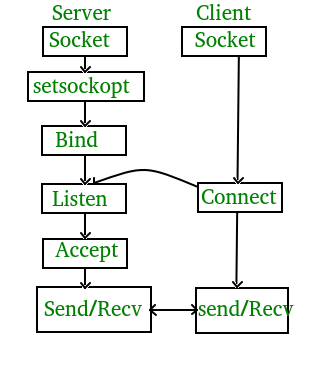
**ECHO server and client in C using socket programming** — for both **TCP** and **UDP**.



**Key Terms Explained:**

| **Term** | **Layer** | **Meaning** |
| --- | --- | --- |
| **Data** | Application Layer | Raw message/content (e.g., "Hello") that user wants to send. |
| **Packet** | Network Layer | Formatted data including IP headers (used for routing). |
| **Frame** | Data Link Layer | Encapsulated packet with MAC address headers (used on LAN). |
| **Socket** | Interface (between app & transport layer) | Programming endpoint for communication (e.g., send() and recv() use sockets). |

**Real-world Analogy:**

Imagine you’re **sending a letter**:

* **Data** = The message you wrote in the letter.
* **Packet** = The envelope with the message + address of the city.
* **Frame** = A parcel bag used inside a transport truck (MAC address).
* **Socket** = The post office counter window (a communication endpoint).

**Part A: TCP ECHO Server and Client (C)**

**tcp\_echo\_server.c**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <netinet/in.h>

#define PORT 10000+prn (for example if prn is 55 then 10000+55=10055 port)

#define BUF\_SIZE 1024

int main() {

int server\_fd, new\_socket;

struct sockaddr\_in address;

char buffer[BUF\_SIZE];

int addrlen = sizeof(address);

// Create TCP socket

server\_fd = socket(AF\_INET, SOCK\_STREAM, 0);

// Prepare server address

address.sin\_family = AF\_INET;

address.sin\_addr.s\_addr = INADDR\_ANY;

address.sin\_port = htons(PORT);

// Bind

bind(server\_fd, (struct sockaddr \*)&address, sizeof(address));

// Listen

listen(server\_fd, 3);

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

// Accept

new\_socket = accept(server\_fd, (struct sockaddr \*)&address, (socklen\_t\*)&addrlen);

while (1) {

memset(buffer, 0, BUF\_SIZE);

int bytes = read(new\_socket, buffer, BUF\_SIZE);

if (bytes <= 0) break;

printf("Received: %s", buffer);

send(new\_socket, buffer, bytes, 0); // echo back

}

close(new\_socket);

close(server\_fd);

return 0;

}

**tcp\_echo\_client.c**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <netinet/in.h>

#define PORT 10000+prn (for example if prn is 55 then 10000+55=10055 port)

#define BUF\_SIZE 1024

int main() {

int sock;

struct sockaddr\_in server;

char buffer[BUF\_SIZE];

// Create socket

sock = socket(AF\_INET, SOCK\_STREAM, 0);

server.sin\_family = AF\_INET;

server.sin\_port = htons(PORT);

server.sin\_addr.s\_addr = INADDR\_ANY;

connect(sock, (struct sockaddr \*)&server, sizeof(server));

while (1) {

printf("Enter message: ");

fgets(buffer, BUF\_SIZE, stdin);

send(sock, buffer, strlen(buffer), 0);

memset(buffer, 0, BUF\_SIZE);

read(sock, buffer, BUF\_SIZE);

printf("Echo from server: %s", buffer);

}

close(sock);

return 0;

}

**Part B: UDP ECHO Server and Client (C)**

**udp\_echo\_server.c**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <netinet/in.h>

#define PORT 10000+prn

#define BUF\_SIZE 1024

int main() {

int sockfd;

struct sockaddr\_in server, client;

char buffer[BUF\_SIZE];

socklen\_t len = sizeof(client);

sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);

server.sin\_family = AF\_INET;

server.sin\_port = htons(PORT);

server.sin\_addr.s\_addr = INADDR\_ANY;

bind(sockfd, (struct sockaddr \*)&server, sizeof(server));

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

while (1) {

memset(buffer, 0, BUF\_SIZE);

recvfrom(sockfd, buffer, BUF\_SIZE, 0, (struct sockaddr \*)&client, &len);

printf("Received: %s", buffer);

sendto(sockfd, buffer, strlen(buffer), 0, (struct sockaddr \*)&client, len);

}

close(sockfd);

return 0;

}

**udp\_echo\_client.c**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <netinet/in.h>

#define PORT 9090

#define BUF\_SIZE 1024

int main() {

int sockfd;

struct sockaddr\_in server;

char buffer[BUF\_SIZE];

socklen\_t len = sizeof(server);

sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);

server.sin\_family = AF\_INET;

server.sin\_port = htons(PORT);

server.sin\_addr.s\_addr = INADDR\_ANY;

while (1) {

printf("Enter message: ");

fgets(buffer, BUF\_SIZE, stdin);

sendto(sockfd, buffer, strlen(buffer), 0, (struct sockaddr \*)&server, len);

memset(buffer, 0, BUF\_SIZE);

recvfrom(sockfd, buffer, BUF\_SIZE, 0, (struct sockaddr \*)&server, &len);

printf("Echo from server: %s", buffer);

}

close(sockfd);

return 0;

}

**How to Compile & Run:**

**Compile TCP:**

gcc tcp\_echo\_server.c -o tcp\_server

gcc tcp\_echo\_client.c -o tcp\_client

./tcp\_server # in one terminal

./tcp\_client # in another

**Compile UDP:**

gcc udp\_echo\_server.c -o udp\_server

gcc udp\_echo\_client.c -o udp\_client

./udp\_server # in one terminal

./udp\_client # in another