CHAT APPLICATION – UDP

SERVER

#include<stdio.h>

#include<netinet/in.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netdb.h>

#include<string.h>

#include<stdlib.h>

#define MAX 80

#define PORT 43454

#define SA struct sockaddr

void func(int sockfd)

{

char buff[MAX];

int n,clen;

struct sockaddr\_in cli;

clen=sizeof(cli);

for(;;)

{

bzero(buff,MAX);

recvfrom(sockfd,buff,sizeof(buff),0,(SA \*)&cli,&clen);

printf("From client %s To client",buff);

bzero(buff,MAX);

n=0;

while((buff[n++]=getchar())!='\n');

sendto(sockfd,buff,sizeof(buff),0,(SA \*)&cli,clen);

if(strncmp("exit",buff,4)==0)

{

printf("Server Exit...\n");

break;

}

}

}

int main()

{

int sockfd;

struct sockaddr\_in servaddr;

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

if(sockfd==-1)

{

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

bzero(&servaddr,sizeof(servaddr));

servaddr.sin\_family=AF\_INET;

servaddr.sin\_addr.s\_addr=htonl(INADDR\_ANY);

servaddr.sin\_port=htons(PORT);

if((bind(sockfd,(SA \*)&servaddr,sizeof(servaddr)))!=0)

{

printf("socket bind failed...\n");

exit(0);

}

else

printf("Socket successfully binded..\n");

func(sockfd);

close(sockfd);

}

CLIENT-

#include<sys/socket.h>

#include<netdb.h>

#include<string.h>

#include<stdlib.h>

#include<stdio.h>

#define MAX 80

#define PORT 43454

#define SA struct sockaddr

int main()

{

char buff[MAX];

int sockfd,len,n;

struct sockaddr\_in servaddr;

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

if(sockfd==-1)

{

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

bzero(&servaddr,sizeof(len));

servaddr.sin\_family=AF\_INET;

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

servaddr.sin\_port=htons(PORT);

len=sizeof(servaddr);

for(;;)

{

printf("\nEnter string : ");

n=0;

while((buff[n++]=getchar())!='\n');

sendto(sockfd,buff,sizeof(buff),0,(SA \*)&servaddr,len);

bzero(buff,sizeof(buff));

recvfrom(sockfd,buff,sizeof(buff),0,(SA \*)&servaddr,&len);

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

if(strncmp("exit",buff,4)==0)

{

printf("Client Exit...\n");

