

SSN College of Engineering

UCS1511 - Networks Lab

Exercise 3 – Chat using TCP

Prasanna Kumaran D
185001110

October 3, 2020

1 Chat using TCP

1.1 Aim

To write a socket program to perform chat with multiple clients.

1.2 Algorithm

1.2.1 Server

1. Create a socket descriptor with **socket()** system call and use `AF_INET` as domain and `SOCK_STREAM` for domain and communication type, store the socket descriptor in `sockfd`. Initialize client file descriptors to 0
2. If `sockfd` is a negative number,
 - (a) print socket creation failed, terminate program.
3. Assign family, address and port to the server `socketadd_in` object. Set the family as `AF_INET` to access IPv4 protocols, and `INADDR_ANY` for address to accept connections from any client.
4. bind the socket to the server `sockadd_in` object
5. If bind is non zero,
 - (a) Print bind creation failed and terminate.
6. While there are open FDS
 - (a) Initialize file descriptor set using **FD_ZERO**

- (b) Add server socket FD to FD set using **FD_SET**
 - (c) Looping through the client list
 - if `client_fd > 0` then add it to FD set
 - if `client_fd > max(client_fds)` make `client_fd` maximum
 - (d) Use select system call to monitor multiple file descriptors and trigger once a file descriptor becomes ready.
 - (e) If server socket is present in file descriptor set (using **FD_ISSET**)
 - i. Accept the client request
 - ii. Iterate through the client list. Assign the retrieved FD to the first client whose fd is 0.
 - (f) loop through client list
 - i. if client fd found in FD set, read data from the client using (**read()**) and write data from the server onto the buffer using (**write**)
7. Close socket connections using **close()** and terminate program.

1.2.2 Client

1. Create a socket descriptor using **socket()**
with `AF_INET`(IPv4 domain), `SOCK_STREAM`(connection type)
2. if `socket < 0`
 - (a) Print socket creation failed and terminate program
3. Create a `sockaddr_in` object for the client and set up family, address and port number.
4. Call `connect()` to establish connection between client and server
5. While there are open FDs
 - Read (**read()**) the user message and write (**write()**) into the buffer
 - Display the message received from the server
 - Close the socket connection using **close()** and terminate program
6. Close socket connection using **close()** and terminate program

1.3 Program

1.3.1 Server

```
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<arpa/inet.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
int main(int argc, char **argv)
{

    int len;
    int server_socket , client[10], n = 0;
    struct sockaddr_in servaddr, cliaddr;
    fd_set descriptors;
    char buff[1024];
    char str[100];
    server_socket = socket(AF_INET, SOCK_STREAM, 0);
    for(int i = 0; i < 10; i++) client[i] = 0;
    if(server_socket < 0)
        perror("cannot create socket");
    // Writes 0 to the contents from the starting address
    bzero(&servaddr, sizeof(servaddr));

    // Intialize server structure
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(8073);

    if(bind(server_socket, (struct sockaddr*)&servaddr, sizeof(servaddr)) < 0)
        perror("Bind error");

    printf("Waiting for client \n");

    listen(server_socket, 10);
    len = sizeof(servaddr);
    int sel_fd;
    while(1)
    {
        FD_ZERO(&descriptors);
```

```

FD_SET(server_socket , &descriptors);
sel_fd = server_socket;
for(int i = 0; i < 10; i++){
    int check = client[i];
    if (check > 0)
        FD_SET(check , &descriptors);
    if (check > sel_fd)
        sel_fd = check;
}
int newfd = select(sel_fd + 1, &descriptors , NULL , NULL, NULL);
if (FD_ISSET(server_socket , &descriptors))
{
    int clientfd = accept(server_socket ,
        (struct sockaddr*)&cliaddr , &len);
    for(int i = 0; i < 10; i++){
        if (client[i] == 0){
            client[i] = clientfd;
            break;
        }
    }
}
for (int i = 0; i < 10; i++){
    if(FD_ISSET(client[i] , &descriptors))
    {
        //Receiving the message
        read(client[i] , buff , sizeof(buff));
        if (buff[0] == '^')
        {
            close(client[i]);
            client[i] = 0;
            printf("Client %d terminated\n",i);
        }
        else {
            printf("\nMessage from Client %d:\t%s",i , buff);
            printf("\nEnter message to be sent:");
            scanf("%s",str);
            strcpy(buff , str);
            n = write(client[i] , buff , sizeof(buff));
            printf("\nMessage Sent:\t%s\n",buff);
        }
    }
}

close(server_socket);
return 0;

