CS6111 - COMPUTER NETWORKS LAB 2

SOCKET PROGRAMMING

Name : Ajitesh M

Reg. No : 2019103503

Aim :

To build a chat application using socket programming, handling multiple clients.

Code :

Chatroom_Server.c

```
--------------------------------------------------------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <pthread.h>

#define PORT 3503
#define MAX 256
#define N 5

int client_sockets[N];

void *manageClients(void *client_socket)
{
    int c_socket = *((int *)client_socket);
    int cur_index = 0;
    char buffer[2 * MAX];
    char msg[MAX];
    for (int i = 0; i < N; i++)
    {
        if (client_sockets[i] == c_socket)
        {
            cur_index = i;
            break;
        }
    }
    printf("[+] New client %d connected.\n", cur_index);
    char *welcome = "Welcome to chatroom!";
    send(c_socket, welcome, strlen(welcome), 0);
    while (1)
    {
        msg[0] = '\0';
        int len = recv(c_socket, msg, sizeof(msg), 0);
```

```
        msg[len] = '\0';
        if (strlen(msg) > 0)
        {
            printf("From client %d : %s\n", cur_index, msg);
            if (strncmp(msg, "EXIT", 4) == 0)
            {
                printf("[+] Client %d disconnected.\n", cur_index);
                char *term = "TERM";
                send(c_socket, term, strlen(term), 0);
                close(c_socket);
                client_sockets[cur_index] = 0;
                return NULL;
            }
            else {
                for (int i = 0; i < N; i++)
                {
                    if (client_sockets[i] != 0)
                    {
                        buffer[0] = '\0';
                        if (i == cur_index)
                        {
                            sprintf(buffer, "SENT : %s", msg);
                            send(client_sockets[i], buffer, strlen(buffer), 0);
                            continue;
                        }
                        else
                        {
                            sprintf(buffer, "From client %d : %s", cur_index, msg
);
                            send(client_sockets[i], buffer, strlen(buffer), 0);
                            printf("Sent to client %d.\n", i);
                        }
                    }
                }
            }
        }
    }
    return NULL;
}

int main()
{
    pthread_t client_threads[N];
    int serverSocket;
    struct sockaddr_in serverAddr;
    serverSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (serverSocket < 0)
    {
        printf("[-] Socket creation failed.\n");
        exit(1);
    }
    int opt = 1;
```

```
    if (setsockopt(serverSocket, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt, s
izeof(opt)) < 0)
    {
        printf("[-] Socket set option failed.\n");
        exit(1);
    }
    bzero(&serverAddr, sizeof(serverAddr));
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(PORT);
    serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    if (bind(serverSocket, (struct sockaddr *)&serverAddr, sizeof(serverAddr)) <
0)
    {
        printf("[-] Socket bind failed.\n");
        exit(1);
    }
    if (listen(serverSocket, N) < 0)
    {
        printf("[-] Socket listen failed.\n");
        exit(1);
    }
    printf("[+] Server running and listening on port %d.\n", PORT);
    for (int i = 0; i <= N; i++)
        client_sockets[i] = 0;

    int index = N;
    while (1)
    {
        for (int i = 0; i < N; i++)
        {
            if (client_sockets[i] == 0)
            {
                index = i;
                break;
            }
        }
        if (index == N)
            continue;
        client_sockets[index] = accept(serverSocket, NULL, NULL);

        if (client_sockets[index] < 0)
        {
            printf("[-] Unable to accept client request.\n");
            break;
        }
        pthread_create(&client_threads[index], NULL, manageClients, (void *)&clie
nt_sockets[index]);
    }
    for (int i = 0; i < N; i++)
        pthread_join(client_threads[i], NULL);
    close(serverSocket);
    return 0;
}
```

Explanation :

In order to handle multiple client connections, threads are used. Unlike creating a child process for each client using fork(), threads are more resource efficient and can be created to execute a particular sub-routine. Here, one thread for each client is created to execute the manageClients() routine which receives an incoming message from a client and sends the message to other clients.

