**Ex No 1 Create Socket and display Socket ID**

**Aim**

To create a Socket and display its ID

**Program**

#include<stdio.h>

#include<string.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

void main(){

int SocketFd;

//Create socket

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

if(SocketFd == -1)

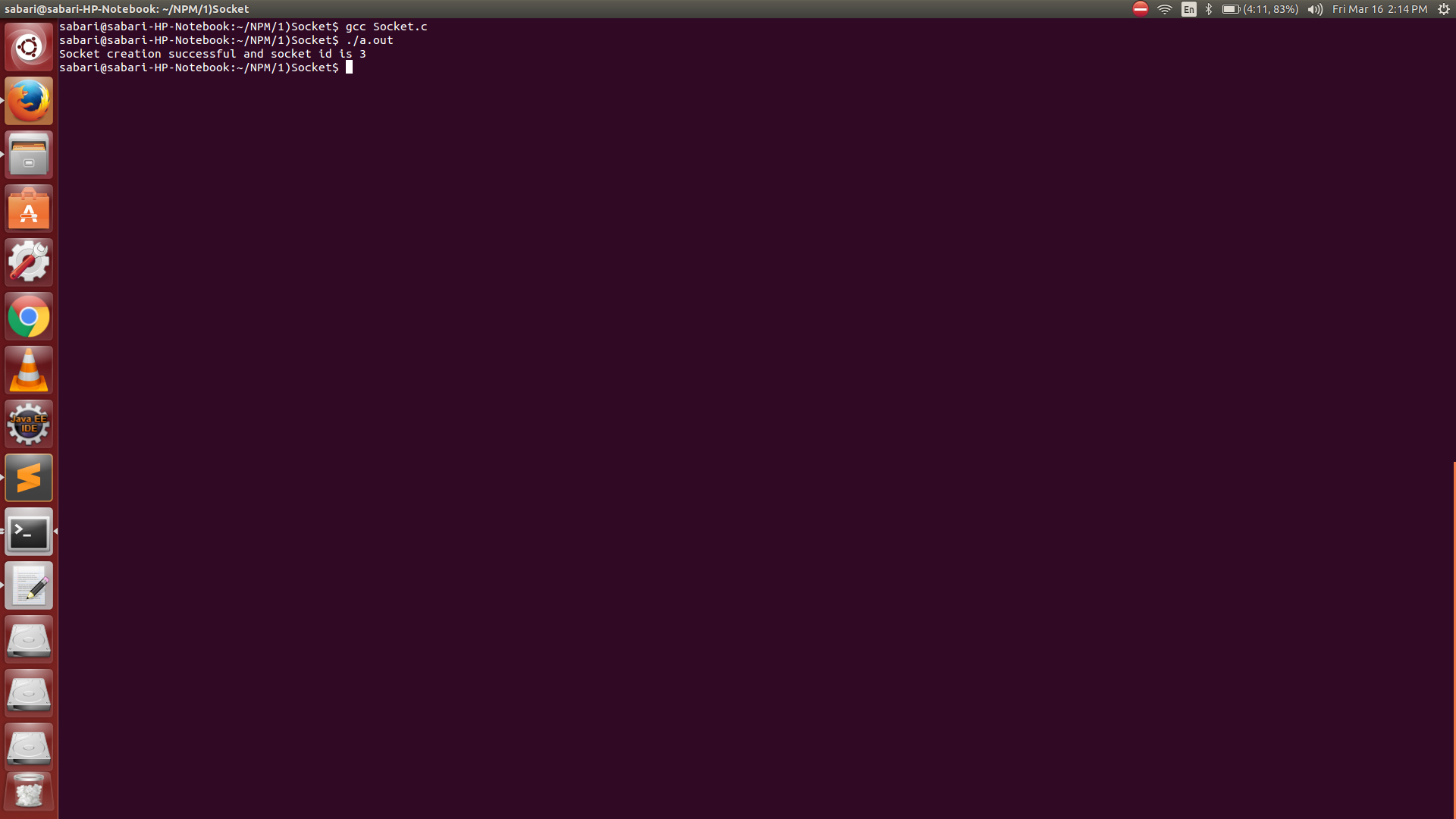
printf("Socket creation failed \n");

else

printf("Socket creation successful and socket id is %d\n",SocketFd);

}

**Output**



**Ex No 2 Implementation of Address Conversion Routines**

**Aim**

To implement Address Conversion routines

**Program**

#include<stdio.h>

#include<stdlib.h>

#include<netinet/in.h>

#include<sys/socket.h>

#include<arpa/inet.h>

int main(){

struct in\_addr ipAddress;

long int address;

char \*ptr;

char \*hostAddress = (char \*)malloc(sizeof(char));

printf("Enter the host Address \n");

fgets(hostAddress,100,stdin);

address = inet\_addr(hostAddress);

printf("%s in binary form is %ld \n",hostAddress,address);

