// UDP Client Template
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define MAX_BUFFER 1024
#define SERVER_PORT 8888

int main() {
int sockfd;
struct sockaddr_in server_addr;
char buffer[MAX_BUFFER];

```
// Create UDP socket
if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
    perror("Socket creation failed");
    exit(EXIT_FAILURE);
}

memset(&server_addr, 0, sizeof(server_addr));

// Configure server address
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(SERVER_PORT);
server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");  // Change to
server IP if needed

while (1) {
    printf("Enter message: ");
    fgets(buffer, MAX_BUFFER, stdin);
    buffer[strcspn(buffer, "\n")] = 0;  // Remove newline

    // Send message to server
    sendto(sockfd, buffer, strlen(buffer), 0, (struct
sockaddr*)&server_addr, sizeof(server_addr));

    // Receive response from server
    int n = recvfrom(sockfd, buffer, MAX_BUFFER, 0, NULL, NULL);
    buffer[n] = '\0';
    printf("Server: %s\n", buffer);
}

close(sockfd);
return 0;
```

}

// UDP Server Template
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define MAX_BUFFER 1024
#define SERVER_PORT 8888

int main() {
int sockfd;
struct sockaddr_in server_addr, client_addr;
char buffer[MAX_BUFFER];
socklen_t client_len = sizeof(client_addr);

```
// Create UDP socket
if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
    perror("Socket creation failed");
    exit(EXIT_FAILURE);
}

memset(&server_addr, 0, sizeof(server_addr));
memset(&client_addr, 0, sizeof(client_addr));

// Configure server address
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = INADDR_ANY;
server_addr.sin_port = htons(SERVER_PORT);

// Bind socket to server address
if (bind(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr)) <
0) {
    perror("Bind failed");
    exit(EXIT_FAILURE);
}

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

while (1) {
    // Receive message from client
    int n = recvfrom(sockfd, buffer, MAX_BUFFER, 0, (struct
sockaddr*)&client_addr, &client_len);
    buffer[n] = '\0';
    printf("Client: %s\n", buffer);

    // Process the message (echo back in this example)
```

```
    sendto(sockfd, buffer, strlen(buffer), 0, (struct
sockaddr*)&client_addr, client_len);
}

close(sockfd);
return 0;
```

}

// TCP Client Template
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define MAX_BUFFER 1024
#define SERVER_PORT 8888

int main() {
int sockfd;
struct sockaddr_in server_addr;
char buffer[MAX_BUFFER];

```
// Create TCP socket
if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    perror("Socket creation failed");
    exit(EXIT_FAILURE);
}

memset(&server_addr, 0, sizeof(server_addr));

// Configure server address
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(SERVER_PORT);
server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");  // Change to
server IP if needed

// Connect to server
if (connect(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr))
< 0) {
    perror("Connection failed");
    exit(EXIT_FAILURE);
}

while (1) {
    printf("Enter message: ");
    fgets(buffer, MAX_BUFFER, stdin);
```

```
    buffer[strcspn(buffer, "\n")] = 0;  // Remove newline

    // Send message to server
    send(sockfd, buffer, strlen(buffer), 0);

    // Receive response from server
    int n = recv(sockfd, buffer, MAX_BUFFER, 0);
    if (n <= 0) {
        printf("Server disconnected\n");
        break;
    }
    buffer[n] = '\0';
    printf("Server: %s\n", buffer);
}

close(sockfd);
return 0;
```

}

// TCP Server Template
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define MAX_BUFFER 1024
#define SERVER_PORT 8888

int main() {
int server_fd, client_fd;
struct sockaddr_in server_addr, client_addr;
char buffer[MAX_BUFFER];
socklen_t client_len = sizeof(client_addr);

```
// Create TCP socket
if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    perror("Socket creation failed");
    exit(EXIT_FAILURE);
}

memset(&server_addr, 0, sizeof(server_addr));
memset(&client_addr, 0, sizeof(client_addr));

// Configure server address
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = INADDR_ANY;
```

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

    // Bind socket to server address
    if (bind(server_fd, (struct sockaddr*)&server_addr, sizeof(server_addr))
    < 0) {
        perror("Bind failed");
        exit(EXIT_FAILURE);
    }

    // Listen for incoming connections
    if (listen(server_fd, 5) < 0) {
        perror("Listen failed");
        exit(EXIT_FAILURE);
    }

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

    while (1) {
        // Accept client connection
        if ((client_fd = accept(server_fd, (struct sockaddr*)&client_addr,
    &client_len)) < 0) {
            perror("Accept failed");
            continue;
        }

        printf("New client connected\n");

        while (1) {
            // Receive message from client
            int n = recv(client_fd, buffer, MAX_BUFFER, 0);
            if (n <= 0) {
                printf("Client disconnected\n");
                break;
            }
            buffer[n] = '\0';
            printf("Client: %s\n", buffer);

            // Process the message (echo back in this example)
            send(client_fd, buffer, strlen(buffer), 0);
        }

        close(client_fd);
    }

    close(server_fd);
    return 0;
```

}