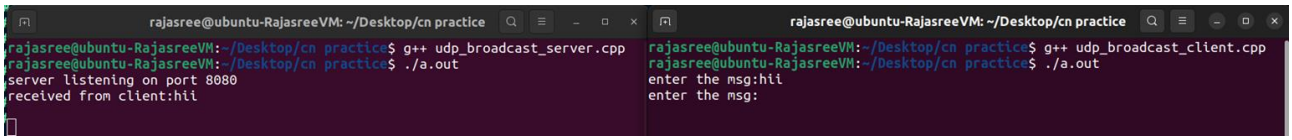
```
//udp broadcast server client
//udp server
#include<iostream>
#include<string>
#include<cstring>
#include<unistd.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#define PORT 8080
#define BUFFER_SIZE 1024

int main(){
int sockfd;
struct sockaddr_in servaddr,cliaddr;
char buffer[BUFFER_SIZE];
socklen_t len=sizeof(cliaddr);

//socket creation
if((sockfd=socket(AF_INET,SOCK_DGRAM,0))<0){
std::cerr<<"socket creation failed"<<std::endl;
exit(EXIT_FAILURE);
}

memset(&servaddr,0,sizeof(servaddr));

//filling server info
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=INADDR_ANY;
servaddr.sin_port=htons(PORT);

//binding
if(bind(sockfd,(const struct sockaddr *)&servaddr,sizeof(servaddr))<0){
std::cerr<<"binding failed"<<std::endl;
exit(EXIT_FAILURE);
}

std::cout<<"server listening on port "<<PORT<<std::endl;

while(1){
int n=recvfrom(sockfd,(char *)buffer,BUFFER_SIZE,MSG_WAITALL,(struct sockaddr
*)&cliaddr,&len);
buffer[n]='\0';
std::cout<<"received from client:"<<buffer<<std::endl;
}
return 0;
}
```

```
//udp broadcast server client
//udp client
#include<iostream>
#include<string>
#include<cstring>
#include<unistd.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#define PORT 8080
#define BUFFER_SIZE 1024

int main(){
int sockfd;
struct sockaddr_in servaddr;
char buffer[BUFFER_SIZE];


//socket creation
if((sockfd=socket(AF_INET,SOCK_DGRAM,0))<0){
std::cerr<<"socket creation failed"<<std::endl;
exit(EXIT_FAILURE);
}

memset(&servaddr,0,sizeof(servaddr));

//filling server info
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=INADDR_BROADCAST;
servaddr.sin_port=htons(PORT);

//setsocketopt
int be=1;
if(setsockopt(sockfd,SOL_SOCKET,SO_BROADCAST,&be,sizeof(be))<0){
std::cerr<<"setsocketopt failed"<<std::endl;
close(sockfd);
exit(EXIT_FAILURE);
}


while(1){
std::cout<<"enter the msg:";
fgets(buffer,BUFFER_SIZE,stdin);

sendto(sockfd,(const char *)buffer,sizeof(buffer),MSG_CONFIRM,(const struct sockaddr
*)&servaddr,sizeof(servaddr));
close(sockfd);
}
return 0;
```

```
}
```




```
//multicast sender receiver
//multicast receiver
#include<iostream>
#include<string>
#include<cstring>
#include<unistd.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#define PORT 8080
#define BUFFER_SIZE 1024

int main(){
int sockfd;
struct sockaddr_in servaddr,cliaddr;
char buffer[BUFFER_SIZE];
socklen_t len=sizeof(cliaddr);

//socket creation
if((sockfd=socket(AF_INET,SOCK_DGRAM,0))<0){
std::cerr<<"socket creation failed"<<std::endl;
exit(EXIT_FAILURE);
}

memset(&servaddr,0,sizeof(servaddr));

//filling server info
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=INADDR_ANY;
servaddr.sin_port=htons(PORT);

//binding
if(bind(sockfd,(const struct sockaddr *)&servaddr,sizeof(servaddr))<0){
std::cerr<<"binding failed"<<std::endl;
exit(EXIT_FAILURE);
}

std::cout<<"waiting for the msg on port "<<PORT<<std::endl;

