# 🖥️ server.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8080

int main() {

int server\_fd, new\_socket;

struct sockaddr\_in address;

int addrlen = sizeof(address);

char buffer[1024] = {0};

char message[1024];

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

if (server\_fd == 0) {

perror("Socket failed");

exit(EXIT\_FAILURE);

}

address.sin\_family = AF\_INET;

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

address.sin\_port = htons(PORT);

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

listen(server\_fd, 3);

printf("Server waiting for connection...\n");

fflush(stdout);

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

printf("Client connected!\n");

fflush(stdout);

while (1) {

memset(buffer, 0, sizeof(buffer));

read(new\_socket, buffer, 1024);

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

fflush(stdout);

if (strncmp(buffer, "exit", 4) == 0) break;

printf("Server: ");

fflush(stdout);

fgets(message, sizeof(message), stdin);

send(new\_socket, message, strlen(message), 0);

if (strncmp(message, "exit", 4) == 0) break;

}

close(new\_socket);

close(server\_fd);

return 0;

}

# 💻 client.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <netdb.h>

#define PORT 8080

int main() {

int sock = 0;

struct sockaddr\_in serv\_addr;

struct hostent \*server;

char buffer[1024] = {0};

char message[1024];

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

if (sock < 0) {

printf("Socket creation error\n");

return -1;

}

server = gethostbyname("server");

if (server == NULL) {

printf("No such host\n");

return -1;

}

memset(&serv\_addr, 0, sizeof(serv\_addr));

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

memcpy(&serv\_addr.sin\_addr.s\_addr, server->h\_addr, server->h\_length);

if (connect(sock, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0) {

printf("Connection Failed\n");

return -1;

}

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

fflush(stdout);

while (1) {

printf("Client: ");

fflush(stdout);

fgets(message, sizeof(message), stdin);

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

if (strncmp(message, "exit", 4) == 0) break;

memset(buffer, 0, sizeof(buffer));

read(sock, buffer, 1024);

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

fflush(stdout);

if (strncmp(buffer, "exit", 4) == 0) break;

}

close(sock);

return 0;

}

**🐳 Step 2: Docker Setup**

**Dockerfile (shared for both)**

FROM gcc:latest

WORKDIR /app

# Copy C source files

COPY server.c .

COPY client.c .

# Compile both programs

RUN gcc -o server server.c

RUN gcc -o client client.c

# Default command (can be overridden by docker-compose)

CMD ["./server"]

# 🛠 Step 3: docker-compose.yml

version: '3.8'

services:

server:

build: .

container\_name: chat\_server

command: ./server

networks:

- chatnet

stdin\_open: true # Keep STDIN open

tty: true # Allocate a pseudo-TTY

client:

build: .

container\_name: chat\_client

command: ./client

depends\_on:

- server

networks:

- chatnet

stdin\_open: true

tty: true

networks:

chatnet:

# ✅ How to Run It

1. Put server.c, client.c, Dockerfile, and docker-compose.yml in the same folder.
2. Run the following commands from that folder:

docker-compose build

1. Start the **server** (in one terminal):

docker-compose run server

1. Then start the **client** (in another terminal):

docker-compose run client

1. 🛑 To Stop All Containers

docker-compose down