COMPUTER NETWORKS

SAKSHAM KUMAR

COMPUTER SCIENCE AND ENGINEERING
ID - 2022UCP1700
SECTION- A4

ASSIGNMENT – 1

Server Implementation:

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

#dene PORT 8080
#dene BUFFER_SIZE 1024

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

        // 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_amily = AF_INET;
        server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
        server_address.sin_port = htons(PORT);


        if (bind(server_socket, (struct sockaddr*)&server_address,
        sizeo(server_address)) == -1) {
                perror("Bind ailed");
                exit(EXIT_FAILURE);
        }

        // Listen or incoming connections
        if (listen(server_socket, 3) == -1) {
                perror("Listen ailed");
                exit(EXIT_FAILURE);
        }
        print("Server listening on port%d...\n", PORT);
```

```
        // Accept incoming connection
        socklen_t client_address_len = sizeo(client_address);
        if ((client_socket = accept(server_socket, (struct sockaddr*)&client_address,
        &client_address_len)) == -1) {
                perror("Accept ailed");
                exit(EXIT_FAILURE);
        }
        print("Client connected\n");

        // Receive and send messages
        while (1) {
                memset(bufer, 0, sizeo(bufer));
                if (recv(client_socket, bufer, sizeo(bufer), 0) == -1) {
                        perror("Receive ailed");
                        exit(EXIT_FAILURE);
                }
                if (strcmp(bufer, "exit") == 0) {
                        print("Client disconnected\n");
                        break;
                }
                print("Client:%s\n", bufer);

                // Sendmessage to client
                print("Server: ");
                gets(bufer, sizeo(bufer), stdin);
                bufer[strlen(bufer) - 1] = '\0';
                if (send(client_socket, bufer, strlen(bufer), 0) == -1) {
                        perror("Send ailed");
                        exit(EXIT_FAILURE);
                }
                if (strcmp(bufer, "exit") == 0) {
                        print("Server shutting down\n");
                        break;
                }
        }
        // Close sockets
        close(client_socket);
        close(server_socket);
        return 0;
}
```

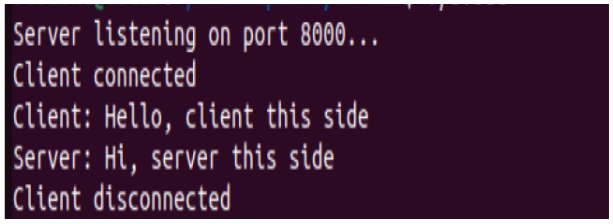
Client Implementation:

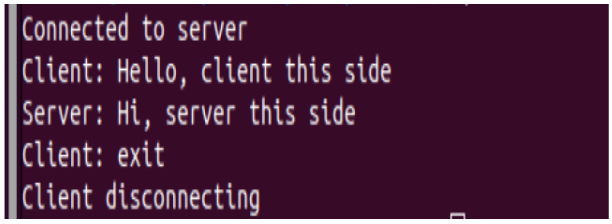
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#dene PORT 8080
#dene BUFFER_SIZE 1024

int main() {
        int client_socket;
        struct sockaddr_in server_address;
        char bufer[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_amily = AF_INET;
        server_address.sin_port = htons(PORT);
        // Convert IPv4 and IPv6 addresses romtext to binary orm
        if (inet_pton(AF_INET, "127.0.0.1", &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,
        sizeo(server_address)) == -1)
        {
                perror("Connection ailed");
                exit(EXIT_FAILURE);
        }
        print("Connected to server\n");

        // Receive and sendmessages
        while (1) {
                print("Client: ");
                gets(bufer, sizeo(bufer), stdin);
                bufer[strlen(bufer) - 1] = '\0';
```

```
        // Sendmessage to server
        if (send(client_socket, bufer, strlen(bufer), 0) == -1) {
            perror("Send ailed");
            exit(EXIT_FAILURE);
        }
        if (strcmp(bufer, "exit") == 0) {
            print("Client disconnecting\n");
            break;
        }
        memset(bufer, 0, sizeo(bufer));

        // Receivemessage from server
        if (recv(client_socket, bufer, sizeoð](bufer), 0) == -1) {
            perror("Receive ð]ailed");
            exit(EXIT_FAILURE);
        }
        if(strcmp(bufer, "exit") == 0) {
            printf("Server disconnected\n");
        break;
        }
        printf("Server:%s\n", bufer);
    }

    // Close socket
    close(client_socket);
    return 0;
}
```

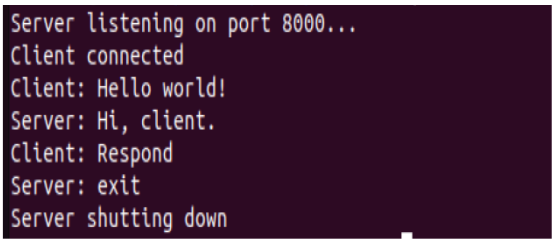
Testing the Connection:

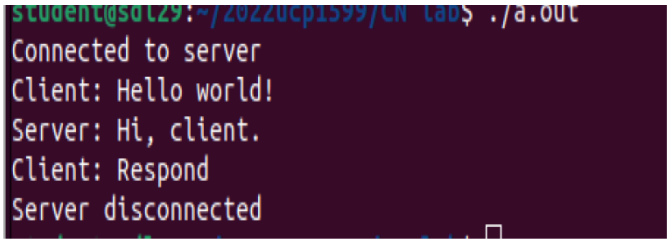
1. **Client exit**



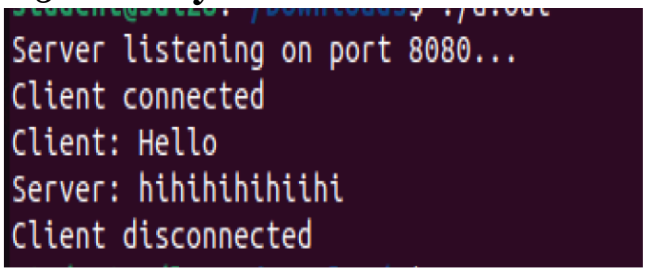


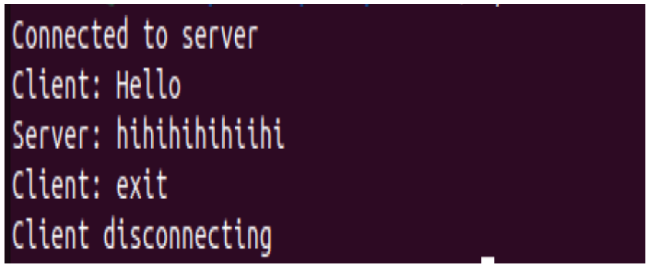
2. **Server exit**





3. **Two systems:**





THE END

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