//tcp_server.c

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

#define MAX 80
#define PORT 9734

int main() {
int server_sockfd, client_sockfd;
char command[MAX], response[MAX];
struct sockaddr_in server_addr, client_addr;
socklen_t client_len;

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

// 2. Initialize server address structure
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(PORT);
if (inet_pton(AF_INET, "127.0.0.1", &server_addr.sin_addr) <= 0) {
perror("Invalid address");
exit(EXIT_FAILURE);
}

// 3. Bind the socket
if (bind(server_sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr)) == -1) {
perror("Bind failed");
close(server_sockfd);
exit(EXIT_FAILURE);
}

// 4. Listen for incoming connections
if (listen(server_sockfd, 5) == -1) {
perror("Listen failed");
close(server_sockfd);
exit(EXIT_FAILURE);
}

printf("Server listening on port %d...\n", PORT);

while (1) {
client_len = sizeof(client_addr);

// 5. Accept client connection
client_sockfd = accept(server_sockfd, (struct sockaddr*)&client_addr, &client_len);
if (client_sockfd == -1) {
perror("Accept failed");
```

```
continue;
}

printf("Client connected!\n");

while (1) {
memset(command, 0, MAX);
int bytes_read = read(client_sockfd, command, MAX - 1);
if (bytes_read <= 0) {
printf("Client disconnected.\n");
break;
}

command[bytes_read] = '\0'; // Null-terminate string
printf("\nServer received: %s", command);

printf("\nServer response: ");
scanf("%s", response);

write(client_sockfd, response, strlen(response));
}

close(client_sockfd);
}

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

//tcp_client.c

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

#define MAX 25
#define PORT 9734

int main() {
    int client_sockfd;
    char command[MAX], response[MAX];
    struct sockaddr_in server_addr;
    socklen_t client_len;
    int result;

    // 1. Create socket at client side
    client_sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (client_sockfd == -1) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    // 2. Initialize structure
    server_addr.sin_family = AF_INET;
```

```
    server_addr.sin_port = htons(PORT);

    // Convert IP address from text to binary
    if (inet_pton(AF_INET, "127.0.0.1", &server_addr.sin_addr) <= 0) {
        perror("Invalid address/ Address not supported");
        exit(EXIT_FAILURE);
    }

    client_len = sizeof(server_addr);

    // 3. Connect to server
    result = connect(client_sockfd, (struct sockaddr*)&server_addr, client_len);
    if (result == -1) {
        perror("Cannot connect");
        close(client_sockfd);
        return 1;
    }

    printf("Connected to server...\n");

    while (1) {
        // Client writes command
        printf("Client write: ");
        scanf("%s", command);

        // Send data to server
        write(client_sockfd, command, strlen(command));

        // Read server response
        memset(response, 0, MAX);
        int bytes_read = read(client_sockfd, response, MAX - 1);
        if (bytes_read <= 0) {
            printf("Server disconnected.\n");
            break;
        }

        response[bytes_read] = '\0';  // Null-terminate response
        printf("Client read: %s\n", response);
    }

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

//udp_server.c

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

#define MAX 80
#define PORT 9734

int main() {
```

```
    int server_sockfd;
    char buffer[MAX], response[MAX];
    struct sockaddr_in server_addr, client_addr;
    socklen_t client_len = sizeof(client_addr);

    // 1. Create UDP socket
    server_sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (server_sockfd == -1) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    // 2. Initialize server address structure
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = INADDR_ANY;

    // 3. Bind the socket
    if (bind(server_sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr)) == -1) {
        perror("Bind failed");
        close(server_sockfd);
        exit(EXIT_FAILURE);
    }

    printf("UDP Server listening on port %d...\n", PORT);

    while (1) {
        memset(buffer, 0, MAX);

        // 4. Receive data from client
        int bytes_received = recvfrom(server_sockfd, buffer, MAX - 1, 0,
                                      (struct sockaddr*)&client_addr, &client_len);
        if (bytes_received < 0) {
            perror("Receive failed");
            continue;
        }

        buffer[bytes_received] = '\0'; // Null-terminate string
        printf("\nServer received: %s", buffer);

        printf("\nServer response: ");
        scanf("%s", response);

        // 5. Send response back to client
        sendto(server_sockfd, response, strlen(response), 0,
               (struct sockaddr*)&client_addr, client_len);
    }

    close(server_sockfd);
    return 0;
}

//udp_client.c
#include <stdio.h>
#include <stdlib.h>
```

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

#define MAX 80
#define PORT 9734

int main() {
   int client_sockfd;
   char command[MAX], response[MAX];
   struct sockaddr_in server_addr;
   socklen_t server_len = sizeof(server_addr);

   // 1. Create UDP socket
   client_sockfd = socket(AF_INET, SOCK_DGRAM, 0);
   if (client_sockfd == -1) {
      perror("Socket creation failed");
      exit(EXIT_FAILURE);
   }

   // 2. Initialize server address structure
   memset(&server_addr, 0, sizeof(server_addr));
   server_addr.sin_family = AF_INET;
   server_addr.sin_port = htons(PORT);
   server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

   while (1) {
      // Client writes command
      printf("Client write: ");
      scanf("%s", command);

      // 3. Send data to server
      sendto(client_sockfd, command, strlen(command), 0,
            (struct sockaddr*)&server_addr, server_len);

      // 4. Receive server response
      memset(response, 0, MAX);
      int bytes_received = recvfrom(client_sockfd, response, MAX - 1, 0,
                                    (struct sockaddr*)&server_addr, &server_len);
      if (bytes_received < 0) {
         perror("Receive failed");
         break;
      }

      response[bytes_received] = '\0'; // Null-terminate response
      printf("Client read: %s\n", response);
   }

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