**//TCP\_Echo//** int main() {

int server\_fd, new\_socket;

char buffer[1024];

struct sockaddr\_in server\_addr;

socklen\_t addrlen = sizeof(server\_addr);

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

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

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

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

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

listen(server\_fd, 5);

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

while (1) {

int n = read(new\_socket, buffer, sizeof(buffer));

if (n <= 0) break;

write(new\_socket, buffer, n); // Echo back

} close(new\_socket);

close(server\_fd);

return 0;}

**//TCP\_Echo//** int main() {

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

struct sockaddr\_in server;

char buffer[1024];

server.sin\_family = AF\_INET;

server.sin\_port = htons(PORT);

inet\_pton(AF\_INET, "127.0.0.1", &server.sin\_addr);

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

while (1) {

printf("Enter message: ");

fgets(buffer, sizeof(buffer), stdin);

write(sock, buffer, strlen(buffer));

int n = read(sock, buffer, sizeof(buffer));

buffer[n] = '\0';

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

} close(sock);

return 0;}

**//UDP\_Echo//**

int main() {

int sockfd;

char buffer[BUFFER\_SIZE];

struct sockaddr\_in server\_addr;

socklen\_t addr\_len = sizeof(server\_addr);

if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0) {

perror("Socket creation failed");

exit(EXIT\_FAILURE);}

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(SERVER\_IP);

while (1) {

printf("Enter message: ");

fgets(buffer, BUFFER\_SIZE, stdin);

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

sendto(sockfd, buffer, strlen(buffer), 0, (const struct sockaddr \*)&server\_addr, addr\_len);

int n = recvfrom(sockfd, buffer, BUFFER\_SIZE, 0, (struct sockaddr \*)&server\_addr, &addr\_len);

buffer[n] = '\0';

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

} close(sockfd);

return 0; }

**//UDP Echo//** int main() {

int sockfd;

char buffer[BUFFER\_SIZE];

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

socklen\_t addr\_len = sizeof(client\_addr);

if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0) {

perror("Socket creation failed");

exit(EXIT\_FAILURE);

}

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

server\_addr.sin\_family = AF\_INET; // IPv4

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

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

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

perror("Bind failed");

close(sockfd);

exit(EXIT\_FAILURE);

}

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

while (1) {

memset(buffer, 0, BUFFER\_SIZE);

int n = recvfrom(sockfd, (char \*)buffer, BUFFER\_SIZE, 0, (struct sockaddr \*)&client\_addr, &addr\_len);

buffer[n] = '\0';

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

sendto(sockfd, buffer, strlen(buffer), 0, (const struct sockaddr \*)&client\_addr, addr\_len);

} close(sockfd);

return 0;}

**//TCP\_CHAT//** int sockfd;

void \*recv\_thread(void \*arg) {

char buf[BUF];

while (1) {

int n = recv(sockfd, buf, BUF - 1, 0);

if (n <= 0) exit(0);

buf[n] = 0;

printf("%s", buf);

fflush(stdout); }}

int main(int c, char \*\*v) {

if (c != 3) return printf("Usage: %s <ip> <port>\n", v[0]), 1;

struct sockaddr\_in s = {.sin\_family = AF\_INET, .sin\_port = htons(atoi(v[2]))};

inet\_pton(AF\_INET, v[1], &s.sin\_addr);

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

if (connect(sockfd, (void \*)&s, sizeof(s))) return perror("connect"), 1;

char buf[BUF];

recv(sockfd, buf, BUF - 1, 0); // receive prompt

printf("%s", buf);

fgets(buf, BUF, stdin); // send username

send(sockfd, buf, strlen(buf), 0);

pthread\_t t;

pthread\_create(&t, 0, recv\_thread, 0);

while (fgets(buf, BUF, stdin))

send(sockfd, buf, strlen(buf), 0);

pthread\_cancel(t);

close(sockfd);

return 0; }

**//TCP\_CHAT//**  struct Client {

int fd;

int in\_use;

char username[MAX\_USERNAME\_LENGTH];

} clients[MAX\_CLIENTS];

int find\_client\_by\_username(const char \*username) {

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

if (clients[i].in\_use && strcmp(clients[i].username, username) == 0)

return clients[i].fd;

return -1;}

void broadcast(int sender\_fd, const char \*sender\_name, const char \*msg) {

char buffer[BUFFER\_SIZE];

snprintf(buffer, sizeof(buffer), "[%s]: %s\n", sender\_name, msg);

