Department of Computer Science and Engineering

S.G.Shivanirudh , 185001146, Semester V

9 September 2020

UCS1511 - Networks Laboratory

Exercise 3: Chat using TCP

Objective:

Develop a simple chat using TCP socket.
To a chat server, multiple stations chat simultaneously.

Code:

Server:

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
```

```c
#include<string.h>
#include<unistd.h>
#include<sys/time.h>

int main(int argc,char **argv){
    //Server and Client addresses
    struct sockaddr_in server_address, client_address;
    //Buffer to handle messages
    char buffer[1024];
    //Storing sockets for client
    int client_sockets[30];
    //Set of file descriptors
    fd_set clientfds;
    //Socket file descriptor for accepting connections
    int newfd;

    for(int i = 0; i < 30; i++)
        client_sockets[i] = 0;

    int sockfd = socket(AF_INET, SOCK_STREAM, 0);//domain = IPv4, type = TCP, protocol = IP
    if(sockfd < 0)
        perror("Error: Unable to create socket");

    //Filling server_address with null bytes
    bzero(&server_address, sizeof(server_address));

    server_address.sin_family = AF_INET;// Uses the Internet address family
    server_address.sin_addr.s_addr = INADDR_ANY;// Use any of the available addresses
    server_address.sin_port = htons(4500);// Connect to specified port 4500

    //Bind socket to the specified port
    if(bind(sockfd, (struct sockaddr*)&server_address, sizeof(server_address))<0)
        perror("Bind error");

    //Look for clients to serve, with a maximum limit of 5.
    listen(sockfd, 5);

    //New socket file descriptor to handle connections.
    int len = sizeof(client_address);
    while(1){
```

```
    //Clears socket set
    FD_ZERO(&clientfds);

    //Add main socket to the set
    FD_SET(sockfd, &clientfds);
    int max_sd = sockfd;

    //Adding valid secondary sockets to the set
    for(int i = 0;i < 30;i++){
        int sd = client_sockets[i];
        //Checking validity
        if(sd > 0)
            FD_SET(sd, &clientfds);


        //Store highest valued file descriptor
        if(sd > max_sd)
            max_sd = sd;
    }

    //Wait indefinitely for action on one of the sockets
    int action = select(max_sd + 1, &clientfds, NULL, NULL, NULL);
    if(action<0){
        perror("\nSelect error!\n");
    }

    //A change in main socket descriptor value implies that it is an incoming connection request
    if(FD_ISSET(sockfd, &clientfds)){
        newfd = accept(sockfd, (struct sockaddr*)&client_address, &len);
        if(newfd < 0)
            perror("\nUnable to accept new connection.\n");

        printf("\nNew connection established.\nSocket descriptor:%d", newfd);


        strcpy(buffer, "Connection established");
        write(newfd, buffer, sizeof(buffer));

        //Add new client socket to list of sockets
        for(int i =0;i<30;i++){
                if(client_sockets[i] == 0){
                    client_sockets[i] = newfd;
                    printf("\nConnection at socket %d\n", i);
                    break;
                }
            }
        }
        //I/O operation on an established connection
        for(int i = 0;i<30;i++){
            int sd = client_sockets[i];
            //Check for change in ddescriptors
            if(FD_ISSET(sd, &clientfds)){
                read(sd, buffer, sizeof(buffer));
                //Check end of connection
                if(strcmp(buffer, "end") == 0){
                    getpeername(sd, (struct sockaddr*)&client_address,&len);
                    printf("\nHost disconnected. Socket: %d.\n",client_sockets[i]);
                    close(sd);
                    client_sockets[i] = 0;
                }
                else{
                    printf("\nMessage from Client %d: %s\n", client_sockets[i], buffer);

                    bzero(buffer, sizeof(buffer));
                    //Write message in buffer
                    printf("\nEnter message: ");scanf(" %[^\n]", buffer);
                    write(newfd, buffer, sizeof(buffer));
                }

            }
        }

    }

    return 0;
}
```

Client:

```c
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<unistd.h>
#include<sys/time.h>

int main(int argc,char** argv){
    //Server and client addresses
    struct sockaddr_in server_address, client_address;
    //Buffer to handle messages
    char buffer[1024];

    //Server socket file descriptor
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);//(domain = Ipv4, type = TCP, protocol = 0
    if(sockfd < 0)
        perror("Error: Unable to create socket");
    //Filling server address with null bytes
    bzero(&server_address, sizeof(server_address));

    server_address.sin_family = AF_INET;//Use the Internet address family
    server_address.sin_addr.s_addr = inet_addr(argv[1]);//Use ip address passed as command line argument
    server_address.sin_port = htons(4500);//Connect socket to port 4500

    //Attempt to connect client to socket on specified port
    connect(sockfd, (struct sockaddr*)&server_address, sizeof(server_address));

    int len = sizeof(client_address);
    //Chat session ends if buffer contains "end"
    while(strcmp(buffer, "end") != 0){
        bzero(buffer, sizeof(buffer));
        read(sockfd, buffer, sizeof(buffer));
        printf("\nMessage from Server: %s", buffer);

        bzero(buffer, sizeof(buffer));
        //Write message in buffer
        printf("\nEnter the message: ");scanf(" %[^\n]", buffer);
        write(sockfd, buffer, sizeof(buffer));
    }
    printf("\nSession end");
    close(sockfd);
    return 0;
}
```

Output:

Server:

```
New connection established.
Socket descriptor:4
Connection at socket 0

Message from Client 4: hi

Enter message: hello client 4

New connection established.
Socket descriptor:5
Connection at socket 1

Message from Client 5: hi server

Enter message: hlo client 5

Message from Client 5: ssn

Enter message: cse

Message from Client 4: department

Enter message: end

Host disconnected. Socket: 5.
Host disconnected. Socket: 4.
```

Client 1:

```
Message from Server: Connection established
Enter the message: hi

Message from Server: hello client 4
Enter the message: department
end

Message from Server: end
Enter the message: end
Session end
```

Client 2:

```
Message from Server: Connection established
Enter the message: hi server

Message from Server: hlo client 5
Enter the message: ssn

Message from Server: cse
Enter the message: end

Session end
```