**1.Program to implement TCP echo.**

**a)Exchange single message between server and client.**

**Server:**

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include <sys/types.h>

int main(int argc, char \*argv[])

{

int sfd,newsfd,s,len,n,i; char buff[100];

struct sockaddr\_in servaddr, ca,client;

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

if(sfd<0)

{

perror("socket() error:");

exit(-1);

}

//creates a socket

servaddr.sin\_family=AF\_INET;

servaddr.sin\_port=htons(atoi(argv[1]));

//convert port address given from command line into network byte order

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

//extracts the machine IP address

//servaddr.sin\_addr.s\_addr=inet\_addr("172.16.0.100");

s=bind(sfd,(struct sockaddr\*)&servaddr,sizeof(servaddr));

//bind the server with the specified address

if(s<0)

{

perror("bind() error:");

exit(-1);

}

s=listen(sfd,5);

//tells its readiness to TCP

if(s<0)

{

perror("listen() error:");

exit(-1);

}

printf("Server is ready and waiting for client requests...........\n");

newsfd=accept(sfd,(struct sockaddr\*)&ca,&len); //server blocks fr connection request

if(newsfd<0)

{

perror("accept() error:");

exit(-1);

}

printf("Connected to client........");

n=recv(newsfd,&buff,100,0);

//reads data into ‘buff’ from the scoket

buff[n]='\0';

printf("\nMessage from client: %s",buff);

//displays the data on to the terminal

for(i=0;buff[i]!='\0';++i)

buff[i]=toupper(buff[i]);

//converts the data in the ‘buff’ to upper case

send(newsfd,&buff, n,0);

//writes the data from ‘buff’ over to socket.

close(sfd); close(newsfd);

//close the connection

return 0;

}

**Client:**

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include <sys/types.h>

#define MAX 100

int main(int argc, char \*argv[])

{

int sfd,newsfd,s,len,n; char buff[MAX+1];

struct sockaddr\_in servaddr,sa;

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

//creates an unnamed socket

if(sfd<0)

{

perror("socket() error:");

exit(-1);

}

servaddr.sin\_family=AF\_INET;

servaddr.sin\_port=htons(atoi(argv[1]));

servaddr.sin\_addr.s\_addr=inet\_addr("172.16.2.132");

s=connect(sfd,(struct sockaddr\*)&servaddr,sizeof(servaddr));

//requests for connection with the server (172.16.0.100)

if(s<0)

{

perror("connect() error:");

exit(-1);

}

write(1,"Enter a message: ",17);

n=read(0,&buff,100);

//read a message from keyboard (i.e. From user)

send(sfd,&buff,n,0);

//writes the message over to the socket from the buffer

n=recv(sfd,&buff,MAX,0);

//reads a message from the socket into the buffer

write(1,"Received from server: ",22);

write(1,&buff,n);

//display the message on the terminal

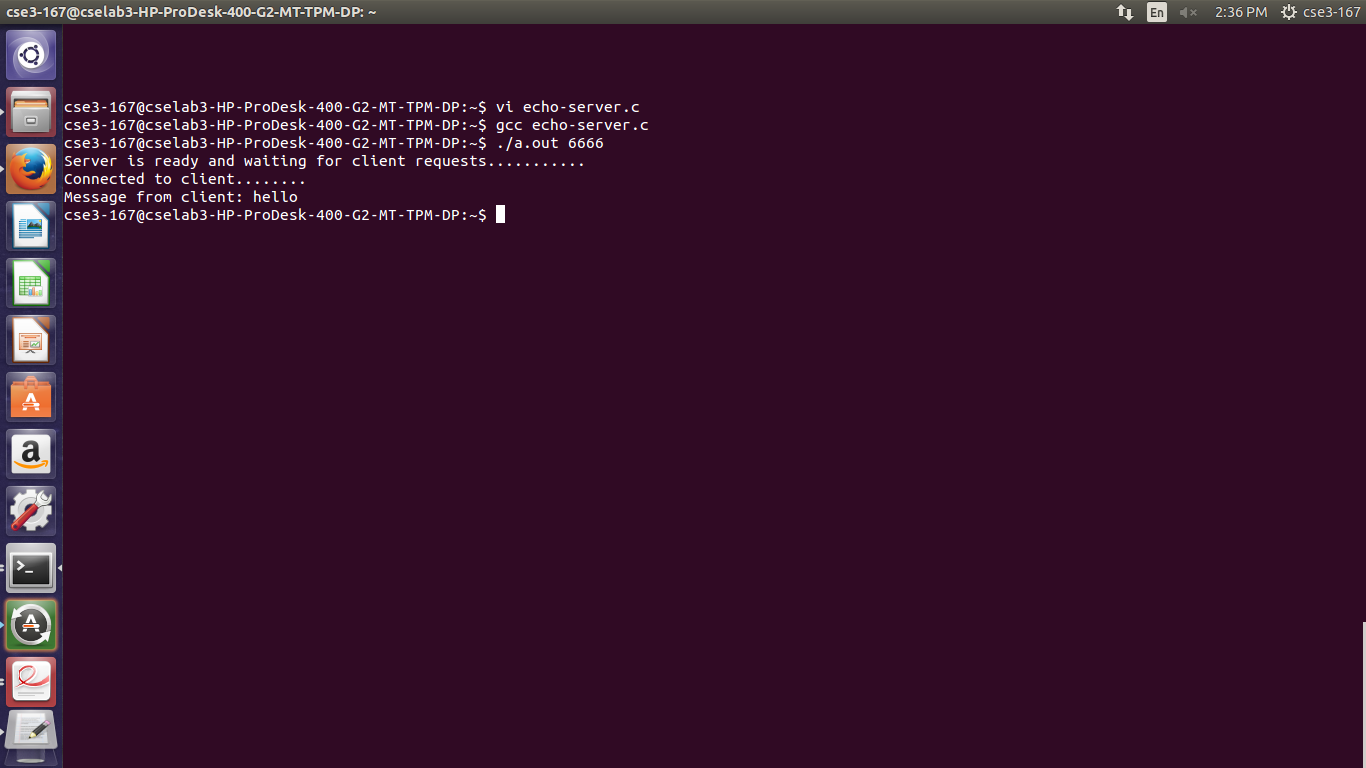
close(sfd);

