**Server Code (multi\_client\_server.c)**

// multi\_client\_server.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <sys/select.h>

#define PORT 8080

#define MAX\_CLIENTS 10

#define BUFFER\_SIZE 1024

int main() {

int server\_fd, client\_fd, max\_fd, activity, i, valread, new\_socket;

int client\_sockets[MAX\_CLIENTS] = {0};

struct sockaddr\_in server\_addr, client\_addr;

socklen\_t addrlen;

char buffer[BUFFER\_SIZE];

fd\_set readfds;

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

if (server\_fd == 0) {

perror("Socket failed");

exit(EXIT\_FAILURE);

}

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

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

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

if (bind(server\_fd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0) {

perror("Bind failed");

close(server\_fd);

exit(EXIT\_FAILURE);

}

if (listen(server\_fd, 3) < 0) {

perror("Listen");

close(server\_fd);

exit(EXIT\_FAILURE);

}

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

addrlen = sizeof(client\_addr);

while (1) {

FD\_ZERO(&readfds);

FD\_SET(server\_fd, &readfds);

max\_fd = server\_fd;

for (i = 0; i < MAX\_CLIENTS; i++) {

client\_fd = client\_sockets[i];

if (client\_fd > 0)

FD\_SET(client\_fd, &readfds);

if (client\_fd > max\_fd)

max\_fd = client\_fd;

activity = select(max\_fd + 1, &readfds, NULL, NULL, NULL);

if ((activity < 0) && (errno != EINTR)) {

printf("Select error");

}

}

if (FD\_ISSET(server\_fd, &readfds)) {

if ((new\_socket = accept(server\_fd, (struct sockaddr \*)&client\_addr,

&addrlen)) < 0) {

perror("Accept");

exit(EXIT\_FAILURE);

}

printf("New connection: socket fd %d, IP %s, PORT %d\n",

new\_socket, inet\_ntoa(client\_addr.sin\_addr),

ntohs(client\_addr.sin\_port));

for (i = 0; i < MAX\_CLIENTS; i++) {

if (client\_sockets[i] == 0) {

client\_sockets[i] = new\_socket;

break;

}

}

}

for (i = 0; i < MAX\_CLIENTS; i++) {

client\_fd = client\_sockets[i];

if (FD\_ISSET(client\_fd, &readfds)) {

valread = read(client\_fd, buffer, BUFFER\_SIZE);

if (valread == 0) {

getpeername(client\_fd, (struct sockaddr \*)&client\_addr, &addrlen);

printf("Client disconnected: IP %s, PORT %d\n",

inet\_ntoa(client\_addr.sin\_addr),

ntohs(client\_addr.sin\_port));

close(client\_fd);

client\_sockets[i] = 0;

} else {

buffer[valread] = '\0';

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

send(client\_fd, buffer, strlen(buffer), 0);

}

}

}

}

return 0;

}

**Client Code (client.c)**

// client.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8080

#define BUFFER\_SIZE 1024

int main() {

int sock = 0;

struct sockaddr\_in serv\_addr;

char buffer[BUFFER\_SIZE];

char message[BUFFER\_SIZE];

if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {

printf("Socket creation error\n");

return -1;

}

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

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

if (inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr) <= 0) {

printf("Invalid address\n");

return -1;

}

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

printf("Connection failed\n");

return -1;

}

while (1) {

printf("Enter message: ");

fgets(message, BUFFER\_SIZE, stdin);

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

int valread = read(sock, buffer, BUFFER\_SIZE);

buffer[valread] = '\0';

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

}

return 0;}