while(1){
int n=recvfrom(sockfd,(char *)buffer,BUFFER_SIZE,MSG_WAITALL,(struct sockaddr
*)&cliaddr,&len);
buffer[n]='\0';
std::cout<<"received from sender:"<<buffer<<std::endl;
}
```

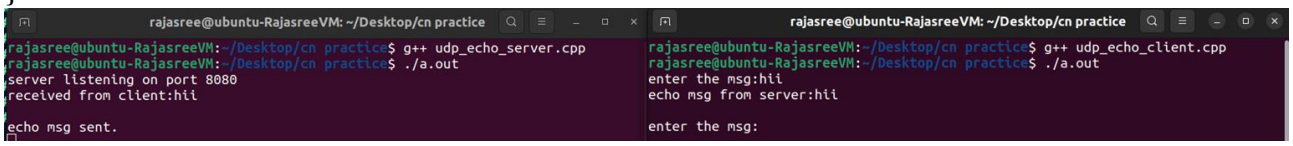
```
return 0;
}




//multicast sender receiver
//multicast sender
#include<iostream>
#include<string>
#include<cstring>
#include<unistd.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#define PORT 8080
#define BUFFER_SIZE 1024

int main(){
int sockfd;
struct sockaddr_in servaddr;
char buffer[BUFFER_SIZE];


//socket creation
if((sockfd=socket(AF_INET,SOCK_DGRAM,0))<0){
std::cerr<<"socket creation failed"<<std::endl;
exit(EXIT_FAILURE);
}

memset(&servaddr,0,sizeof(servaddr));

//filling server info
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=INADDR_BROADCAST;
servaddr.sin_port=htons(PORT);

//setsocketopt
int be=1;
if(setsockopt(sockfd,SOL_SOCKET,SO_BROADCAST,&be,sizeof(be))<0){
std::cerr<<"setsocketopt failed"<<std::endl;
close(sockfd);
exit(EXIT_FAILURE);
}


while(1){
std::cout<<"enter the msg:";
fgets(buffer,BUFFER_SIZE,stdin);
```

```
sendto(sockfd,(const char *)buffer,sizeof(buffer),MSG_CONFIRM,(const struct sockaddr
*)&servaddr,sizeof(servaddr));
close(sockfd);
}
return 0;
}
```


```
//udp echo server client
//udp echo server
#include<iostream>
#include<string>
#include<cstring>
#include<unistd.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#define PORT 8080
#define BUFFER_SIZE 1024

int main(){
int sockfd;
struct sockaddr_in servaddr,cliaddr;
char buffer[BUFFER_SIZE];
socklen_t len=sizeof(cliaddr);

//socket creation
if((sockfd=socket(AF_INET,SOCK_DGRAM,0))<0){
std::cerr<<"socket creation failed"<<std::endl;
exit(EXIT_FAILURE);
}

memset(&servaddr,0,sizeof(servaddr));
memset(&cliaddr,0,sizeof(cliaddr));

//filling server info
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=INADDR_ANY;
servaddr.sin_port=htons(PORT);

//binding
if(bind(sockfd,(const struct sockaddr *)&servaddr,sizeof(servaddr))<0){
std::cerr<<"binding failed"<<std::endl;
exit(EXIT_FAILURE);
}

std::cout<<"server listening on port "<<PORT<<std::endl;

while(1){
int n=recvfrom(sockfd,(char *)buffer,BUFFER_SIZE,MSG_WAITALL,(struct sockaddr
*)&cliaddr,&len);
```

```
buffer[n]='\0';
std::cout<<"received from client:"<<buffer<<std::endl;
sendto(sockfd,(const char *)buffer,sizeof(buffer),MSG_CONFIRM,(const struct sockaddr
*)&cliaddr,len);
std::cout<<"echo msg sent."<<std::endl;
}
return 0;
}

//udp echo server client
//udp echo client
#include<iostream>
#include<string>
#include<cstring>
#include<unistd.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#define PORT 8080
#define BUFFER_SIZE 1024

int main(){
int sockfd;
struct sockaddr_in servaddr;
char buffer[BUFFER_SIZE];
//socklen_t len=sizeof(servaddr);


//socket creation
if((sockfd=socket(AF_INET,SOCK_DGRAM,0))<0){