return 0;

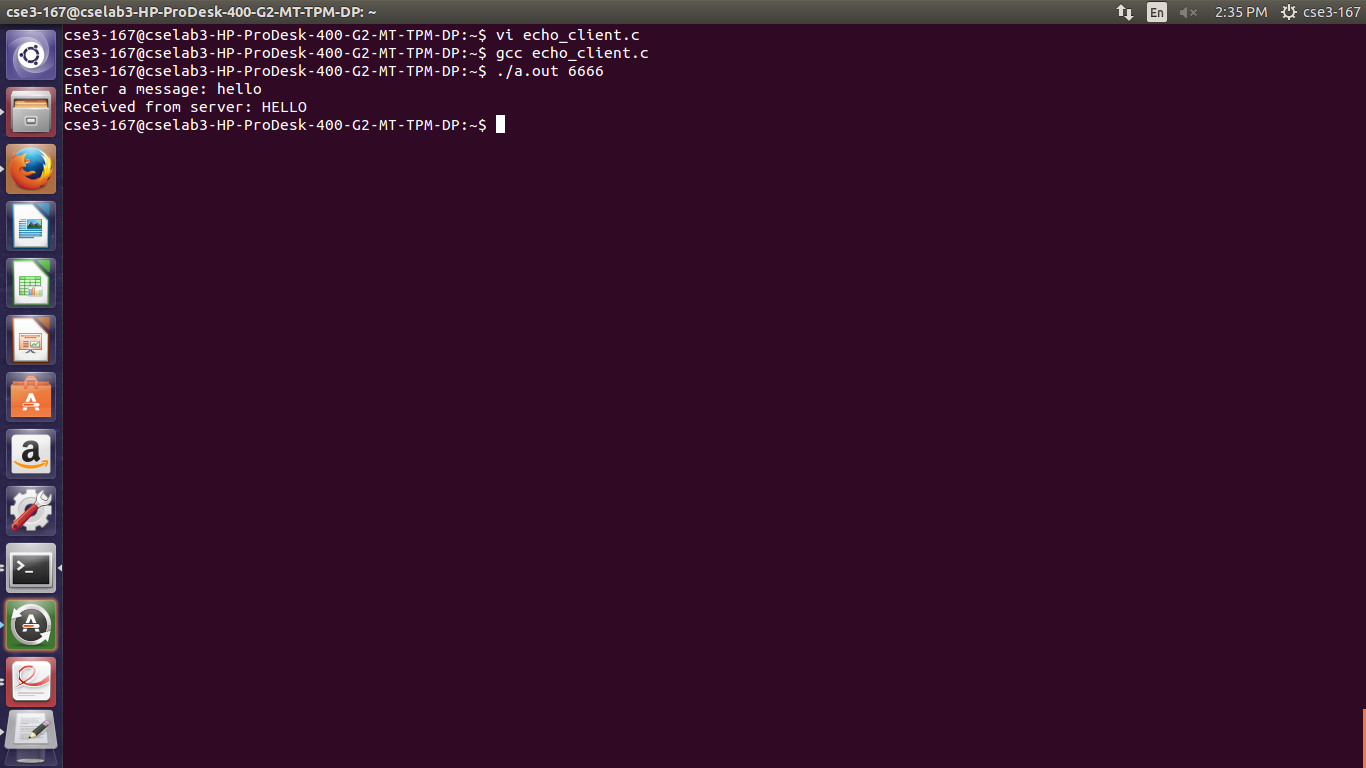
}

**OUTPUT:**

**Server**



**Client**



**b)Exchange multiple messages between server and client.**

**Server:**

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include <sys/types.h>

int main(int argc, char \*argv[])

