**1. TCP CHAT SERVER**

int main() {

int s\_sd, c\_sd;

struct sockaddr\_in serv, cli;

char recv\_msg[200], send\_msg[200];

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

if (s\_sd < 0) {

perror("Socket creation failed");

exit(1);

}

serv.sin\_family = AF\_INET;

serv.sin\_port = htons(8080);

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

socklen\_t serv\_len = sizeof(serv);

if (bind(s\_sd, (struct sockaddr \*)&serv, serv\_len) < 0) {

perror("Bind failed");

exit(1);

}

if (listen(s\_sd, 5) < 0) {

perror("Listen failed");

exit(1);

}

printf("Server is waiting for connections...\n");

socklen\_t cli\_len = sizeof(cli);

c\_sd = accept(s\_sd, (struct sockaddr \*)&cli, &cli\_len);

if (c\_sd < 0) {

perror("Accept failed");

exit(1);

}

while (1) {

read(c\_sd, recv\_msg, sizeof(recv\_msg));

printf("\nMessage from client: %s\n", recv\_msg);

printf("Enter message for client: ");

scanf(" %[^\n]", send\_msg);

write(c\_sd, send\_msg, sizeof(send\_msg));

}

close(c\_sd);

close(s\_sd);

return 0;

}

**2. TCP CHAT CLIENT**

int main() {

int sockfd, result;

struct sockaddr\_in address;

char ch1[200], ch2[200];

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

if (sockfd < 0) {

perror("Socket creation failed");

exit(1);

}

address.sin\_family = AF\_INET;

address.sin\_port = htons(8080);

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

socklen\_t len = sizeof(address);

result = connect(sockfd, (struct sockaddr \*)&address, len);

if (result == -1) {

perror("Connection failed");

exit(1);

}

printf("Connected to server.\n");

while (1) {

printf("\nEnter message to server: ");

scanf(" %[^\n]", ch1);

write(sockfd, ch1, sizeof(ch1));

read(sockfd, ch2, sizeof(ch2));

printf("Message from server: %s\n", ch2);

}

close(sockfd);

return 0;

**3. UPD CHAT SERVER**

int main()

{

int s\_sd, serv\_len;

char sour[200], dest[200];

struct sockaddr\_in serv;

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

serv\_len = sizeof(serv);

serv.sin\_family = AF\_INET;

serv.sin\_port = htons(8002);

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

bind(s\_sd, (struct sockaddr\*)&serv, serv\_len);

printf("\nThe SERVER is WAITING .........\n");

while(1)

{

recvfrom(s\_sd, sour, sizeof(sour), 0, (struct sockaddr\*)&serv, &serv\_len);

printf("\nThe message from client is : %s", sour);

printf("\nEnter the message for Client: ");

scanf(" %[^\n]", dest);

sendto(s\_sd, dest, strlen(dest), 0, (struct sockaddr\*)&serv, serv\_len);

}

close(s\_sd);

return 0;

}

**4.UDP CHAT CLIENT**

#include<stdio.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<arpa/inet.h>

#include<netinet/in.h>

#include<unistd.h>

#include<string.h>

int main()

{

int len, sd;

char ch1[200], ch2[200];

struct sockaddr\_in sock\_address;

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

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

sock\_address.sin\_port = htons(8002);

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

len = sizeof(sock\_address);

while(1)

{

printf("\nEnter the message to Server: ");

scanf(" %[^\n]", ch1);

sendto(sd, ch1, strlen(ch1), 0, (struct sockaddr\*)&sock\_address, len);

recvfrom(sd, ch2, sizeof(ch2), 0, (struct sockaddr\*)&sock\_address, &len);

printf("\nThe message from server: %s\n", ch2);

}

close(sd);

return 0;

}

**Send to Message Queue**#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

struct msgbuf {

long mtype;

char mtext[100];

};

int main() {

int msgid, len;

key\_t key;

struct msgbuf message;

key = 131;

msgid = msgget(key, IPC\_CREAT | 0666);

printf("\nq = %d\n", msgid);

printf("Enter the message : ");

scanf("%[^\n]s",message.mtext);

message.mtype = 1;

len = strlen(message.mtext);

if (msgsnd(msgid, &message, len, 0) == -1) {

perror("msgsnd failed");

exit(1);

}

return 0;

}

**Get form Message Queue**  
#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

struct msgbuf {

long mtype;

char mtext[100];

};

int main() {

int msgid, len;

key\_t key;

struct msgbuf message;

key = 131;

msgid = msgget(key, IPC\_CREAT | 0666);

printf("\nq = %d\n", msgid);

if (msgid == -1) {

perror("msgget failed");

exit(1);

}

len = msgrcv(msgid, &message, sizeof(message.mtext), 1, 0);

if (len == -1) {

perror("msgrcv failed");

exit(1);

}

message.mtext[len] = '\0';

printf("Message received: %s\n", message.mtext);

return 0;

}