Pradeep KM
3122225001092

Exercise 4: Chat using TCP

Server algorithm

9. Create socket.
10. Bind socket to port.
11. Listen for client connection.
12. Accept client connection.
13. Receive filename from client.
14. Open file for reading.
15. Send file data in chunks.
16. Close file and socket.

Client algorithm

7. Create socket.
8. Connect to server.
9. Input and send filename.
10. Receive file data.
11. Write data to new file.
12. Close socket.

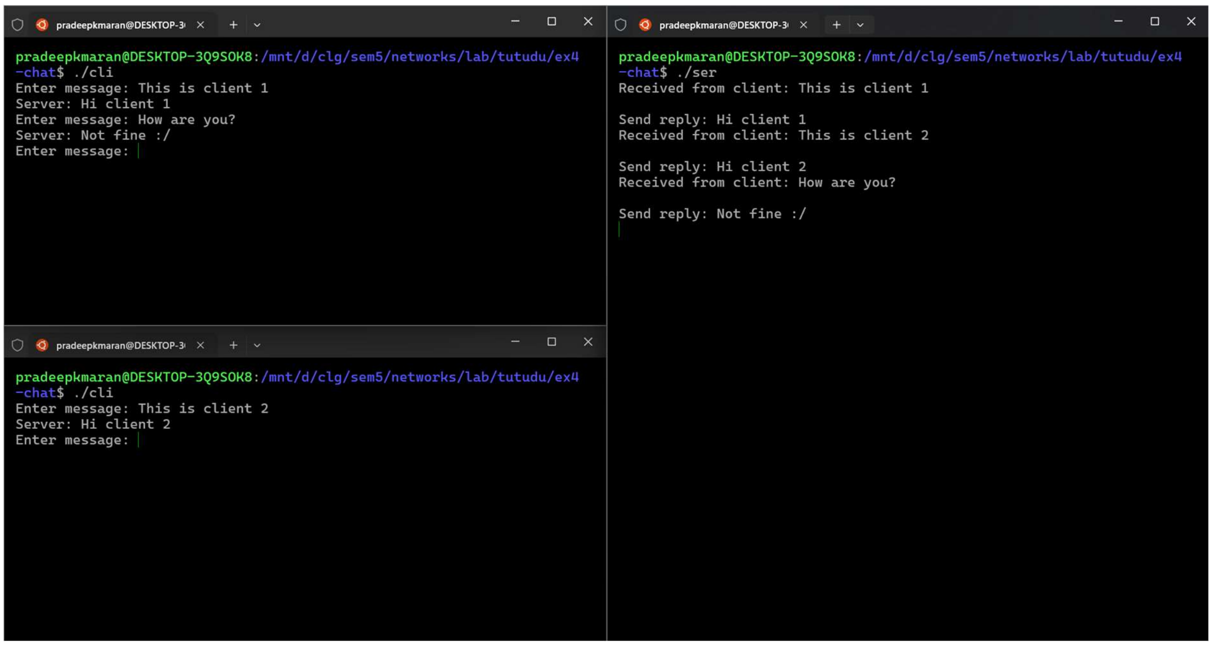
server.c

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

#define PORT 8080
#define MAX_CLIENTS 10
#define BUF_SIZE 1024

void handle_client(int client_socket) {
    char buffer[BUF_SIZE];
    int n;

    while ((n = read(client_socket, buffer, sizeof(buffer))) > 0) {
        buffer[n] = '\0';
        printf("Received from client: %s\n", buffer);

        char user_input[BUF_SIZE];
        printf("Send reply: ");
        fgets(user_input, BUF_SIZE-1, stdin);
        user_input[strcspn(user_input, "\n")] = '\0';
        write(client_socket, user_input, strlen(user_input));
    }
    close(client_socket);
}

int main() {
    int server_socket, client_socket, client_len;
    struct sockaddr_in server_addr, client_addr;
    pid_t child_pid;

    server_socket = socket(AF_INET, SOCK_STREAM, 0);
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);

    bind(server_socket, (struct sockaddr*)&server_addr, sizeof(server_addr));
    listen(server_socket, MAX_CLIENTS);

    while (1) {
        client_len = sizeof(client_addr);
        client_socket = accept(server_socket, (struct sockaddr*)&client_addr, &client_len);

        if ((child_pid = fork()) == 0) {
            close(server_socket);
            handle_client(client_socket);
            exit(0);
        } else {
            close(client_socket);
        }
    }

    return 0;
}
```

client.c

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

#define PORT 8080
#define BUF_SIZE 1024

int main() {
    int sock;
    struct sockaddr_in server_addr;
    char buffer[BUF_SIZE];
    ssize_t n;

    sock = socket(AF_INET, SOCK_STREAM, 0);
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    connect(sock, (struct sockaddr*)&server_addr, sizeof(server_addr));

    while (1) {
        printf("Enter message: ");
        fgets(buffer, sizeof(buffer), stdin);
        write(sock, buffer, strlen(buffer));

        n = read(sock, buffer, sizeof(buffer));
        buffer[n] = '\0';
        printf("Server: %s\n", buffer);
    }

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

Output



```
pradeepkmaran@DESKTOP-3Q9SOK8:/mnt/d/clg/sem5/networks/lab/tutudu/ex4
-chat$ ./cli
Enter message: This is client 1
Server: Hi client 1
Enter message: How are you?
Server: Not fine :/
Enter message:
```

```
pradeepkmaran@DESKTOP-3Q9SOK8:/mnt/d/clg/sem5/networks/lab/tutudu/ex4
-chat$ ./cli
Enter message: This is client 2
Server: Hi client 2
Enter message:
```

```
pradeepkmaran@DESKTOP-3Q9SOK8:/mnt/d/clg/sem5/networks/lab/tutudu/ex4
-chat$ ./ser
Received from client: This is client 1

Send reply: Hi client 1
Received from client: This is client 2

Send reply: Hi client 2
Received from client: How are you?

Send reply: Not fine :/
```