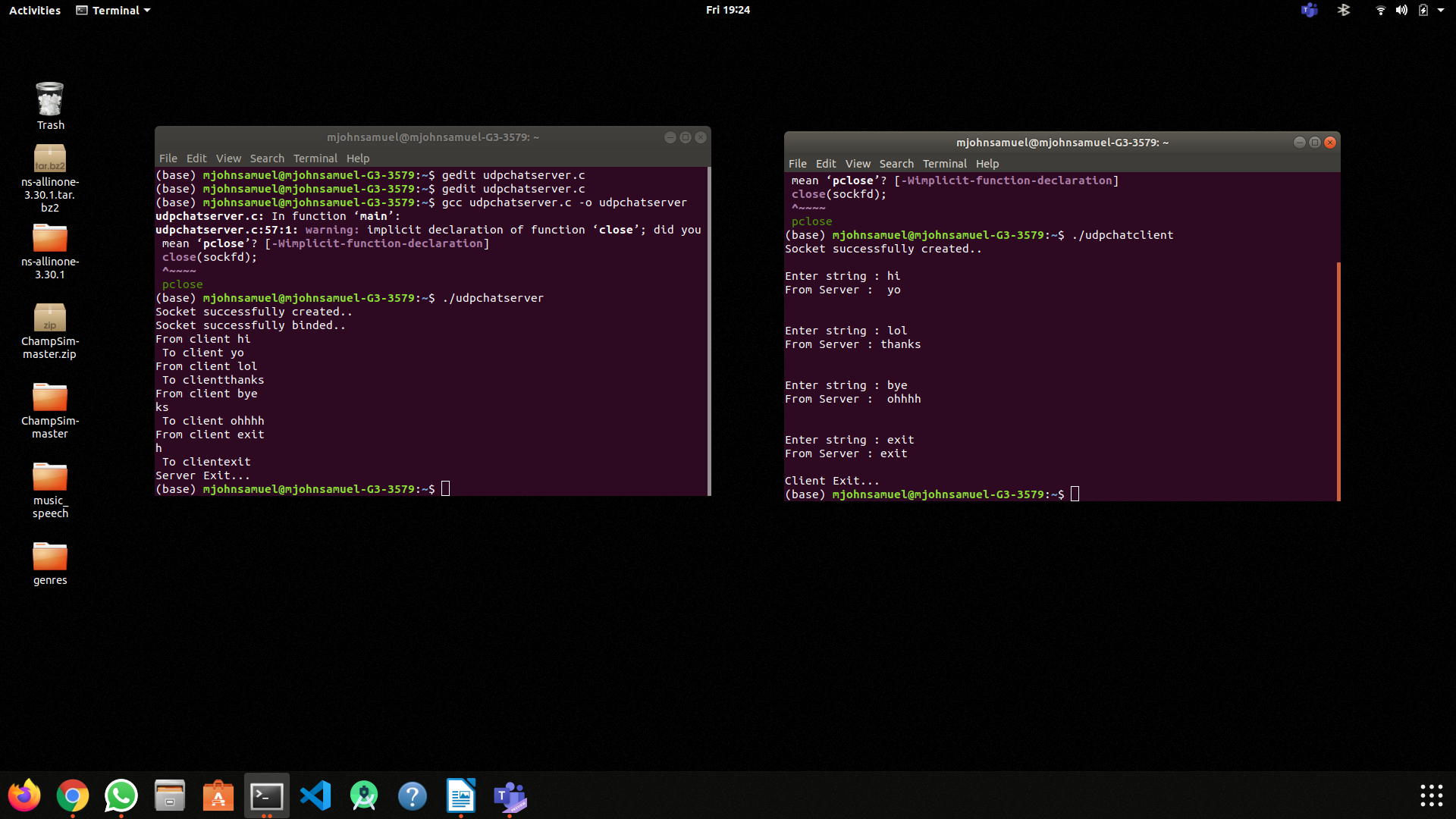
break;

}

}

close(sockfd);

}



CHAT APPLICATION – TCP

SERVER

#include <stdio.h>

#include <netdb.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

// Function designed for chat between client and server.

void func(int sockfd)

{

char buff[MAX];

int n;

// infinite loop for chat

for (;;) {

bzero(buff, MAX);

// read the message from client and copy it in buffer

read(sockfd, buff, sizeof(buff));

// print buffer which contains the client contents

printf("From client: %s\t To client : ", buff);

bzero(buff, MAX);

n = 0;

// copy server message in the buffer

while ((buff[n++] = getchar()) != '\n')

;

// and send that buffer to client

write(sockfd, buff, sizeof(buff));

// if msg contains "Exit" then server exit and chat ended.

if (strncmp("exit", buff, 4) == 0) {

printf("Server Exit...\n");

break;

}

}

}

// Driver function

int main()

{

int sockfd, connfd, len;

struct sockaddr\_in servaddr, cli;

// socket create and verification

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

if (sockfd == -1) {

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

bzero(&servaddr, sizeof(servaddr));

// assign IP, PORT

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

servaddr.sin\_port = htons(PORT);

// Binding newly created socket to given IP and verification

if ((bind(sockfd, (SA\*)&servaddr, sizeof(servaddr))) != 0) {

printf("socket bind failed...\n");

exit(0);

}

else

printf("Socket successfully binded..\n");

// Now server is ready to listen and verification

if ((listen(sockfd, 5)) != 0) {

printf("Listen failed...\n");

exit(0);

}

else

printf("Server listening..\n");

len = sizeof(cli);

// Accept the data packet from client and verification

connfd = accept(sockfd, (SA\*)&cli, &len);

if (connfd < 0) {

printf("server acccept failed...\n");

exit(0);

}

else

printf("server acccept the client...\n");

// Function for chatting between client and server

func(connfd);

// After chatting close the socket

close(sockfd);

}

CLIENT

#include <netdb.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

void func(int sockfd)

{

char buff[MAX];

int n;

for (;;) {

bzero(buff, sizeof(buff));

printf("Enter the string : ");

n = 0;

while ((buff[n++] = getchar()) != '\n')