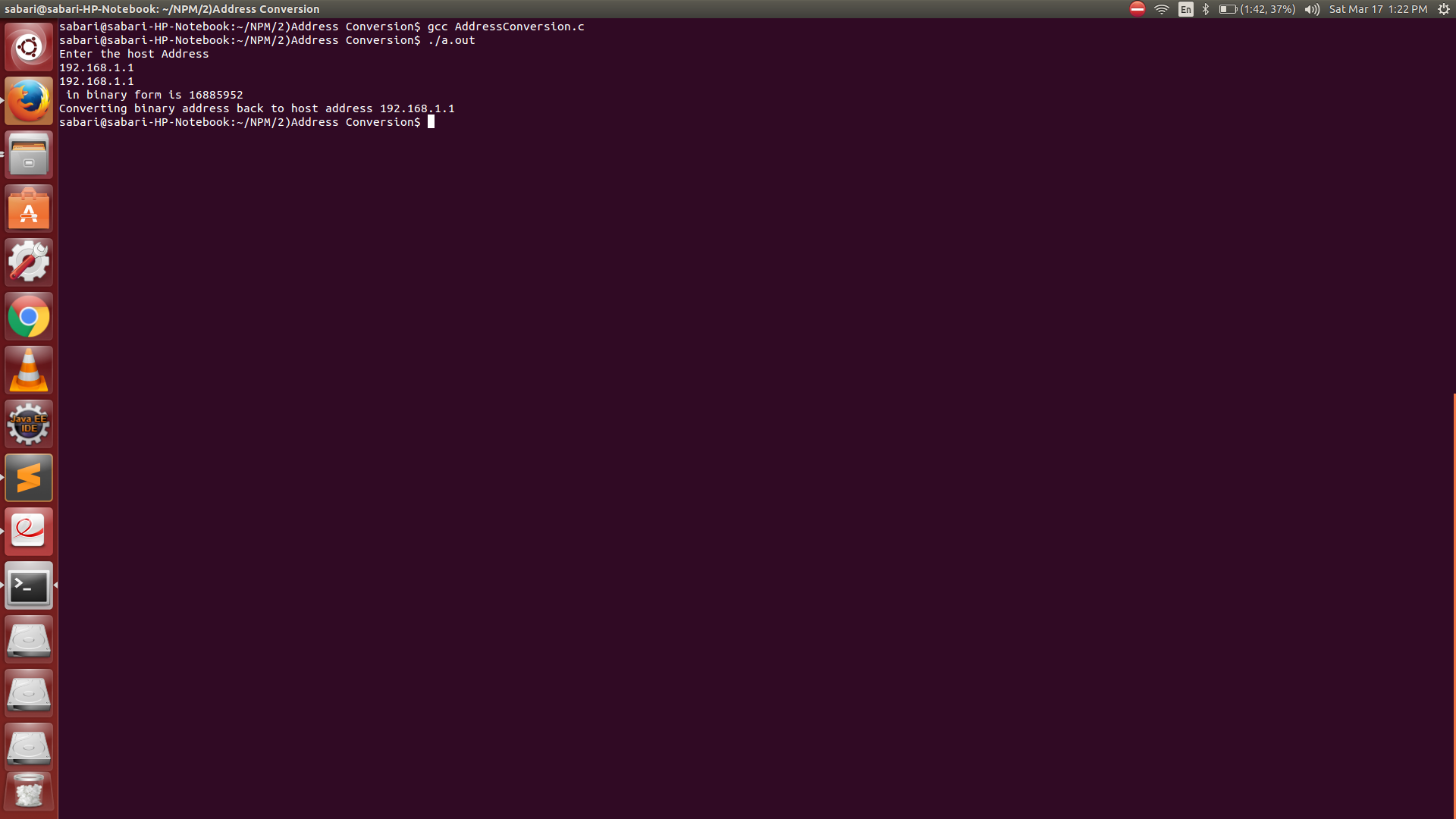
ipAddress.s\_addr = address;

ptr = inet\_ntoa(ipAddress);

printf("Converting binary address back to host address %s\n",ptr);

return 0; }

**Output**



**Ex No 3 Develop a Client Server Application for Chat using TCP**

**Aim**

To develop a client Server Application for chat using TCP

**Program**

**Server**

#include<stdio.h>

#include<string.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

void communicateClient(int connectionSocket){

char receive\_buffer[1024];

char send\_buffer[1024];

while(1){

bzero(receive\_buffer,1024);

read(connectionSocket,receive\_buffer,sizeof(receive\_buffer));

printf("Message from client \n");

printf("%s \n",receive\_buffer);

bzero(send\_buffer,1024);

printf("Enter the message for client \n");

fgets(send\_buffer,1024,stdin);

write(connectionSocket,send\_buffer,sizeof(send\_buffer));

if(strncmp(send\_buffer,"exit",4)==0)

break;

