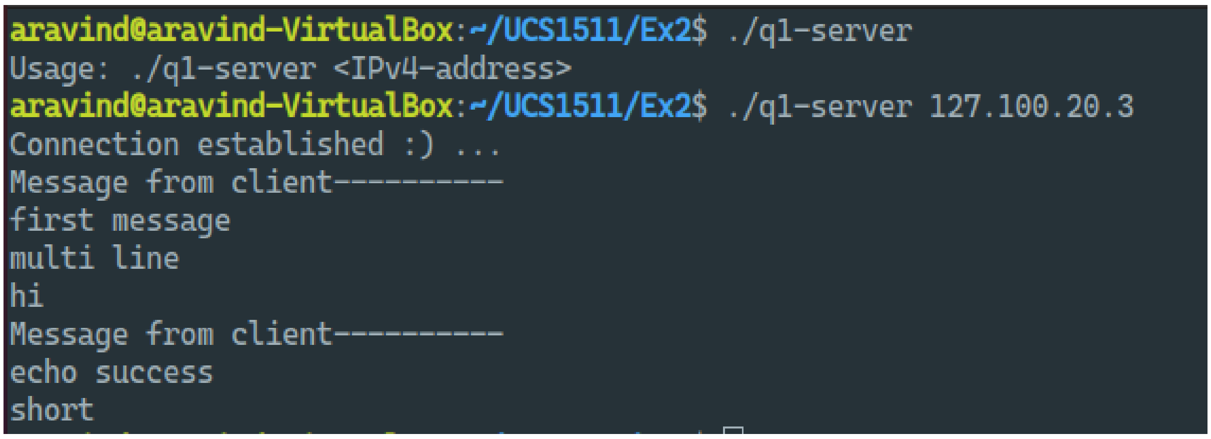
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h> // strcmp(), bzero()

#include <sys/types.h>  // socket(), bind(), listen(), accept()
#include <sys/socket.h> // socket(), inet_addr(), bind(), listen(),
accept()

#include <netinet/in.h> // inet_addr()
#include <arpa/inet.h>  // htons(), inet_addr()

#include  <unistd.h> // read(), write(), close()

#define PORT 9002

int main(int argc, char* argv[])
{
  // Check if args contain IPv4 address
  if(argc < 2)
  {
    printf("Usage: ./q1-server <IPv4-address>\n");
    return -1;
  }

  // create the server socket
  int server_socket;
  server_socket = socket(AF_INET, SOCK_STREAM, 0);

  // define the server address
  struct sockaddr_in server_address;

  // clear the address pointer
  bzero(&server_address, sizeof(server_address));

  server_address.sin_family = AF_INET;
  server_address.sin_port = htons(PORT);
  // server_address.sin_addr.s_addr = INADDR_ANY;
  server_address.sin_addr.s_addr =  inet_addr(argv[1]);

  // bind the socket to the specified IP and port
  bind(server_socket, (struct sockaddr*) &server_address,
sizeof(server_address));

  listen(server_socket, 5);

  int client_socket;
  client_socket = accept(server_socket, NULL, NULL);

  if(client_socket != -1)
  {
    printf("Connection established :) ...\n");
  }

  /*
  char server_message[256] = "This is a server message! Hello!";
  // send the message
  send(client_socket, server_message, sizeof(server_message), 0);
  */

  char client_msg_dest[256];
  // char server_msg_src[256];

  while(1)
  {
    // bzero(server_msg_src, sizeof(server_msg_src));
    read(client_socket, &client_msg_dest, sizeof(client_msg_dest));
    if(strcmp(client_msg_dest, "exit") == 0)
    {
      break;
    }
    printf("Message from client----------\n%s\n", client_msg_dest);
    write(client_socket, client_msg_dest, sizeof(client_msg_dest));
  }

  // close the socket
  close(server_socket);

  return 0;
}
```

q1-client.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h> // strcmp(), bzero()

#include <sys/types.h>  // socket(), connect()
#include <sys/socket.h> // socket(), inet_addr(), connect()

#include <netinet/in.h> // inet_addr()
#include <arpa/inet.h>  // htons(), inet_addr()

#include  <unistd.h> // read(), write(), close()

#define PORT 9002

int main(int argc, char* argv[])
{
  // Check if args contain IPv4 address
  if(argc < 2)
  {
    printf("Usage: ./q1-client <IPv4-address>\n");
    return -1;
  }
  // create a socket
  int network_socket;

  network_socket = socket(AF_INET, SOCK_STREAM, 0);
  // AF_INET - Domain[Protocol Family](IPv4 Internet protocols)
  // SOCK_STREAM - TCP Socket
  // 0 - Protocol Specification (not a raw socket)


  // specify the address for the socket
  struct sockaddr_in server_address;
  server_address.sin_family = AF_INET; // same family as our socket
  server_address.sin_port = htons(PORT); // set a port to listen to
  // server_address.sin_addr.s_addr =  INADDR_ANY; // INADDR_ANY = 0.0.0.0
  server_address.sin_addr.s_addr =  inet_addr(argv[1]); // convert a
numbers and dots TO byte order

  int connection_status = connect(network_socket, (struct sockaddr *)
&server_address, sizeof(server_address));
  // check for error in connection
  if (connection_status == -1) {
    printf("Error in connecting to server :/ Try again:(...\n");
  }
  else {
    printf("Connected :) ...\n");
  }

  char client_msg_src[256];
  char server_msg_dest[256];

  while(1)
  {
    bzero(client_msg_src, sizeof(client_msg_src));
    printf("Enter your message: ");
    scanf(" %[^;]%*c", client_msg_src);
    write(network_socket, client_msg_src, sizeof(client_msg_src));
    if(strcmp(client_msg_src, "exit") == 0)
    {
      break;
    }

    read(network_socket, &server_msg_dest, sizeof(server_msg_dest));
    printf("Message from server----------\n%s\n", server_msg_dest);
  }

  // close the socket
  close(network_socket);

  return 0;
}
```