;

write(sockfd, buff, sizeof(buff));

bzero(buff, sizeof(buff));

read(sockfd, buff, sizeof(buff));

printf("From Server : %s", buff);

if ((strncmp(buff, "exit", 4)) == 0) {

printf("Client Exit...\n");

break;

}

}

}

int main()

{

int sockfd, connfd;

struct sockaddr\_in servaddr, cli;

// socket create and varification

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

if (sockfd == -1) {

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

bzero(&servaddr, sizeof(servaddr));

// assign IP, PORT

servaddr.sin\_family = AF\_INET;

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

servaddr.sin\_port = htons(PORT);

// connect the client socket to server socket

if (connect(sockfd, (SA\*)&servaddr, sizeof(servaddr)) != 0) {

printf("connection with the server failed...\n");

exit(0);

}

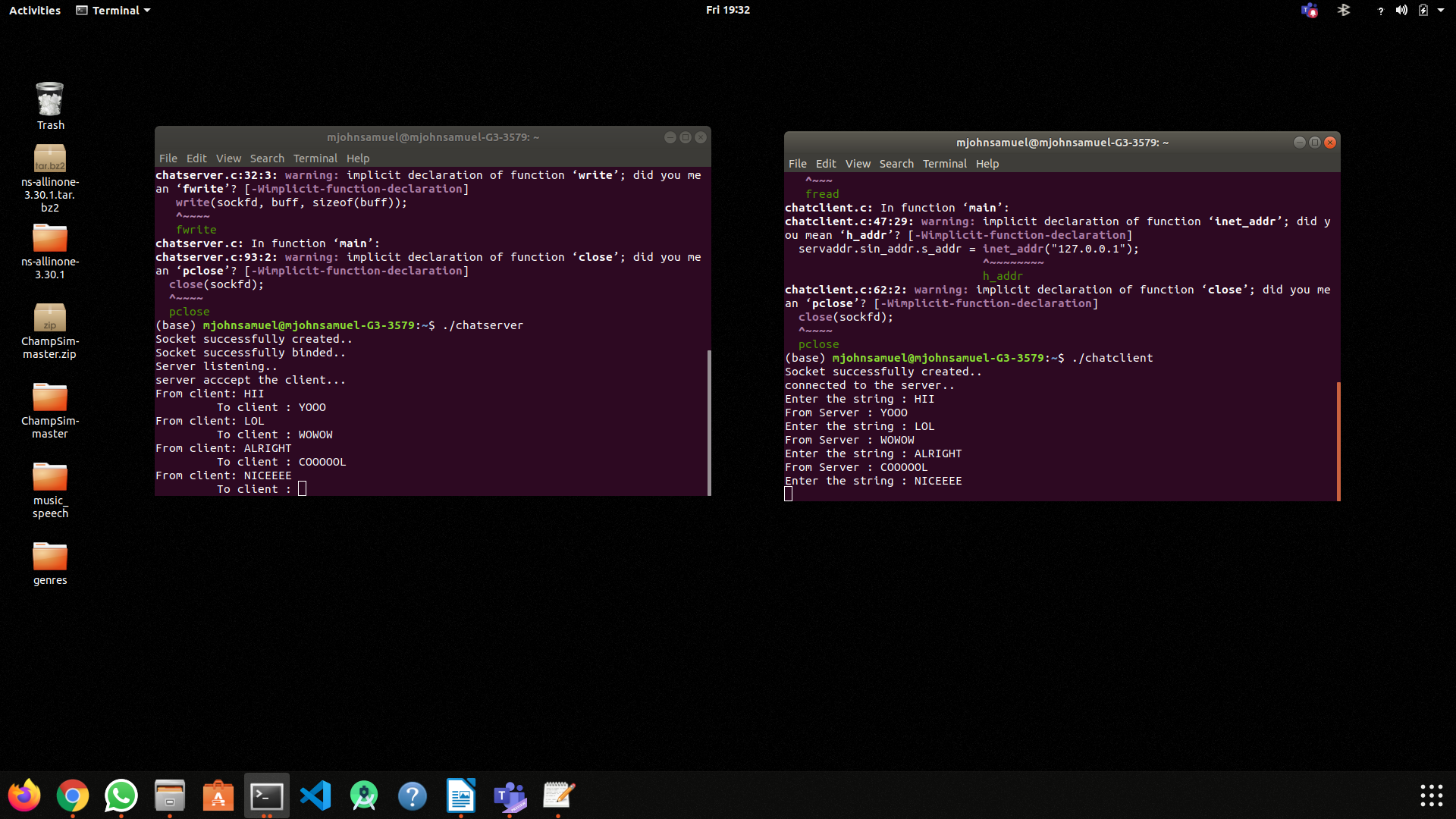
else

printf("connected to the server..\n");