}

}

void main(){

int serverSocket,connectionSocket,clientAddressLength;

int bind\_result,listen\_result;

struct sockaddr\_in serverAddress,clientAddress;

//Create socket

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

if(serverSocket == -1)

printf("Socket creation failed \n");

else

printf("Socket creation successful \n");

//Configure server address

serverAddress.sin\_family = AF\_INET;

serverAddress.sin\_port = htons(8000);

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

//Binds the created socket with the given address

bind\_result = bind(serverSocket,(struct sockaddr \*)&serverAddress,sizeof(serverAddress));

if(bind\_result == -1)

printf("Bind process failed \n");

else

printf("Bind successful \n");

//Listens for client connection on the specified socket

listen\_result = listen(serverSocket,10); //10 is the backlog value that specifies maximum number of clients can wait in the connection queue

if(listen\_result == -1 )

printf("Server is not listening \n");

else

printf("Server is listening \n");

clientAddressLength = sizeof(clientAddress);

connectionSocket = accept(serverSocket,(struct sockaddr \*)&clientAddress,&clientAddressLength);

if(connectionSocket == -1 )

printf("New connection rejected \n");

else{

printf("New connection accepted \n");

communicateClient(connectionSocket);

}

}

**Client**

#include<stdio.h>

#include<string.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

void communicateServer(int clientSocket){

char send\_buffer[1024];

char receive\_buffer[1024];

while(1){

bzero(send\_buffer,1024);

printf("Enter the message for server \n");

fgets(send\_buffer,1024,stdin);

write(clientSocket,send\_buffer,sizeof(send\_buffer));

bzero(receive\_buffer,1024);

read(clientSocket,receive\_buffer,sizeof(receive\_buffer));

printf("Message from server \n");

printf("%s \n",receive\_buffer);

if(strncmp(receive\_buffer,"exit",4)==0)

break;

}

}

void main(){

int clientSocket,connectionResult;

struct sockaddr\_in serverAddress;

//Create a socket

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

if(clientSocket == -1 )

printf("Socket creation failed \n");

else

printf("Socket creation successful \n");

serverAddress.sin\_family = AF\_INET;

serverAddress.sin\_port = htons(8000);

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

//Connect with server

connectionResult = connect(clientSocket,(struct sockaddr \*)&serverAddress,sizeof(serverAddress));

if(connectionResult == -1 )

printf("Connection failed \n");

else{

printf("Connection scuccessful \n");

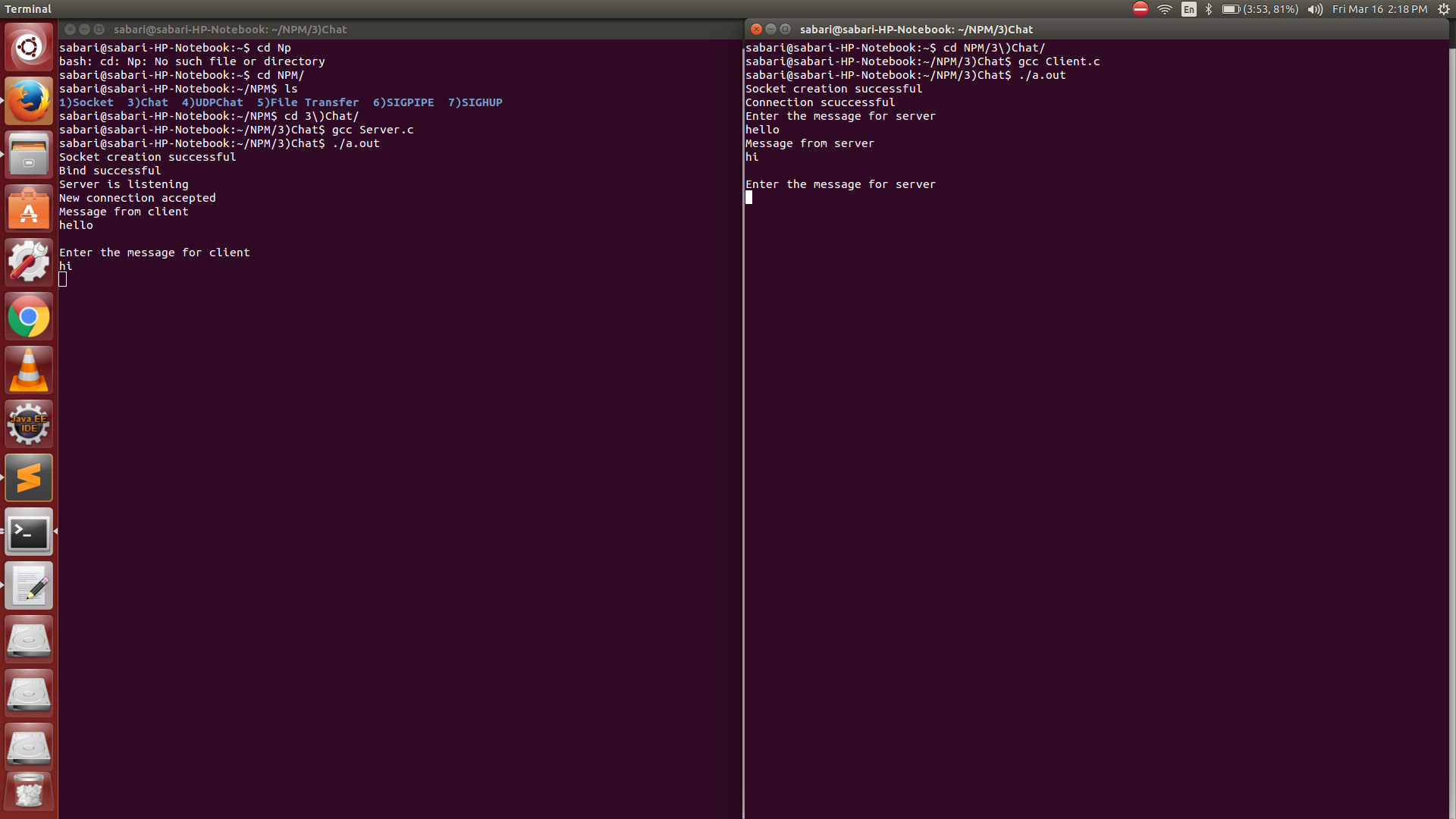
communicateServer(clientSocket);

