# TASK1:

**SERVER.c:**

**﻿**#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

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

if (argc != 2) {

printf("Usage: %s <port>\n", argv[0]);

exit(0);

}

char \*ip="127.0.0.1";

int port = atoi(argv[1]);

int sockfd;

struct sockaddr\_in server\_addr, client\_addr; char buffer[1024];

socklen\_t addr\_size;

int n;

printf("PORT Number: %d\n", port);

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

if (sockfd< 0) {

perror("[-] socket error");

exit(1);

}

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 (ip);

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

if (n< 0) {

perror("[-] bind error");

exit(1);

}

bzero(buffer, 1024);

addr\_size = sizeof(client\_addr);

recvfrom(sockfd, buffer, 1024, 0, (struct sockaddr \*)&client\_addr, &addr\_size);

printf("[+] data received: %s\n", buffer);

bzero(buffer, 1024);

strcpy(buffer, "HI I am a server");

sendto(sockfd, buffer, 1024, 0, (struct sockaddr\*)&client\_addr, sizeof(client\_addr));

printf("[+] data sent: %s\n", buffer);

return 0;

}

**CLIENT.c:**

**﻿**#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

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

if (argc != 2) {

printf("Usage: %s <port>\n", argv[0]);

exit(0);

}

char \*ip="127.0.0.1";

int port = atoi(argv[1]);

int sockfd;

struct sockaddr\_in addr;

char buffer[1024];

socklen\_t addr\_size;

int n;

printf("PORT Number: %d\n", port);

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

if (sockfd< 0) {

perror("[-] socket error");

exit(1);

}

memset(&addr, '\0', sizeof(addr));

addr.sin\_family = AF\_INET;

addr.sin\_port = htons (port);

addr.sin\_addr.s\_addr = inet\_addr(ip);

bzero(buffer, 1024);

strcpy(buffer, "hello");

sendto(sockfd, buffer, 1024, 0, (struct sockaddr \*)&addr, sizeof(addr));

printf("[+] data sent: %s\n", buffer);

bzero(buffer, 1024);

addr\_size = sizeof(addr);

recvfrom (sockfd, buffer, 1024, 0, (struct sockaddr\*)&addr, &addr\_size);

printf("[+] data received: %s\n", buffer);

return 0;

}

# OUTPUT:

****

TASK2:

**SERVER.c:**

#include <iostream>

#include <cstring>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <cstdlib>

#include <thread>

using namespace std;

#define PORT 4444

void handleClient(int newsocket, struct sockaddr\_in newaddr) {

char buffer[1024];

cout << "Connection accepted from " << inet\_ntoa(newaddr.sin\_addr) << ":"<< ntohs(newaddr.sin\_port) << endl;

while (true) {

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

int bytesReceived = recv(newsocket, buffer, sizeof(buffer), 0);

if (bytesReceived <= 0) {

cout << "Client " << inet\_ntoa(newaddr.sin\_addr) << ":"<< ntohs(newaddr.sin\_port) << " disconnected." << endl;

close(newsocket);

return;

}

cout << "Client " << inet\_ntoa(newaddr.sin\_addr) << ":"<< ntohs(newaddr.sin\_port) << " says: " << buffer << endl;

// Check if the client wants to exit

if (strcmp(buffer, "exit") == 0) {

cout << "Client " << inet\_ntoa(newaddr.sin\_addr) << ":"<< ntohs(newaddr.sin\_port) << " requested to exit." << endl;

close(newsocket);

return;

}

// Reply to the client

string reply;

cout << "Server: ";

getline(cin, reply); // Get input from the server user

send(newsocket, reply.c\_str(), reply.size(), 0);

}

}

int main() {

int sockfd, ret;

struct sockaddr\_in serveraddr;

int newsocket;

struct sockaddr\_in newaddr;

socklen\_t addr\_size;

pid\_t childpid;

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

if (sockfd < 0) {

cerr << "Socket error" << endl;

exit(1);

}

cout << "Socket created for server." << endl;

memset(&serveraddr, '\0', sizeof(serveraddr));

serveraddr.sin\_family = AF\_INET;

serveraddr.sin\_port = htons(PORT);

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

ret = bind(sockfd, (struct sockaddr\*)&serveraddr, sizeof(serveraddr));

if (ret < 0) {

cerr << "Error in bind." << endl;

exit(1);

}

if (listen(sockfd, 10) == 0) {

cout << "Listening.." << endl;

}

while (true) {

newsocket = accept(sockfd, (struct sockaddr\*)&newaddr, &addr\_size);

if (newsocket < 0) {

exit(1);

}

thread clientThread(handleClient, newsocket, newaddr);

clientThread.detach();

}

return 0;

}

**CLIENT.c:**

#include <iostream>

#include <cstring>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

using namespace std;

#define PORT 4444

int main() {

int clientsocket, ret;

struct sockaddr\_in serveraddr;

char buffer[1024];

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

if (clientsocket < 0) {

cout << "Socket error" << endl;

exit(1);

}

cout << "Socket created for client." << endl;

memset(&serveraddr, '\0', sizeof(serveraddr));

serveraddr.sin\_family = AF\_INET;

serveraddr.sin\_port = htons(PORT);

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

ret = connect(clientsocket, (struct sockaddr\*)&serveraddr, sizeof(serveraddr));

if (ret < 0) {

cout << "Error in connection." << endl;

exit(1);

}

cout << "Successful connection." << endl;

while (true) {

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

// Send a message to the server

string message;

cout << "Client: ";

getline(cin, message);

send(clientsocket, message.c\_str(), message.size(), 0);

// Check if the client wants to exit

if (message == "exit") {

cout << "You requested to exit. Closing the connection." << endl;

close(clientsocket);

return 0;

}

// Receive a message from the server

int bytesReceived = recv(clientsocket, buffer, sizeof(buffer), 0);

if (bytesReceived <= 0) {

cout << "Server closed the connection." << endl;

close(clientsocket);

exit(1);

}

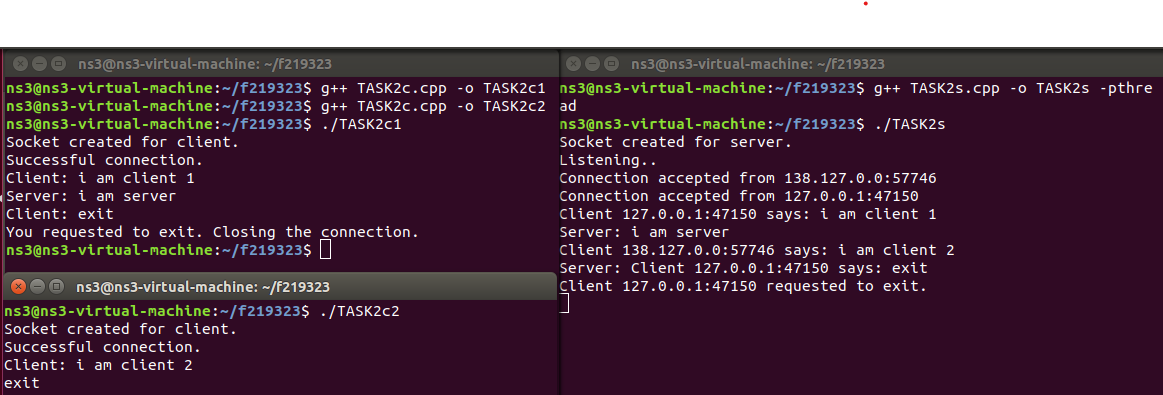
cout << "Server: " << buffer << endl;

}

close(clientsocket);

return 0;

}

OUTPUT: ****