// function for chat

func(sockfd);

// close the socket



close(sockfd);}

FILE TRANSFER TCP

SERVER -

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <arpa/inet.h>

#define SIZE 1024

void write\_file(int sockfd){

int n;

FILE \*fp;

char \*filename = "recv.txt";

char buffer[SIZE];

fp = fopen(filename, "w");

while (1) {

n = recv(sockfd, buffer, SIZE, 0);

if (n <= 0){

break;

return;

}

fprintf(fp, "%s", buffer);

bzero(buffer, SIZE);

}

return;

}

int main(){

char \*ip = "127.0.0.1";

int port = 8080;

int e;

int sockfd, new\_sock;

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

socklen\_t addr\_size;

char buffer[SIZE];

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

if(sockfd < 0) {

perror("[-]Error in socket");

exit(1);

}

printf("[+]Server socket created successfully.\n");

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

server\_addr.sin\_port = port;

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

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

if(e < 0) {

perror("[-]Error in bind");

exit(1);

}

printf("[+]Binding successfull.\n");

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

printf("[+]Listening....\n");

}else{

perror("[-]Error in listening");

exit(1);

}

addr\_size = sizeof(new\_addr);

new\_sock = accept(sockfd, (struct sockaddr\*)&new\_addr, &addr\_size);

write\_file(new\_sock);

printf("[+]Data written in the file successfully.\n");

return 0;

}

CLIENT-

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <arpa/inet.h>

#define SIZE 1024

void send\_file(FILE \*fp, int sockfd){

int n;

char data[SIZE] = {0};

while(fgets(data, SIZE, fp) != NULL) {

if (send(sockfd, data, sizeof(data), 0) == -1) {

perror("[-]Error in sending file.");

exit(1);

}

bzero(data, SIZE);

}

}

int main(){

char \*ip = "127.0.0.1";

int port = 8080;

int e;

int sockfd;

struct sockaddr\_in server\_addr;

FILE \*fp;

char \*filename = "samplefile.txt";

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

if(sockfd < 0) {

perror("[-]Error in socket");

exit(1);

}

printf("[+]Server socket created successfully.\n");

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

server\_addr.sin\_port = port;

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

e = connect(sockfd, (struct sockaddr\*)&server\_addr, sizeof(server\_addr));

if(e == -1) {

perror("[-]Error in socket");

exit(1);

}

printf("[+]Connected to Server.\n");

fp = fopen(filename, "r");

if (fp == NULL) {

perror("[-]Error in reading file.");

exit(1);

}

send\_file(fp, sockfd);

printf("[+]File data sent successfully.\n");

printf("[+]Closing the connection.\n");

close(sockfd);

return 0;

}

UDP SERVER -

// server code for UDP socket programming

#include <arpa/inet.h>

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <unistd.h>

#define IP\_PROTOCOL 0

#define PORT\_NO 15050

#define NET\_BUF\_SIZE 32

#define cipherKey 'S'

#define sendrecvflag 0

#define nofile "File Not Found!"

// function to clear buffer

void clearBuf(char\* b)

{

int i;

for (i = 0; i < NET\_BUF\_SIZE; i++)

b[i] = '\0';

}

// function to encrypt

char Cipher(char ch)

{

return ch ^ cipherKey;

}

// function sending file

int sendFile(FILE\* fp, char\* buf, int s)

{

int i, len;

if (fp == NULL) {

strcpy(buf, nofile);

len = strlen(nofile);

buf[len] = EOF;

for (i = 0; i <= len; i++)

buf[i] = Cipher(buf[i]);

return 1;

}

char ch, ch2;

for (i = 0; i < s; i++) {

ch = fgetc(fp);

ch2 = Cipher(ch);

buf[i] = ch2;

if (ch == EOF)

return 1;