close(clientSocket);

}

}

**Output**



**Ex No 4 Implementation of UDP Client Server Communication using Bind, SendTo, and RecvFrom System Calls**

**Aim**

To develop a client Server Application for chat using UDP

**Program**

#include<stdio.h>

#include<arpa/inet.h>

#include<netinet/in.h>

#include<sys/socket.h>

#include<string.h>

void communicateClient(int connectionSocket,struct sockaddr\_in clientAddress){

char receive\_buffer[1024];

char send\_buffer[1024];

int clientAddressLength = sizeof(clientAddress);

while(1){

bzero(receive\_buffer,1024);

recvfrom(connectionSocket,receive\_buffer,sizeof(receive\_buffer),0,(struct sockaddr \*)&clientAddress,&clientAddressLength);

printf("Message from client \n");

printf("%s \n",receive\_buffer);

bzero(send\_buffer,1024);

printf("Enter the message for client \n");

fgets(send\_buffer,1024,stdin);

sendto(connectionSocket,send\_buffer,sizeof(send\_buffer),0,(struct sockaddr \*)&clientAddress,sizeof(clientAddress));

if(strncmp(send\_buffer,"exit",4)==0)

break;

}

}

void main(){

int serverSocket;

int bindResult;

char buffer[100];

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

struct sockaddr\_in serverAddress,clientAddress;

if(serverSocket == -1)

printf("Socket is not created \n");

else{

printf("Socket created successfully \n");

serverAddress.sin\_family = AF\_INET;

serverAddress.sin\_port = htons(9000);

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

bindResult = bind(serverSocket,(struct sockaddr \*)&serverAddress,sizeof(serverAddress));

if(bindResult == -1)

printf("Bind not successful \n");

else{

printf("Bind successful \n");

communicateClient(serverSocket,clientAddress);

/\*recvfrom(serverSocket,buffer,sizeof(buffer),0,(struct sockaddr \*)&clientAddress,&clientAddressLength);

printf("%s \n ",buffer);

sendto(serverSocket,buffer,sizeof(buffer),0,(struct sockaddr \*)&clientAddress,sizeof(clientAddress));\*/

}

}

}

**Client**

#include<stdio.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<string.h>

#include<sys/socket.h>

void communicateServer(int clientSocket,struct sockaddr\_in serverAddress){

char send\_buffer[1024];

char receive\_buffer[1024];

while(1){

bzero(send\_buffer,1024);

printf("Enter the message for server \n");

fgets(send\_buffer,1024,stdin);

sendto(clientSocket,send\_buffer,sizeof(send\_buffer),0,(struct sockaddr \*)&serverAddress,sizeof(serverAddress));

bzero(receive\_buffer,1024);

recvfrom(clientSocket,receive\_buffer,sizeof(receive\_buffer),0,NULL,NULL);

printf("Message from server \n");

printf("%s \n",receive\_buffer);

if(strncmp(receive\_buffer,"exit",4)==0)

break;

}

}

void main(){

int clientSocket;

char buffer[100];

struct sockaddr\_in serverAddress;

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

if(clientSocket == -1)

printf("Socket creation failed \n");

else{

printf("Socket creation successful \n");

serverAddress.sin\_family = AF\_INET;

serverAddress.sin\_port = htons(9000);

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

communicateServer(clientSocket,serverAddress);

/\*int serverAddressLength = sizeof(serverAddress);

sendto(clientSocket,"Hello Server",13,0,(struct sockaddr \*)&serverAddress,serverAddressLength);

printf("test \n");