```

```
}
```

1.3.2 Client

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<string.h>
int main(int argc, char **argv)
{

    int len;
    int clientfd, n = 0;
    struct sockaddr_in servaddr;
    char str[1000];
    char buff[1024];
    clientfd = socket(AF_INET, SOCK_STREAM, 0);
    if(clientfd < 0)
        perror("cannot create socket!\n");

    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family=AF_INET;
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port=htons(8073);
    connect(clientfd, (struct sockaddr*)&servaddr, sizeof(servaddr));

    //Sending Message
    while(1)
    {
        printf("Your message:");
        scanf("%s", buff);
        printf("\nClient:%s", buff);
        n = write(clientfd, buff, sizeof(buff));
        n = read(clientfd, buff, sizeof(buff));
        printf("\nMessage from Server:%s\n", buff);
    }
    close(clientfd);
    return 0;
}
```

1.4 Output

```
legion@Legion: ~/Desktop/Networks Lab/Assignment3
File Edit View Search Terminal Help
legion@Legion:~$ cd Desktop/Networks\ Lab/Assignment3/
legion@Legion:~/Desktop/Networks Lab/Assignment3$ gcc client_chat.c -o c
legion@Legion:~/Desktop/Networks Lab/Assignment3$ gcc server_chat.c -o s
legion@Legion:~/Desktop/Networks Lab/Assignment3$ ./s
Waiting for client

Message from Client 0: GoodMorning!Max_here
Enter message to be sent:Hello_max

Message Sent: Hello_max

Message from Client 1: Hiiii!Genny_here
Enter message to be sent:Welcome_Genny

Message Sent: Welcome_Genny

Message from Client 2: Toby_here_to party!!!
Enter message to be sent:Yes_toby!!!We_will!!!

Message Sent: Yes_toby!!!We_will!!!

Message from Client 0: ByeServer!!!
Enter message to be sent:Bye_Max

Message Sent: Bye_Max

Message from Client 1: I'm_SigningOFF
Enter message to be sent:Byeee_Genny

Message Sent: Byeee_Genny

Message from Client 2: I'm_leaving_too
Enter message to be sent:Goodbye_Toby

Message Sent: Goodbye_Toby
^Z
[1]+  Stopped                  ./s
legion@Legion:~/Desktop/Networks Lab/Assignment3$
```

Figure 1: Server side

```
legion@Legion: ~/Desktop/Networks Lab/Assignment3
File Edit View Search Terminal Help
legion@Legion:~/Desktop/Networks Lab/Assignment3$ ./c
Your message:GoodMorning!Max_here

Client:GoodMorning!Max_here
Message from Server:Hello_max
Your message:ByeServer!!!

Client:ByeServer!!!
Message from Server:Bye_Max
Your message:
```

Figure 2: Client side - Client 1

```
legion@Legion: ~/Desktop/Networks Lab/Assignment3
File Edit View Search Terminal Help
legion@Legion:~/Desktop/Networks Lab/Assignment3$ ./c
Your message:Hiiii!Genny_here

Client:Hiiii!Genny_here
Message from Server:Welcome_Genny
Your message:I'm_SigningOFF

Client:I'm_SigningOFF
Message from Server:Byeee_Genny
Your message:
```

Figure 3: Client side - Client 2

```
legion@Legion: ~/Desktop/Networks Lab/Assignment3
File Edit View Search Terminal Help
legion@Legion:~/Desktop/Networks Lab/Assignment3$ ./c
Your message:Toby_here_to_party!!!

Client:Toby_here_to_party!!!
Message from Server:Yes_toby!!!We_will!!!
Your message:I'm_leaving_too

Client:I'm_leaving_too
Message from Server:Goodbye_Toby
Your message:
```

Figure 4: Client side - Client 3

2 Learning Outcomes

- Learnt how to establish a simple client server connection using TCP
- Learnt about the basic syntax and system calls used in socket programming
- Learnt how to handle errors in socket programming
- Learnt how to handle multiple client requests
- Learnt to use select system call which can be used with multiple client requests