}

return 0;

}

// driver code

int main()

{

int sockfd, nBytes;

struct sockaddr\_in addr\_con;

int addrlen = sizeof(addr\_con);

addr\_con.sin\_family = AF\_INET;

addr\_con.sin\_port = htons(PORT\_NO);

addr\_con.sin\_addr.s\_addr = INADDR\_ANY;

char net\_buf[NET\_BUF\_SIZE];

FILE\* fp;

// socket()

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

if (sockfd < 0)

printf("\nfile descriptor not received!!\n");

else

printf("\nfile descriptor %d received\n", sockfd);

// bind()

if (bind(sockfd, (struct sockaddr\*)&addr\_con, sizeof(addr\_con)) == 0)

printf("\nSuccessfully binded!\n");

else

printf("\nBinding Failed!\n");

while (1) {

printf("\nWaiting for file name...\n");

// receive file name

clearBuf(net\_buf);

nBytes = recvfrom(sockfd, net\_buf,

NET\_BUF\_SIZE, sendrecvflag,

(struct sockaddr\*)&addr\_con, &addrlen);

fp = fopen(net\_buf, "r");

printf("\nFile Name Received: %s\n", net\_buf);

if (fp == NULL)

printf("\nFile open failed!\n");

else

printf("\nFile Successfully opened!\n");

while (1) {

// process

if (sendFile(fp, net\_buf, NET\_BUF\_SIZE)) {

sendto(sockfd, net\_buf, NET\_BUF\_SIZE,

sendrecvflag,

(struct sockaddr\*)&addr\_con, addrlen);

break;

}

// send

sendto(sockfd, net\_buf, NET\_BUF\_SIZE,

sendrecvflag,

(struct sockaddr\*)&addr\_con, addrlen);

clearBuf(net\_buf);

}

if (fp != NULL)

fclose(fp);

}

return 0;

}

UDP CLIENT-

// client code for UDP socket programming

#include <arpa/inet.h>

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <unistd.h>

#define IP\_PROTOCOL 0

#define IP\_ADDRESS "127.0.0.1" // localhost

#define PORT\_NO 15050

#define NET\_BUF\_SIZE 32

#define cipherKey 'S'

#define sendrecvflag 0

// function to clear buffer

void clearBuf(char\* b)

{

int i;

for (i = 0; i < NET\_BUF\_SIZE; i++)

b[i] = '\0';

}

// function for decryption

char Cipher(char ch)

{

return ch ^ cipherKey;

}

// function to receive file

int recvFile(char\* buf, int s)

{

int i;

char ch;

for (i = 0; i < s; i++) {

ch = buf[i];

ch = Cipher(ch);

if (ch == EOF)

return 1;

else

printf("%c", ch);

}

return 0;

}

// driver code

int main()

{

int sockfd, nBytes;

struct sockaddr\_in addr\_con;

int addrlen = sizeof(addr\_con);

addr\_con.sin\_family = AF\_INET;

addr\_con.sin\_port = htons(PORT\_NO);

addr\_con.sin\_addr.s\_addr = inet\_addr(IP\_ADDRESS);

char net\_buf[NET\_BUF\_SIZE];

FILE\* fp;

// socket()

sockfd = socket(AF\_INET, SOCK\_DGRAM,

IP\_PROTOCOL);

if (sockfd < 0)

printf("\nfile descriptor not received!!\n");

else

printf("\nfile descriptor %d received\n", sockfd);

while (1) {

printf("\nPlease enter file name to receive:\n");

scanf("%s", net\_buf);

sendto(sockfd, net\_buf, NET\_BUF\_SIZE,

sendrecvflag, (struct sockaddr\*)&addr\_con,

addrlen);

printf("\n---------Data Received---------\n");

while (1) {

// receive

clearBuf(net\_buf);

nBytes = recvfrom(sockfd, net\_buf, NET\_BUF\_SIZE,

sendrecvflag, (struct sockaddr\*)&addr\_con,

&addrlen);

// process

if (recvFile(net\_buf, NET\_BUF\_SIZE)) {

break;

}

}

printf("\n-------------------------------\n");

}

return 0;

}

// Thos screenshot contains both udp and tcp file transfer .