std::cerr<<"socket creation failed"<<std::endl;
exit(EXIT_FAILURE);
}

memset(&servaddr,0,sizeof(servaddr));

//filling server info
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=INADDR_ANY;
servaddr.sin_port=htons(PORT);


int n,len;
while(1){
std::cout<<"enter the msg:";
fgets(buffer,BUFFER_SIZE,stdin);

sendto(sockfd,(const char *)buffer,sizeof(buffer),MSG_CONFIRM,(const struct sockaddr
*)&servaddr,sizeof(servaddr));

n=recvfrom(sockfd,(char *)buffer,BUFFER_SIZE,MSG_WAITALL,(struct sockaddr
*)&servaddr,(socklen_t *)&len);
```

```
buffer[n]='\0';
std::cout<<"echo msg from server:"<<buffer<<std::endl;
close(sockfd);
}
return 0;
}
```




```
//tcp echo server client
//tcp echo server
#include<iostream>
#include<string>
#include<cstring>
#include<unistd.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#define PORT 8080
#define BUFFER_SIZE 1024

int main(){
int sockfd;
struct sockaddr_in servaddr,cliaddr;
char buffer[BUFFER_SIZE];
socklen_t len=sizeof(cliaddr);

//socket creation
if((sockfd=socket(AF_INET,SOCK_DGRAM,0))<0){
std::cerr<<"socket creation failed"<<std::endl;
exit(EXIT_FAILURE);
}

memset(&servaddr,0,sizeof(servaddr));
memset(&cliaddr,0,sizeof(cliaddr));

//filling server info
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=INADDR_ANY;
servaddr.sin_port=htons(PORT);

//binding
if(bind(sockfd,(const struct sockaddr *)&servaddr,sizeof(servaddr))<0){
std::cerr<<"binding failed"<<std::endl;
exit(EXIT_FAILURE);
}

std::cout<<"server listening on port "<<PORT<<std::endl;

while(1){
```

```
int n=recvfrom(sockfd,(char *)buffer,BUFFER_SIZE,MSG_WAITALL,(struct sockaddr
*)&cliaddr,&len);
buffer[n]='\0';
std::cout<<"received from client:"<<buffer<<std::endl;
sendto(sockfd,(const char *)buffer,sizeof(buffer),MSG_CONFIRM,(const struct sockaddr
*)&cliaddr,len);
std::cout<<"echo msg sent."<<std::endl;
}
return 0;
}

//tcp echo server client
//tcp echo client
#include<iostream>
#include<string>
#include<cstring>
#include<unistd.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#define PORT 8080
#define BUFFER_SIZE 1024

int main(){
int sockfd;
struct sockaddr_in servaddr;
char buffer[BUFFER_SIZE];
//socklen_t len=sizeof(servaddr);


//socket creation
if((sockfd=socket(AF_INET,SOCK_DGRAM,0))<0){
std::cerr<<"socket creation failed"<<std::endl;
exit(EXIT_FAILURE);
}

memset(&servaddr,0,sizeof(servaddr));

//filling server info
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=INADDR_ANY;
servaddr.sin_port=htons(PORT);


int n,len;
while(1){
std::cout<<"enter the msg:";
fgets(buffer,BUFFER_SIZE,stdin);

sendto(sockfd,(const char *)buffer,sizeof(buffer),MSG_CONFIRM,(const struct sockaddr
*)&servaddr,sizeof(servaddr));
```

```
n=recvfrom(sockfd,(char *)buffer,BUFFER_SIZE,MSG_WAITALL,(struct sockaddr
*)&servaddr,(socklen_t *)&len);
buffer[n]='\0';
std::cout<<"echo msg from server:"<<buffer<<std::endl;
close(sockfd);
}
return 0;
}
```