{

int sfd,newsfd,s,len,n,i,nmsgs=10,nc=0; char buff[100];

struct sockaddr\_in servaddr, ca,client;

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

if(sfd<0)

{

perror("socket() error:");

exit(-1);

}

//creates a socket

servaddr.sin\_family=AF\_INET;

servaddr.sin\_port=htons(atoi(argv[1]));

//convert port address given from command line into network byte order

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

//extracts the machine IP address

servaddr.sin\_addr.s\_addr=inet\_addr("172.16.5.171");

s=bind(sfd,(struct sockaddr\*)&servaddr,sizeof(servaddr));

//bind the server with the specified address

if(s<0)

{

perror("bind() error:");

exit(-1);

}

s=listen(sfd,5);

//tells its readiness to TCP

if(s<0)

{

perror("listen() error:");

exit(-1);

}

printf("Server is ready and waiting for client requests...........\n");

newsfd=accept(sfd,(struct sockaddr\*)&ca,&len); //server blocks fr connection request

if(newsfd<0)

{

perror("accept() error:");

exit(-1);

}

printf("Connected to client........");

//for(nc=0;nc<nmsgs;nc++)

while(1)

{

printf("\nServer receiving..");

n=recv(newsfd,&buff,100,0);

//reads data into ‘buff’ from the scoket

buff[n]='\0';

printf("\nMessage from client: %s",buff);

//displays the data on to the terminal

for(i=0;buff[i]!='\0';++i)

buff[i]=toupper(buff[i]);

//converts the data in the ‘buff’ to upper case

send(newsfd,&buff, n,0);

}

//writes the data from ‘buff’ over to socket.

close(sfd); close(newsfd);

//close the connection

return 0;

}

**Client:**

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include <sys/types.h>

#define MAX 100

int main(int argc, char \*argv[])

{

int sfd,newsfd,s,len,n,nc=0,nmsgs=10; char buff[MAX+1];

struct sockaddr\_in servaddr,sa;

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

//creates an unnamed socket

if(sfd<0)

{

perror("socket() error:");

exit(-1);

}

servaddr.sin\_family=AF\_INET;

servaddr.sin\_port=htons(atoi(argv[1]));

servaddr.sin\_addr.s\_addr=inet\_addr("172.16.5.171");

s=connect(sfd,(struct sockaddr\*)&servaddr,sizeof(servaddr));

//requests for connection with the server (172.16.0.100)

if(s<0)

{

perror("connect() error:");

exit(-1);

}

printf("\n Enter 10 messages");

for(nc=0;nc<nmsgs;nc++)

{

write(1,"Enter a message: ",17);

n=read(0,&buff,100);

//read a message from keyboard (i.e. From user)

send(sfd,&buff,n,0);

//writes the message over to the socket from the buffer

n=recv(sfd,&buff,MAX,0);

//reads a message from the socket into the buffer

write(1,"Received from server: ",22);

write(1,&buff,n);

}

//display the message on the terminal

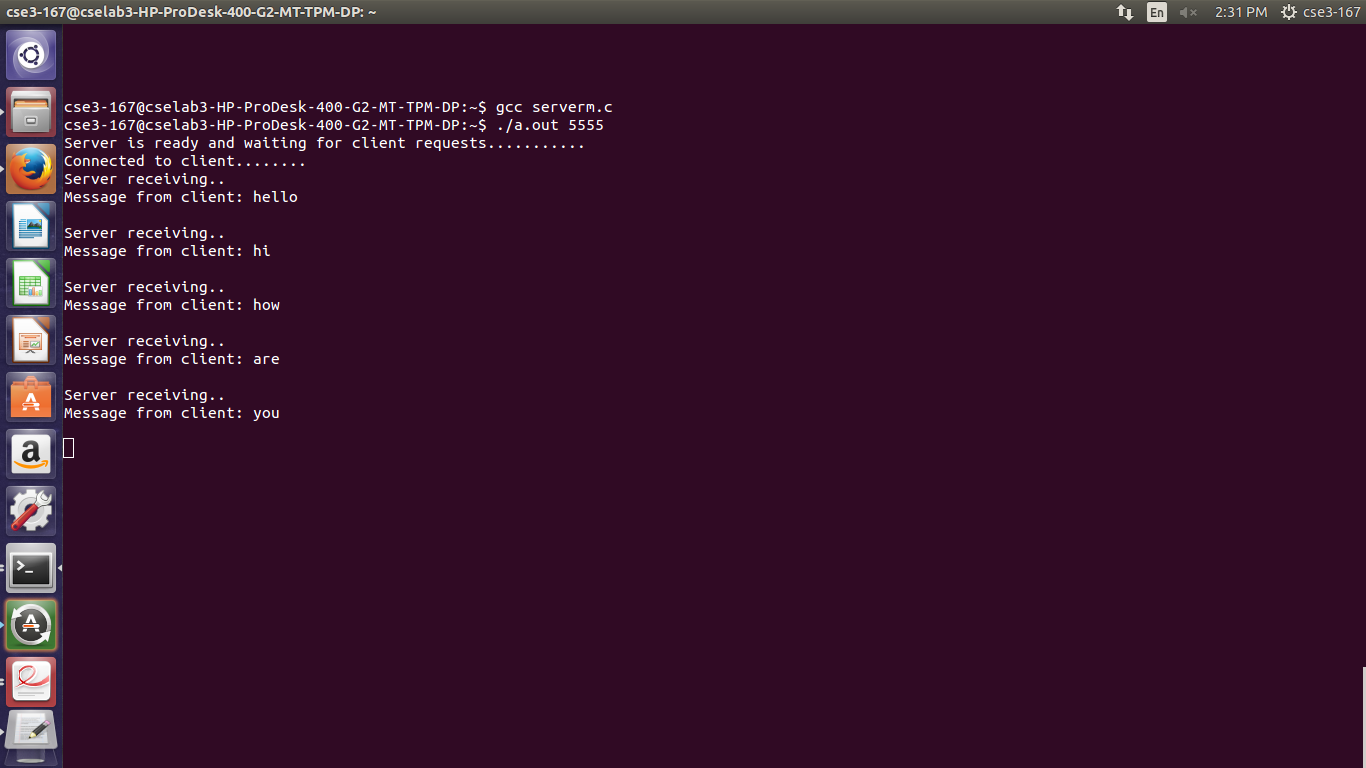
close(sfd);

