**Server.c:**

#include <stdio.h>

#include <unistd.h>

#include <sys/socket.h>

#include <string.h>

#include <netinet/in.h>

#include <errno.h>

#define MSG\_SIZE 512

#define MAX\_CLIENT 5

#define MYPORT 8000

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

//variables declaration

int server\_socket, client\_socket;

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

int server\_addr\_len, client\_addr\_len;

char server\_message[MSG\_SIZE], client\_message[MSG\_SIZE];

int error\_code;

//creating socket

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

if(server\_socket < 0){

perror("Can't create socket\n");

return 1;

}

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

server\_address.sin\_port = htons(MYPORT);

server\_address.sin\_addr.s\_addr = inet\_addr("127.0.0.1");//INADDR\_ANY;

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

//binding socket to specific ip and port

server\_addr\_len = sizeof(server\_address);

if(bind(server\_socket, (struct sockaddr \*)&server\_address, server\_addr\_len) < 0){

perror("Bind failed\n");

return 1;

}

// Listening

if(listen(server\_socket, MAX\_CLIENT) < 0){

printf("Can't listen\n");

return 1;

}

printf("Waiting for client connection.\n");

//client\_addr\_len = sizeof(client\_address);

// Accept connection

//client\_socket = accept(server\_socket, (struct sockaddr \*)&client\_address, &client\_addr\_len);

client\_socket = accept(server\_socket, NULL, NULL);

if(client\_socket < 0){

perror ("Client connection failed\n");

return 1;

}

printf("Client Connected.\n");

printf("Write \"bye\" to close connection.\n");

do{

//sending message to client

printf("Server : ");

gets(server\_message);

if(send(client\_socket, server\_message, MSG\_SIZE, 0) < 0){

perror("Message send failed\n");

return 1;

}

if(strcmp(server\_message,"bye")==0){

break;

}

//receiving message from client

error\_code = recv(client\_socket, client\_message, MSG\_SIZE, 0);

if (error\_code == 0){

printf("Disconnected\n");

}

else if(error\_code < 0){

perror("message received failed\n");

printf("%d\n", errno );

return 1;

}

printf("Client : ");

puts(client\_message);

} while(strcmp(client\_message,"bye")!=0);

//closing server socket

close(server\_socket);

return 0;

}

**Client.c:**

#include <stdio.h>

#include <unistd.h>

#include <netinet/in.h>

#include <sys/socket.h>

#include <string.h>

#include <errno.h>

#define MSG\_SIZE 512

#define MYPORT 8000

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

//variable declaration

int client\_socket;

int server\_addr\_len, error\_code;

struct sockaddr\_in server\_address;

struct sockaddr \*sockaddr\_ptr;

char server\_message[MSG\_SIZE], client\_message[MSG\_SIZE];

// Creating a socket.

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

if(client\_socket < 0){

perror("Can't create client socket.\n");

return 1;

}

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

server\_address.sin\_port = htons(MYPORT);

server\_address.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

//inet\_aton(AF\_INET, "127.0.0.1", &(server\_address.sin\_addr));//INADDR\_ANY;

//connecting to server

server\_addr\_len = sizeof(server\_address);

sockaddr\_ptr = (struct sockaddr \*)&server\_address;

if(connect(client\_socket, sockaddr\_ptr, server\_addr\_len)<0){

printf("Can't connect to server.\n");

return 1;

}

printf("Server Connected.\n");

printf("Write \"bye\" to close connection.\n");

do{

//receiving message from server

error\_code = recv(client\_socket, server\_message, MSG\_SIZE, 0);

if (error\_code == 0){

printf("Server disconnected.\n");

}

else if(error\_code < 0){

perror("message received failed\n");

printf("%d\n", errno );

return 1;

}

printf("Server : ");

puts(server\_message);

if(strcmp(server\_message,"bye")==0){

break;

}

//sending message to server

printf("Client : ");

gets(client\_message);

if(send(client\_socket, client\_message, MSG\_SIZE, 0) < 0){

perror("Can't send message to server.\n");

return 1;

}

} while(strcmp(client\_message,"bye")!=0);

//closing client socket

close(client\_socket);

return 0;

}