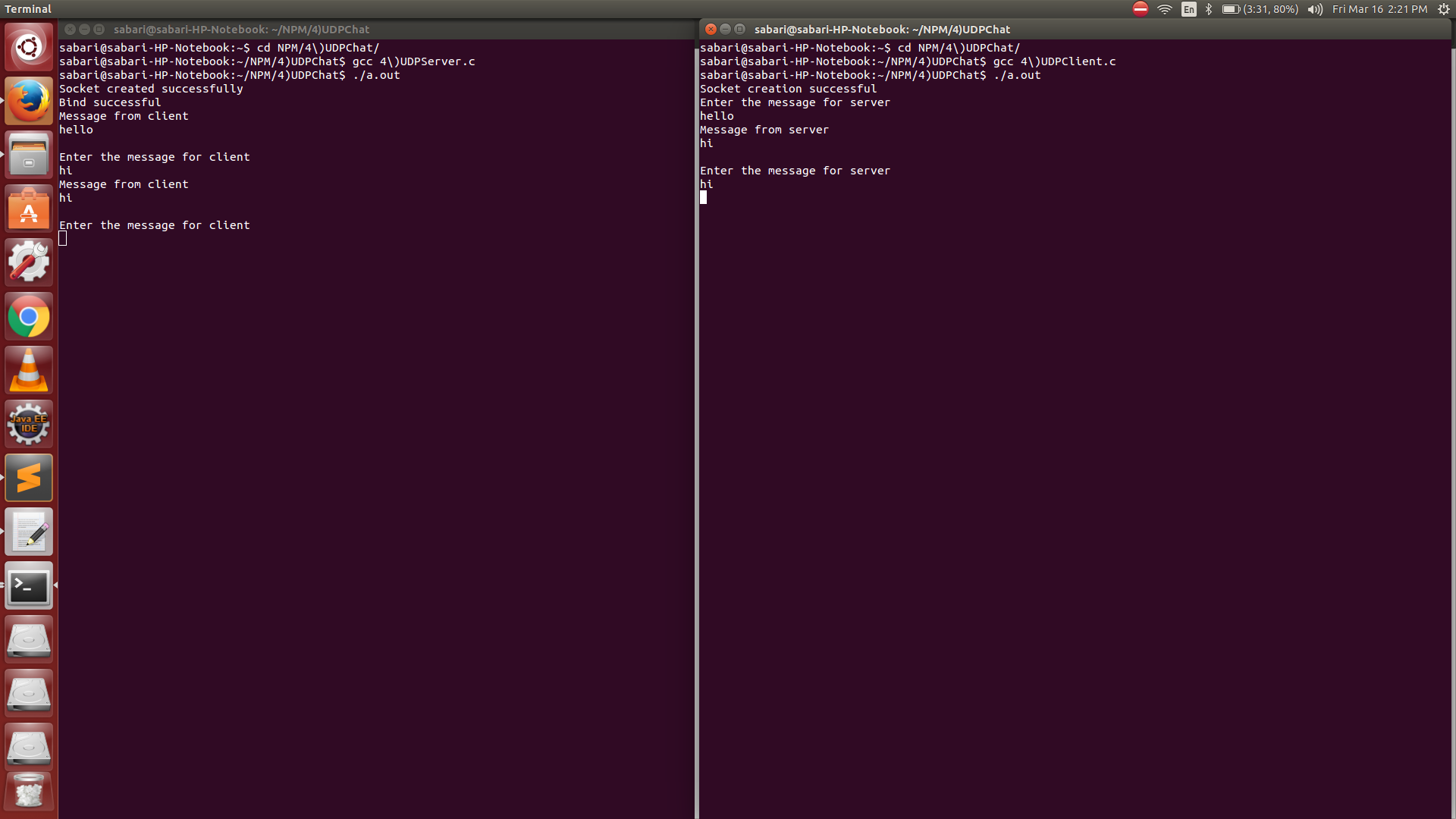
int res = recvfrom(clientSocket,buffer,sizeof(buffer),0,NULL,NULL);

printf("%s \n",buffer);\*/

}

}

**Output**



**Ex No 5 Design TCP Client and Server application to transfer file**

**Aim**

To design a TCP Client and Server Application to transfer files

**Program**

**Server**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<fcntl.h>

#include<errno.h>

void communicateClient(int connectionSocket){

char \*buff = (char \*)malloc(100);

int file;

char \*filename = (char \*)malloc(sizeof(char));

read(connectionSocket,buff,sizeof(buff));

int len = strlen(buff);

int i=0;

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

filename[i]=buff[i];

filename[i]='\0';

printf("Filename %s \n",filename);

file = open(filename,O\_RDONLY);

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

if(file <= 0){

printf("File couldn't be opened with error %s \n",strerror(errno));

strcpy(buff,"File couldn't be opened");