Chatroom_Client.c

```
--------------------------------------------------------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <pthread.h>

#define PORT 3503
#define MAX 1024

int client_socket;

void *receiveMessage(void *client_socket)
{
    int socket = *((int *)client_socket);
    char buffer[MAX];
    while (1)
    {
        buffer[0] = '\0';
        int len = recv(socket, buffer, sizeof(buffer), 0);
        buffer[len] = '\0';
        if (strlen(buffer) > 0)
        {
            if (strncmp(buffer, "TERM", 4) == 0)
                break;
            printf("%s \n", buffer);
        }
    }
    return NULL;
}
int main()
{
    pthread_t client_thread;

    struct sockaddr_in serverAddr;
    client_socket = socket(AF_INET, SOCK_STREAM, 0);

    bzero(&serverAddr, sizeof(serverAddr));
    serverAddr.sin_family = AF_INET;
```

```
    serverAddr.sin_port = htons(PORT);
    serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

    if (connect(client_socket, (struct sockaddr *)&serverAddr,
sizeof(serverAddr)) < 0)
    {
        printf("[-] Connection request failed.\n");
        exit(1);
    }
    printf("[+] Connected to the server.\n");

    pthread_create(&client_thread, NULL, receiveMessage, (void *)&client_socket);

    char buffer[MAX];
    while (1)
    {
        buffer[0] = '\0';
        int n = 0;
        while ((buffer[n++] = getchar()) != '\n')
            ;
        buffer[n] = '\0';
        if (strlen(buffer) > 0)
        {
            send(client_socket, buffer, strlen(buffer), 0);
            if (strncmp(buffer, "EXIT", 4) == 0)
            {
                printf("[+] Exiting..\n");
                break;
            }
        }
    }
    pthread_join(client_thread, NULL);
    close(client_socket);
    return 0;
}
```

Explanation :

Here, send() and recv() system calls being blocking in nature, a separate thread is created for handling the incoming message from the server. Hence, the user can send message and also receive a message at the same time.

Output :

```
[s2019103503@centos8-linux Sat Sep 11 09:16 AM Week2]$ gcc Chatroom_Server.c -o Cha
[s2019103503@centos8-linux Sat Sep 11 09:16 AM Week2]$ ./Chatroom Server
[+] Server running and listening on port 3503.
[+] New client 0 connected.
[+] New client 1 connected.
[+] New client 2 connected.
From client 1 : Hiii

Sent to client 0.
Sent to client 2.
From client 2 : How are you ?

Sent to client 0.
Sent to client 1.
From client 1 : I'm good

Sent to client 0.
Sent to client 2.
From client 0 : Welcome

Sent to client 1.
Sent to client 2.
From client 2 : Great.. Catch u up later..Byee

Sent to client 0.
Sent to client 1.
From client 2 : EXIT

[+] Client 2 disconnected.
```

```
[s2019103503@centos8-linux Sat Sep 11 09:16 AM Week2]$ gcc Chatroom_Client.c -o Ch
[s2019103503@centos8-linux Sat Sep 11 09:17 AM Week2]$ ./Chatroom Client
[+] Connected to the server.
Welcome to chatroom!
From client 1 : Hiii

From client 2 : How are you ?

From client 1 : I'm good

Welcome
SENT : Welcome

From client 2 : Great.. Catch u up later..Byee
```

```
[s2019103503@centos8-linux Sat Sep 11 09:19 AM Week2]$ ./Chatr
oom_Client
[+] Connected to the server.
Welcome to chatroom!
Hiii
SENT : Hiii

From client 2 : How are you ?

I'm good
SENT : I'm good

From client 0 : Welcome

From client 2 : Great.. Catch u up later..Byee
```

```
[s2019103503@centos8-linux Sat Sep 11 09:19 AM Week2]$ ./Chatroom_Clien
[+] Connected to the server.
Welcome to chatroom!
From client 1 : Hiii

How are you ?
SENT : How are you ?

From client 1 : I'm good

From client 0 : Welcome

Great.. Catch u up later..Byee
SENT : Great.. Catch u up later..Byee

EXIT
[+] Exiting..
[s2019103503@centos8-linux Sat Sep 11 09:21 AM Week2]$
```