COMPUTER NETWORKS

SAKSHAM KUMAR

COMPUTER SCIENCE AND ENGINEERING
ID - 2022UCP1700
SECTION- A4

ASSIGNMENT – 2

Server Implementation:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <pthread.h>

#define PORT 8080
#define BUFFER_SIZE 1024

void *handle_client(void *arg);

int main(void)
{
    int server_socket, client_socket;
    struct sockaddr_in server_address, client_address;
    char buffer[BUFFER_SIZE] = {0};
    pthread_t thread_id;

    // Create socket
    if ((server_socket = socket(AF_INET, SOCK_STREAM, 0)) == -1)
    {
        perror("Socket creation error");
        exit(EXIT_FAILURE);
    }

    // Set up server address structure
    server_address.sin_family = AF_INET;
    server_address.sin_addr.s_addr = INADDR_ANY;
    server_address.sin_port = htons(PORT);

    // Bind the socket to the specified address and port
    if (bind(server_socket, (struct sockaddr *)&server_address,
sizeof(server_address)) == -1)
    {
        perror("Bind failed");
        exit(EXIT_FAILURE);
    }
```

```c
void *handle_client(void *arg)
{
    int client_socket = *((int *)arg);
    char buffer[BUFFER_SIZE] = {0};

    // Receive and send messages
    while (1)
    {
        memset(buffer, 0, sizeof(buffer));

        // Receive message from client
        if (recv(client_socket, buffer, sizeof(buffer), 0) == -1)
        {
            perror("Receive failed");
            close(client_socket);
            pthread_exit(NULL);
        }

        if (strcmp(buffer, "exit") == 0)
        {
            printf("Client disconnected\n");
            close(client_socket);
            pthread_exit(NULL);
        }

        printf("Client: %s\n", buffer);

        // Send message to client
        printf("Server: ");
        fgets(buffer, sizeof(buffer), stdin);
        buffer[strlen(buffer) - 1] = '\0'; // Remove newline character

        if (send(client_socket, buffer, strlen(buffer), 0) == -1)
        {
            perror("Send failed");
            close(client_socket);
            pthread_exit(NULL);
        }
```

```
        if (strcmp(buffer, "exit") == 0)
        {
            printf("Server shutting down\n");
            close(client_socket);
            pthread_exit(NULL);
        }
    }
}
```

Client Implementation:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<arpa/inet.h>

#define PORT 8080
#define BUFFER_SIZE 1024

int main(void)
{
    int client_socket;
    struct sockaddr_in server_address;
    char buffer[BUFFER_SIZE] = {0};

    //Create socket
    if((client_socket=socket(AF_INET, SOCK_STREAM,0))==-1)
    {
        perror("Socket creation error");
        exit(EXIT_FAILURE);
    }

    //Set up server address structure
    server_address.sin_family=AF_INET;
    server_address.sin_port=htons(PORT);

    //Connect IPv4 and IPv6 addresses from text to binary form
    if(inet_pton(AF_INET,"192.168.224.131",&server_address.sin_addr)<=0)
    {
```

```
        perror("Invalid address/Address not supported");
        exit(EXIT_FAILURE);
    }

    //Connect to server
    if(connect(client_socket,(struct
sockaddr*)&server_address,sizeof(server_address))==-1)
    {
        perror("Connection failed");
        exit(EXIT_FAILURE);
    }

    printf("Connected to server\n");

    //Recieve and send messages
    while(1)
    {
        printf("Client: ");
        fgets(buffer,sizeof(buffer),stdin);
        buffer[strlen(buffer)-1]='\0';//Remove newline character

        //Send message to server
        if(send(client_socket,buffer,strlen(buffer),0)==-1)
        {
            perror("Send failed");
            exit(EXIT_FAILURE);
        }

        if(strcmp(buffer,"exit")==0)
        {
            printf("Client disconnecting\n");
            break;
        }

        memset(buffer,0,sizeof(buffer));

        //Recieve message from server
        if(recv(client_socket,buffer,sizeof(buffer),0)==-1)
        {
            perror("Recieve failed");
```

```
            exit(EXIT_FAILURE);
        }

        if(strcmp(buffer,"exit")==0)
        {
            printf("Server disconnected\n");
            break;
        }

        printf("Server: %s\n",buffer);
    }

    //Close sockets
    close(client_socket);

    return 0;
}
```

Testing the Connection:

```
Server listening on port 8080...
Client connected
Client connected
Client: Hi
Server: hello
Client: Hello
Server: Client disconnected
bakalakapupu
Client: Hi
Server: hihi hahah
Client disconnected
exit
```

```
Connected to server
Client: Hello
Server: Hello! You are now connected to the server.

Client: Hi
Server: bakalakapupu
Client: exit
Client disconnecting
```

THE END

SAKSHAM KUMAR
COMPUTER SCIENCE AND ENGINEERING
ID - 2022UCP1700
SECTION- D