return 0;

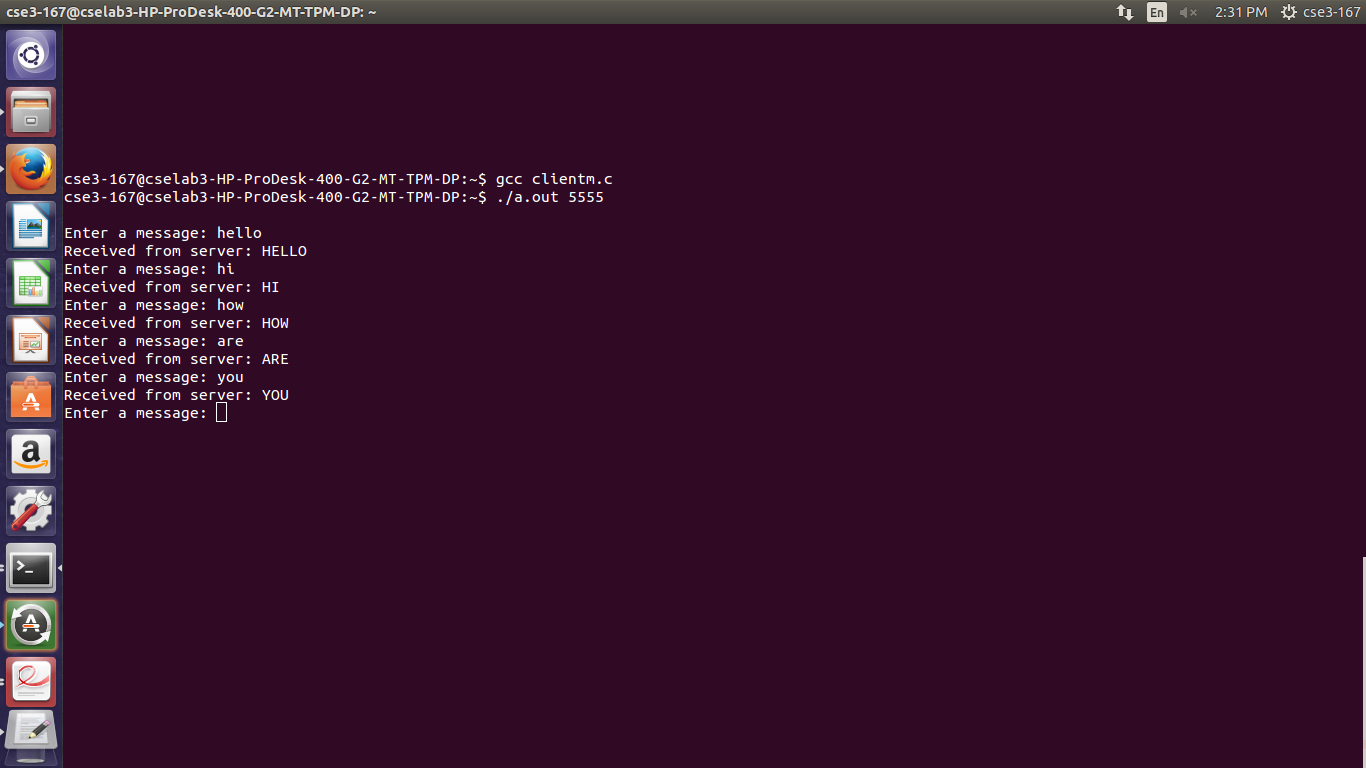
}

**OUTPUT:**

**Server**

****

**Client**

****

**c)To display date and time.**

**Server:**

#include<stdio.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<string.h>

#include<time.h>

void main()