}

else

read(file,buff,sizeof(buff));

write(connectionSocket,buff,sizeof(buff));

}

void main(){

int serverSocket,connectionSocket,clientAddressLength;

int bind\_result,listen\_result;

struct sockaddr\_in serverAddress,clientAddress;

//Create socket

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

if(serverSocket == -1)

printf("Socket creation failed \n");

else{

printf("Socket creation successful \n");

//Configure server address

serverAddress.sin\_family = AF\_INET;

serverAddress.sin\_port = htons(8000);

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

//Binds the created socket with the given address

bind\_result = bind(serverSocket,(struct sockaddr \*)&serverAddress,sizeof(serverAddress));

if(bind\_result == -1)

printf("Bind process failed \n");

else{

printf("Bind successful \n");

//Listens for client connection on the specified socket

listen\_result = listen(serverSocket,10); //10 is the backlog value that specifies maximum number of clients can wait in the connection queue

if(listen\_result == -1 )

printf("Server is not listening \n");

else{

printf("Server is listening \n");

clientAddressLength = sizeof(clientAddress);

connectionSocket = accept(serverSocket,(struct sockaddr \*)&clientAddress,&clientAddressLength);

if(connectionSocket == -1 )

printf("New connection rejected \n");

else{

printf("New connection accepted \n");

communicateClient(connectionSocket);

}

}

}

}

}

**Client**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

void communicateServer(int clientSocket){

char \*buff = (char \*)malloc(sizeof(char));

printf("Enter the filename \n");

fgets(buff,255,stdin);

write(clientSocket,buff,sizeof(buff));

read(clientSocket,buff,sizeof(buff));

printf("Reply from server \n %s",buff);

}

void main(){

int clientSocket,connectionResult;

struct sockaddr\_in serverAddress;

//Create a socket

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

if(clientSocket == -1 )

printf("Socket creation failed \n");

else{

printf("Socket creation successful \n");

serverAddress.sin\_family = AF\_INET;

serverAddress.sin\_port = htons(8000);

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

//Connect with server

connectionResult = connect(clientSocket,(struct sockaddr \*)&serverAddress,sizeof(serverAddress));

if(connectionResult == -1 )

printf("Connection failed \n");

else{

printf("Connection scuccessful \n");

communicateServer(clientSocket);

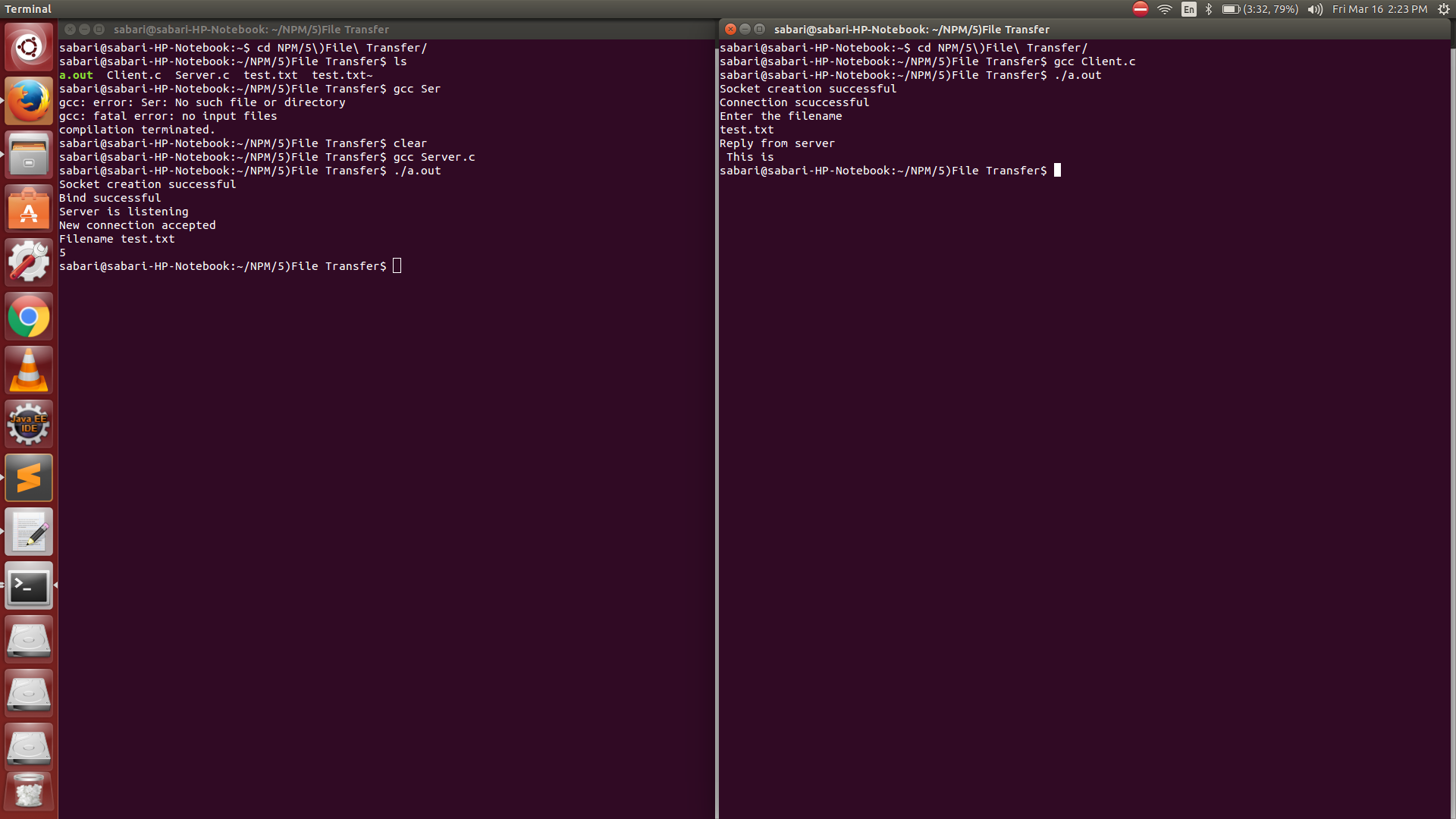
close(clientSocket);

