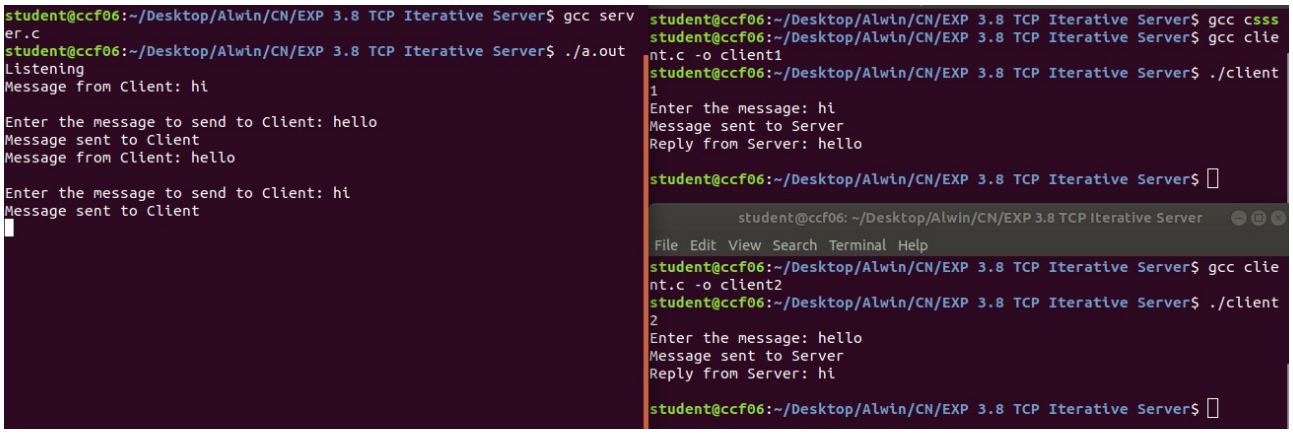
CLIENT

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

#define PORT 2000
#define BUFFER_SIZE 1024

int main() {
    int clientSocket;
    char buffer[BUFFER_SIZE];
    struct sockaddr_in serverAddr;
    socklen_t addr_size;

    clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (clientSocket < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(PORT);
    serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);
    addr_size = sizeof serverAddr;

    if (connect(clientSocket, (struct sockaddr*)&serverAddr, addr_size) < 0) {
        perror("Connection failed");
        close(clientSocket);
        exit(EXIT_FAILURE);
    }

    printf("Enter the message: ");
    fgets(buffer, BUFFER_SIZE, stdin);
    send(clientSocket, buffer, strlen(buffer), 0);
    printf("Message sent to Server\n");

    memset(buffer, 0, BUFFER_SIZE);
    int recv_len = recv(clientSocket, buffer, BUFFER_SIZE, 0);
    if (recv_len > 0) {
        printf("Reply from Server: %s\n", buffer);
    } else {
        perror("Receive failed");
    }

    close(clientSocket);
    return 0;
}
```

SERVER

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

#define PORT 2000
#define BUFFER_SIZE 1024

int main() {
    int welcomeSocket, newSocket;
    char buffer[BUFFER_SIZE];
    char reply[BUFFER_SIZE];
    struct sockaddr_in serverAddr;
    struct sockaddr_storage serverStorage;
    socklen_t addr_size;

    welcomeSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (welcomeSocket < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(PORT);
    serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);

    if (bind(welcomeSocket, (struct sockaddr *)&serverAddr, sizeof(serverAddr)) < 0) {
        perror("Bind failed");
        close(welcomeSocket);
        exit(EXIT_FAILURE);
    }

    if (listen(welcomeSocket, 5) == 0)
        printf("Listening\n");
    else {
        perror("Listen failed");
        close(welcomeSocket);
        exit(EXIT_FAILURE);
    }

    while (1) {
        addr_size = sizeof serverStorage;
        newSocket = accept(welcomeSocket, (struct sockaddr *)&serverStorage, &addr_size);
        if (newSocket < 0) {
            perror("Accept failed");
            continue;
        }
        memset(buffer, 0, BUFFER_SIZE);
        int recv_len = recv(newSocket, buffer, BUFFER_SIZE, 0);
        if (recv_len > 0) {
            printf("Message from Client: %s\n", buffer);

            // Get reply message from server user
            printf("Enter the message to send to Client: ");
            fgets(reply, BUFFER_SIZE, stdin);
            send(newSocket, reply, strlen(reply) + 1, 0);
            printf("Message sent to Client\n");
        } else {
            perror("Receive failed");
        }

        close(newSocket);
    }

    close(welcomeSocket);
    return 0;
}
```



```
student@ccf06:~/Desktop/Alwin/CN/EXP 3.8 TCP Iterative Server$ gcc serv
er.c
student@ccf06:~/Desktop/Alwin/CN/EXP 3.8 TCP Iterative Server$ ./a.out
Listening
Message from Client: hi

Enter the message to send to Client: hello
Message sent to Client
Message from Client: hello

Enter the message to send to Client: hi
Message sent to Client
```

```
student@ccf06:~/Desktop/Alwin/CN/EXP 3.8 TCP Iterative Server$ gcc csss
student@ccf06:~/Desktop/Alwin/CN/EXP 3.8 TCP Iterative Server$ gcc clie
nt.c -o client1
student@ccf06:~/Desktop/Alwin/CN/EXP 3.8 TCP Iterative Server$ ./client
1
Enter the message: hi
Message sent to Server
Reply from Server: hello

student@ccf06:~/Desktop/Alwin/CN/EXP 3.8 TCP Iterative Server$ 
```

student@ccf06: ~/Desktop/Alwin/CN/EXP 3.8 TCP Iterative Server
File Edit View Search Terminal Help

```
student@ccf06:~/Desktop/Alwin/CN/EXP 3.8 TCP Iterative Server$ gcc clie
nt.c -o client2
student@ccf06:~/Desktop/Alwin/CN/EXP 3.8 TCP Iterative Server$ ./client
2
Enter the message: hello
Message sent to Server
Reply from Server: hi

student@ccf06:~/Desktop/Alwin/CN/EXP 3.8 TCP Iterative Server$ 
```