{

int serverfd,b,newsock=-1;

char buff[100];

time\_t tick;

struct tm tm;

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

struct sockaddr\_in serveraddr;

serveraddr.sin\_family=AF\_INET;

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

serveraddr.sin\_port=htons(1234);

b=bind(serverfd,(struct sockaddr \*)&serveraddr,sizeof(serveraddr));

if(b==0)

{

printf("server binded \n");

if(listen(serverfd,1)==0)

{

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

newsock=accept(serverfd,(struct sockaddr \*) &serveraddr, &b);

if(newsock!=-1)

printf("Accepted \n");

tick=time(NULL);

tm = \*localtime(&tick);

//strcpy(buff,ctime(&tick));

strftime(buff,100,"%d-%m-%y %r",&tm);

send(newsock,buff,sizeof(buff),0);

}

}

}

**Client:**

#include<stdio.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<string.h>

void main()

{

int clientfd,clientconn;

char buff[100];

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

struct sockaddr\_in serveraddr;

serveraddr.sin\_family=AF\_INET;

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

serveraddr.sin\_port=htons(1234);

clientconn=connect(clientfd,(struct sockaddr \*)&serveraddr,sizeof(serveraddr));

if(clientconn==0)

{

printf("client connected \n");

recv(clientfd,&buff,sizeof(buff),0);

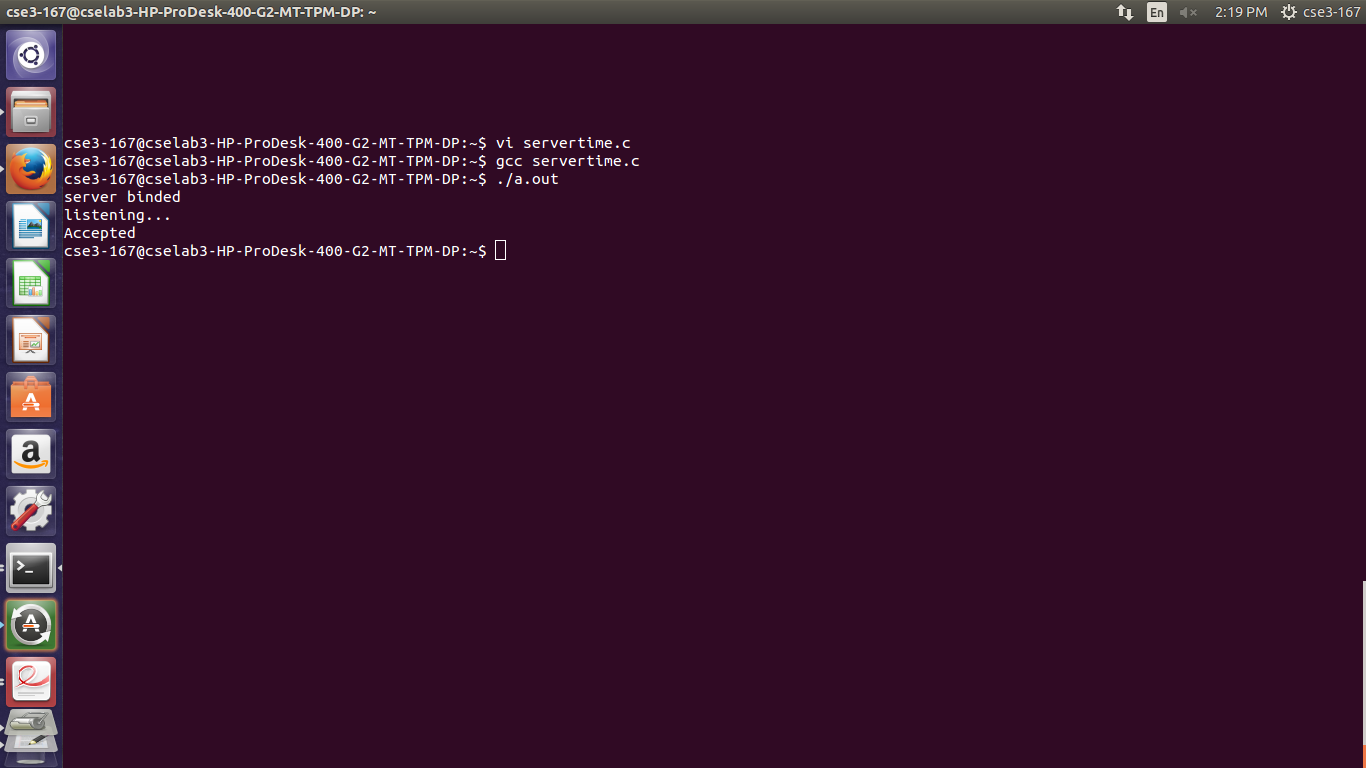
printf("server: %s \n",buff);