}

}

**}**

**Output**



**Ex No 6 Demonstration to generate SIGPIPE Error with Socket**

**Aim**

To generate a SIGPIPE error with socket

**Program**

**Server**

#include<stdio.h>

#include<string.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

void communicateClient(int connectionSocket){

char receive\_buffer[1024];

char send\_buffer[1024];

while(1){

bzero(receive\_buffer,1024);

read(connectionSocket,receive\_buffer,sizeof(receive\_buffer));

printf("Message from client \n");

printf("%s \n",receive\_buffer);

bzero(send\_buffer,1024);

printf("Enter the message for client \n");

fgets(send\_buffer,1024,stdin);

write(connectionSocket,send\_buffer,sizeof(send\_buffer));

if(strncmp(send\_buffer,"exit",4)==0)

break;

}

}

void main(){

int serverSocket,connectionSocket,clientAddressLength;

int bind\_result,listen\_result;

struct sockaddr\_in serverAddress,clientAddress;

//Create socket

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

if(serverSocket == -1)

printf("Socket creation failed \n");

else

printf("Socket creation successful \n");

//Configure server address

serverAddress.sin\_family = AF\_INET;

serverAddress.sin\_port = htons(8000);

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

//Binds the created socket with the given address

bind\_result = bind(serverSocket,(struct sockaddr \*)&serverAddress,sizeof(serverAddress));

if(bind\_result == -1)

printf("Bind process failed \n");

else

printf("Bind successful \n");

//Listens for client connection on the specified socket

listen\_result = listen(serverSocket,10); //10 is the backlog value that specifies maximum number of clients can wait in the connection queue

if(listen\_result == -1 )

printf("Server is not listening \n");

else

printf("Server is listening \n");

clientAddressLength = sizeof(clientAddress);

connectionSocket = accept(serverSocket,(struct sockaddr \*)&clientAddress,&clientAddressLength);

if(connectionSocket == -1 )

printf("New connection rejected \n");

else{

printf("New connection accepted \n");

communicateClient(connectionSocket);

}

}

**Client**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<signal.h>

void sig\_pipe(){

printf("Server terminated prematurely \n");

exit(0);

}

void communicateServer(int clientSocket){

char send\_buffer[1024];

char receive\_buffer[1024];

while(1){

bzero(send\_buffer,1024);

printf("Enter the message for server \n");

fgets(send\_buffer,1024,stdin);

write(clientSocket,send\_buffer,sizeof(send\_buffer));

bzero(receive\_buffer,1024);

read(clientSocket,receive\_buffer,sizeof(receive\_buffer));

printf("Message from server \n");

printf("%s \n",receive\_buffer);

/\*if(strncmp(receive\_buffer,"exit",4)==0)

break;\*/

}

}

void main(){

int clientSocket,connectionResult;

struct sockaddr\_in serverAddress;

//Create a socket

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

if(clientSocket == -1 )

printf("Socket creation failed \n");

else

printf("Socket creation successful \n");

serverAddress.sin\_family = AF\_INET;

serverAddress.sin\_port = htons(8000);

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

signal(SIGPIPE,sig\_pipe);

//Connect with server

connectionResult = connect(clientSocket,(struct sockaddr \*)&serverAddress,sizeof(serverAddress));

if(connectionResult == -1 )

printf("Connection failed \n");

else{

printf("Connection scuccessful \n");