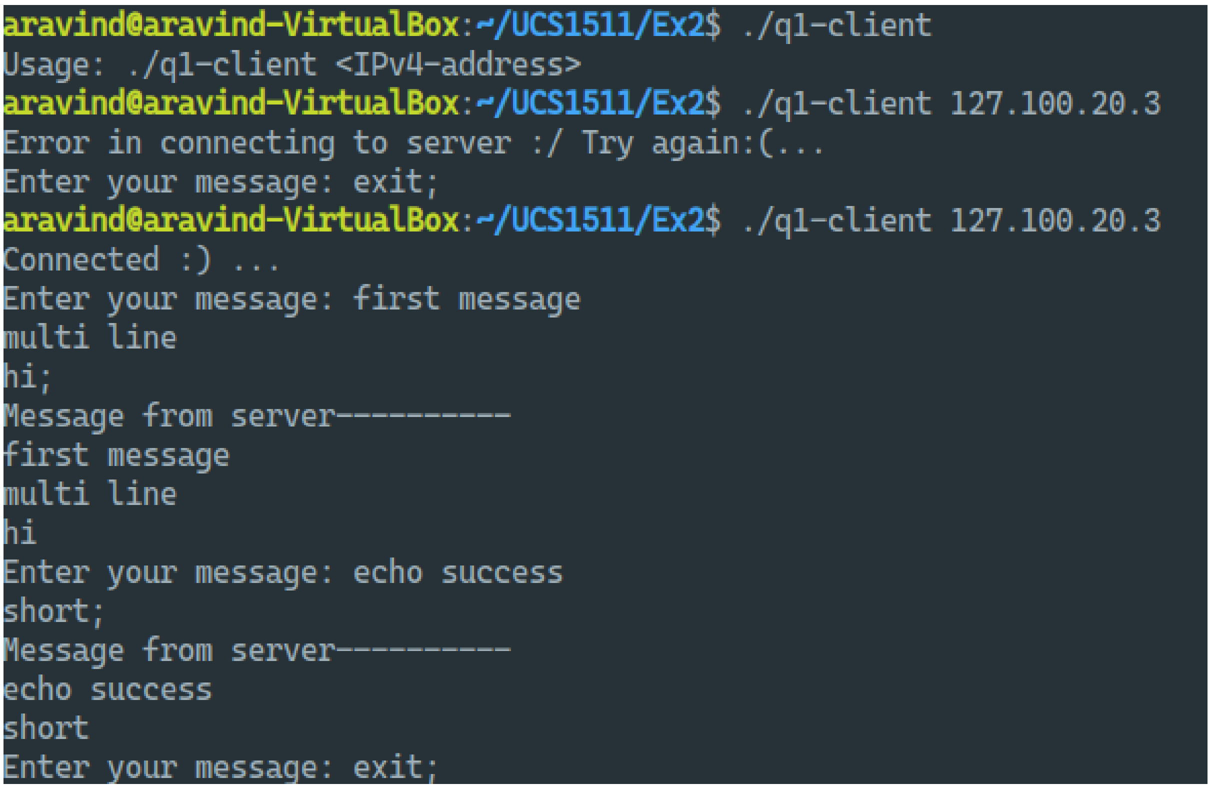
Output:

Server:



```
aravind@aravind-VirtualBox:~/UCS1511/Ex2$ ./q1-server
Usage: ./q1-server <IPv4-address>
aravind@aravind-VirtualBox:~/UCS1511/Ex2$ ./q1-server 127.100.20.3
Connection established :) ...
Message from client----------
first message
multi line
hi
Message from client----------
echo success
short
```

Client:



```
aravind@aravind-VirtualBox:~/UCS1511/Ex2$ ./q1-client
Usage: ./q1-client <IPv4-address>
aravind@aravind-VirtualBox:~/UCS1511/Ex2$ ./q1-client 127.100.20.3
Error in connecting to server :/ Try again:(...
Enter your message: exit;
aravind@aravind-VirtualBox:~/UCS1511/Ex2$ ./q1-client 127.100.20.3
Connected :) ...
Enter your message: first message
multi line
hi;
Message from server----------
first message
multi line
hi
Enter your message: echo success
short;
Message from server----------
echo success
short
Enter your message: exit;
```

Learning outcomes:

1. Introduced to socket programming.
2. Learned to create TCP sockets and connect to server using IPv4 address and port.
3. Explored a server and client using two terminals and server echoing client’s message.
4. Accept a delimited input string.