}

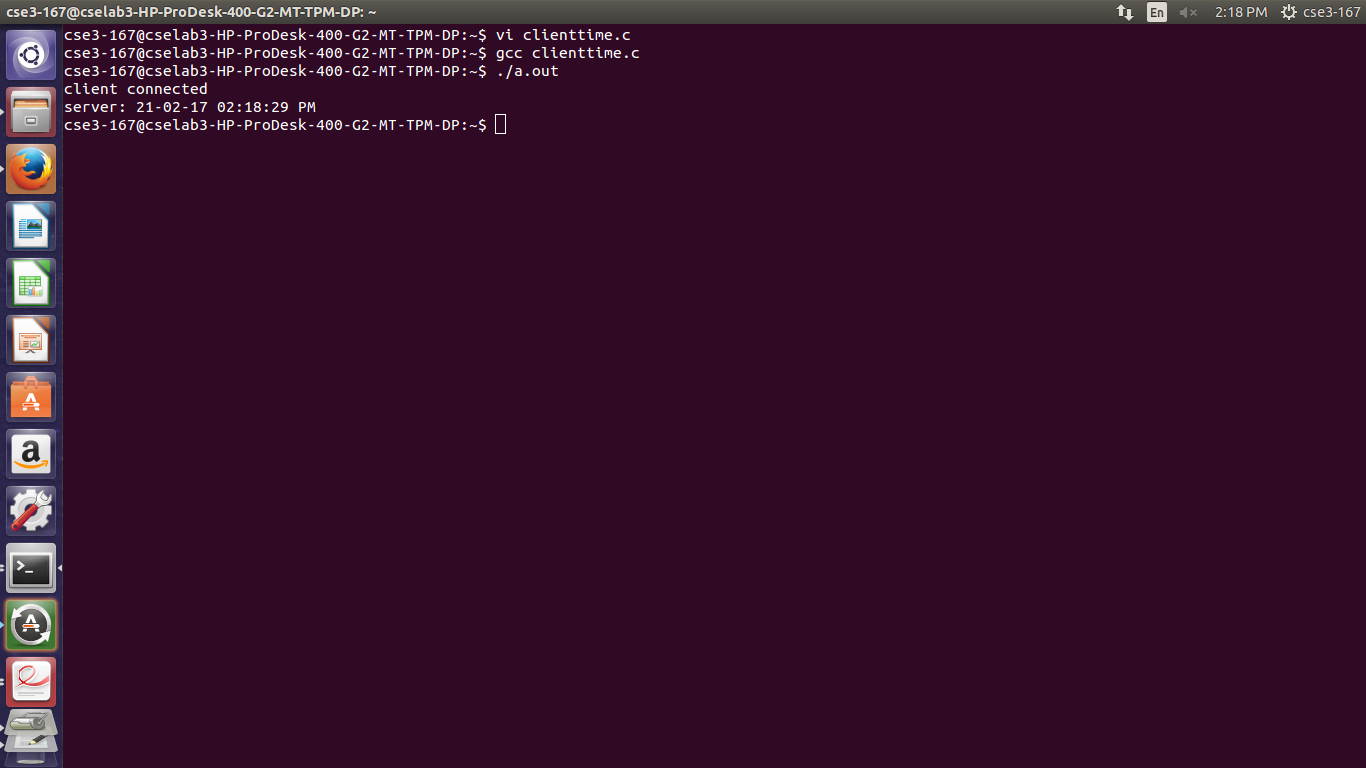
}

**OUTPUT:**

**Server**

****

**Client**

****

**2.** **Program to implement a simple chat service using UDP sockets.**

**Server:**

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<unistd.h>

#include<netdb.h>

#include<stdio.h>

#include<string.h>

#include<arpa/inet.h>

#define MAXLINE 1024

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

{

int sockfd;

int n;

socklen\_t len;

char msg[1024];

struct sockaddr\_in servaddr,cliaddr;

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

bzero(&servaddr,sizeof(servaddr));

servaddr.sin\_family=AF\_INET;

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

servaddr.sin\_port=htons(5035);

printf("\n\n Binded");

bind(sockfd,(struct sockaddr\*)&servaddr,sizeof(servaddr));

printf("\n\n Listening...");

while(1)

{

printf("\n ");

len=sizeof(cliaddr);

n=recvfrom(sockfd,msg,MAXLINE,0,(struct sockaddr\*)&cliaddr,&len);

printf("\n Client's Message : %s\n",msg);

if(n<6)

perror("send error");

sendto(sockfd,msg,n,0,(struct sockaddr\*)&cliaddr,len);

if(strcmp(msg,"bye")==0){

printf("\nExiting");

break;}

}

return 0;