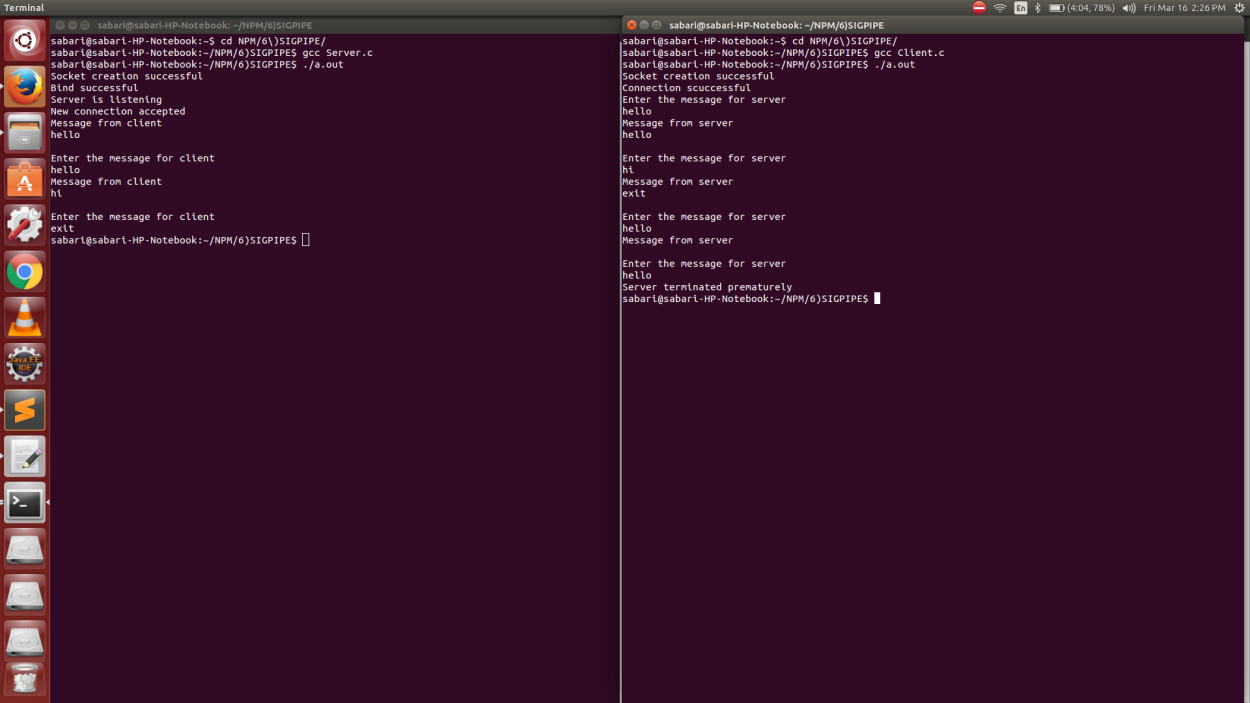
communicateServer(clientSocket);

//close(clientSocket);

}

}

Output



**Ex No 7 Demonstration to restart server by capturing SIGHUP signal**

**Aim**

To restart a server by capturing SIGHUP signal

**Program**

**Server**

#include<netinet/in.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<unistd.h>

#include<arpa/inet.h>

#include<stdio.h>

#include<string.h>

#include<signal.h>

#include<stdlib.h>

int sockfd;

void myhand()

{

printf("SIGHUPcaught\n");

printf("restarting server\n");

close(sockfd);

execl("/udpserver","udpserver",NULL);

printf("Server is not restarted\n");

}

int main()

{

char msg[100]="",rply[100]="";

int len;

struct sockaddr\_in server,client;

signal(SIGHUP,myhand);

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

if(sockfd<0)

{

printf("socket error correction\n");

}

printf("socket created successsssssssssfully\n");

bzero(&server,sizeof(server));

server.sin\_port=htons(3230);

server.sin\_family=AF\_INET;

inet\_aton("127.0.0.1",&server.sin\_addr);

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

while(1)

{

recvfrom(sockfd,(char \*)msg,100,0,(struct sockaddr \*)&client,&len);

puts(msg);

sendto(sockfd,(char \*)rply,strlen(msg),0,(struct sockaddr \*)&client,len);

}

close(sockfd);

}

**Client**

#include<netinet/in.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<unistd.h>

#include<arpa/inet.h>

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

int main()

{

char msg[100]="",msg1[100]="";

int sockfd,confd;

struct sockaddr\_in server,client;

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

if(sockfd<0)

{

printf("socket error correction\n");

}

printf("socket created successfully\n");

bzero(&server,sizeof(server));

server.sin\_port=htons(3230);

server.sin\_family=AF\_INET;

inet\_aton("127.0.0.1",&server.sin\_addr);

int len=sizeof(server);

while(1)

{

printf("enter the msg\n");

gets(msg);

sendto(sockfd,(char \*)msg,strlen(msg),0,(struct sockaddr \*)&server,len);

recvfrom(sockfd,(char \*)msg,100,0,(struct sockaddr \*)&server,&len);

printf("msg from the client is %s",msg);

}

close(sockfd);

}