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

if (clients[i].in\_use && clients[i].fd != sender\_fd)

send(clients[i].fd, buffer, strlen(buffer), 0);

printf("Broadcast from %s: %s", sender\_name, msg);}

void \*handle\_client(void \*arg) {

int fd = \*(int \*)arg;

free(arg);

char name[MAX\_USERNAME\_LENGTH] = {0}, buf[BUFFER\_SIZE];

send(fd, "Enter username: ", 16, 0);

int len = recv(fd, name, sizeof(name) - 1, 0);

if (len <= 0) return close(fd), NULL;

if (name[len - 1] == '\n') name[len - 1] = 0;

int index = -1;

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

if (!clients[i].in\_use) {

clients[i] = (struct Client){fd, 1, ""};

strncpy(clients[i].username, name, MAX\_USERNAME\_LENGTH);

index = i;

break;

}

if (index == -1) return send(fd, "Server full.\n", 13, 0), close(fd), NULL;

while ((len = recv(fd, buf, sizeof(buf) - 1, 0)) > 0) {

buf[len] = 0;

if (strncmp(buf, "/chat:", 6) == 0) {

char \*target = strtok(buf + 6, ":");

char \*msg = strtok(NULL, "");

if (target && msg)

send\_private(fd, name, target, msg);

} else {

broadcast(fd, name, buf);

} }

clients[index].in\_use = 0;

close(fd);

return NULL;}

int main(int argc, char \*argv[]) {

if (argc != 2) return printf("Usage: %s <port>\n", argv[0]), 1;

int port = atoi(argv[1]), server\_fd = socket(AF\_INET, SOCK\_STREAM, 0);

struct sockaddr\_in serv = {.sin\_family = AF\_INET, .sin\_addr.s\_addr = INADDR\_ANY, .sin\_port = htons(port)};

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

listen(server\_fd, 5);

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

while (1) {

struct sockaddr\_in cli;

socklen\_t len = sizeof(cli);

int \*client\_fd = malloc(sizeof(int));

\*client\_fd = accept(server\_fd, (struct sockaddr \*)&cli, &len);

if (\*client\_fd < 0) { free(client\_fd); continue; }

pthread\_t tid;

pthread\_create(&tid, NULL, handle\_client, client\_fd);

pthread\_detach(tid); }

close(server\_fd);}

**//For\_private\_chat\_only//**

void send\_private(int sender\_fd, const char \*sender\_name, const char \*target, const char \*msg) {

int target\_fd = find\_client\_by\_username(target);

char buffer[BUFFER\_SIZE];

if (target\_fd != -1) {

snprintf(buffer, sizeof(buffer), "[%s -> %s] %s\n", sender\_name, target, msg);

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

printf("Private message from %s to %s: %s", sender\_name, target, msg);

} else {

snprintf(buffer, sizeof(buffer), "User '%s' not found.\n", target);

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

printf("Private message failed: User '%s' not found.\n", target);

}

}

//daytime

// #include <time.h>

time\_t timer;

struct tm\* tm\_info;

char daytime\_str[125];

char \*daytime() {    // Get current time and format it

    time\_t timer = time(NULL);

    struct tm \*tm\_info = localtime(&timer);

    strftime(daytime\_str, sizeof(daytime\_str), "%Y-%m-%d %H:%M:%S %Z", tm\_info);

    return daytime\_str;

}

//ftp\_client// int main() {

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

    struct sockaddr\_in a = {AF\_INET, htons(9000)};

    inet\_pton(AF\_INET, "127.0.0.1", &a.sin\_addr);

    connect(s, (struct sockaddr\*)&a, sizeof(a));

    char b[1024], f[256];

    fgets(f, sizeof(f), stdin);

    f[strcspn(f, "\n")] = 0;

    send(s, f, strlen(f), 0);

    FILE \*fp = fopen("recv", "wb");

    int n;

    while ((n = recv(s, b, sizeof(b), 0)) > 0) fwrite(b, 1, n, fp);

    fclose(fp); close(s);

}

//ftp\_server// int main() {

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

    struct sockaddr\_in addr = {AF\_INET, htons(9000), INADDR\_ANY};

    bind(s, (struct sockaddr\*)&addr, sizeof(addr));

    listen(s, 1);

    int c = accept(s, NULL, NULL);

    char buf[1024];

    int r = recv(c, buf, sizeof(buf)-1, 0);

    buf[r] = 0;

    FILE \*f = fopen(buf, "rb");

    if (!f) send(c, "File not found", 14, 0);

    else while ((r = fread(buf, 1, sizeof(buf), f)) > 0) send(c, buf, r, 0);

    fclose(f); close(c); close(s);}