}

**Client:**

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<string.h>

#include<arpa/inet.h>

#include<string.h>

#include<arpa/inet.h>

#include<stdio.h>

#define MAXLINE 1024

int main(int argc,char\* argv[])

{

int sockfd;

int n;

socklen\_t len;

char sendline[1024],recvline[1024];

struct sockaddr\_in servaddr;

strcpy(sendline,"");

while(1)

{

printf("\n Enter the message : ");

scanf("%s",sendline);

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

bzero(&servaddr,sizeof(servaddr));

servaddr.sin\_family=AF\_INET;

servaddr.sin\_addr.s\_addr=inet\_addr("172.16.5.171");

servaddr.sin\_port=htons(5035);

connect(sockfd,(struct sockaddr\*)&servaddr,sizeof(servaddr));

len=sizeof(servaddr);

sendto(sockfd,sendline,MAXLINE,0,(struct sockaddr\*)&servaddr,len);

n=recvfrom(sockfd,recvline,MAXLINE,0,NULL,NULL);

recvline[n]=0;

if(strcmp(recvline,"bye")==0)

{

return 0;

}

printf("\n Server's Echo : %s\n\n",recvline);

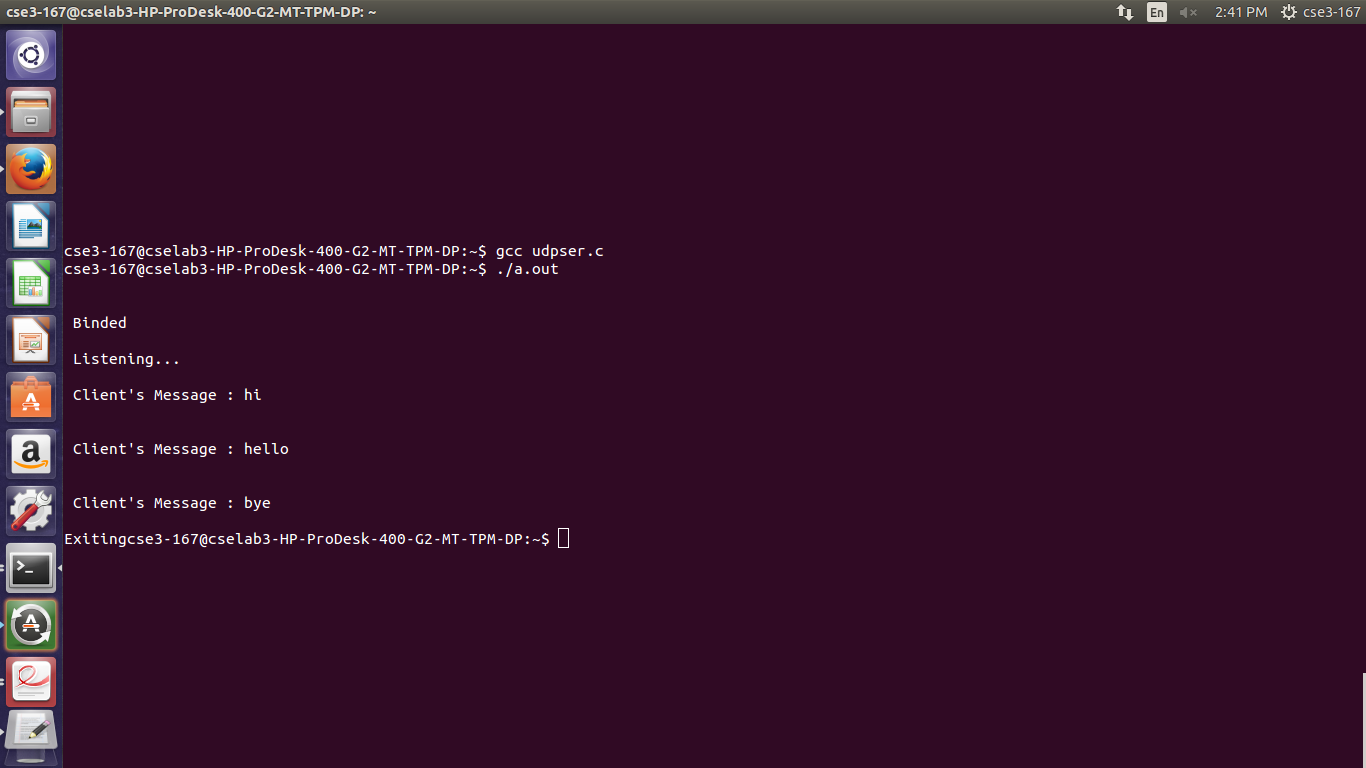
}

return 0;

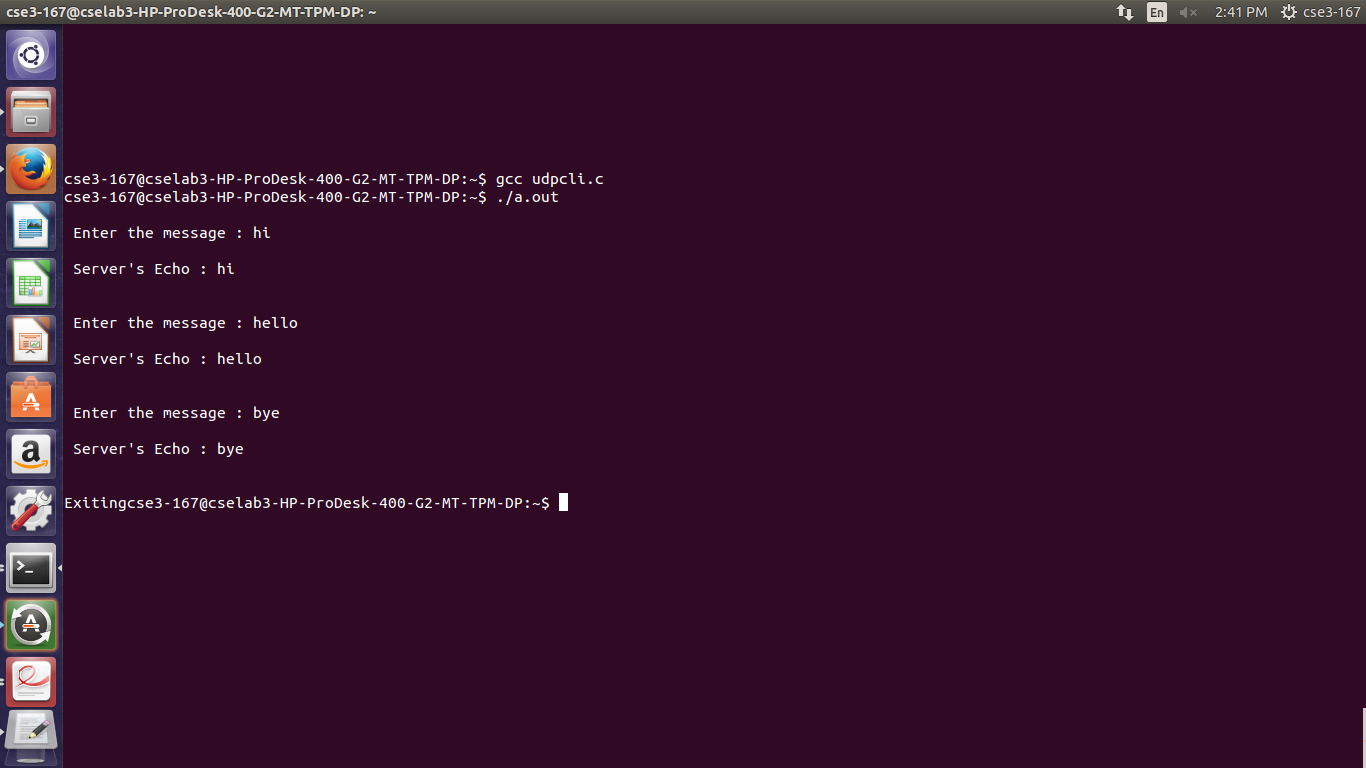
}

**OUTPUT:**

**Server**

****

**Client**

****

**3.Implementation of RSA algorithm.**

**a)Encrypt and decrypt numbers**

#include<stdio.h>

int phi,M,n,e,d,C,FLAG;

int check() //Function that checks whether e is relatively prime to ø (n)

{

int i;

for(i=3; e%i == 0 && phi % i == 0; i +2)

{

FLAG = 1;

return;

}

FLAG = 0;

}

void encrypt() //Function to encrypt the plain text message M

{

int i;

C = 1;

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

C=C\*M % n;

C = C % n;

printf("\n\tEncrypted keyword : %d",C);

}

void decrypt() //Function to decrypt the cipher text C into plaintext M

{

int i;

M = 1;

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

M = M\*C % n;

M = M % n;

printf("\n\tDecrypted keyword : %d\n",M);

}

void main()

{

int p,q,s;

printf("Enter any two Prime Numbers\t: ");

scanf("%d%d",&p,&q);

n = p\*q;

phi=(p-1)\*(q-1); //computation of ø value

printf("\n\tF(n) phi value\t= %d",phi);

do

{

printf("\nEnter e which is relatively prime and less than phi \t: ",n);

scanf("%d",&e);

check();

}while(FLAG==1);

d = 1;

do

{

s = (d\*e)%phi;

d++;

}while(s!=1);

d = d-1;

printf("\n\tPublic Key\t: {%d,%d}",e,n);

printf("\n\tPrivate Key\t: {%d,%d}",d,n);

printf("\n\nEnter The Plain Text\t: ");

scanf("%d",&M);

encrypt();

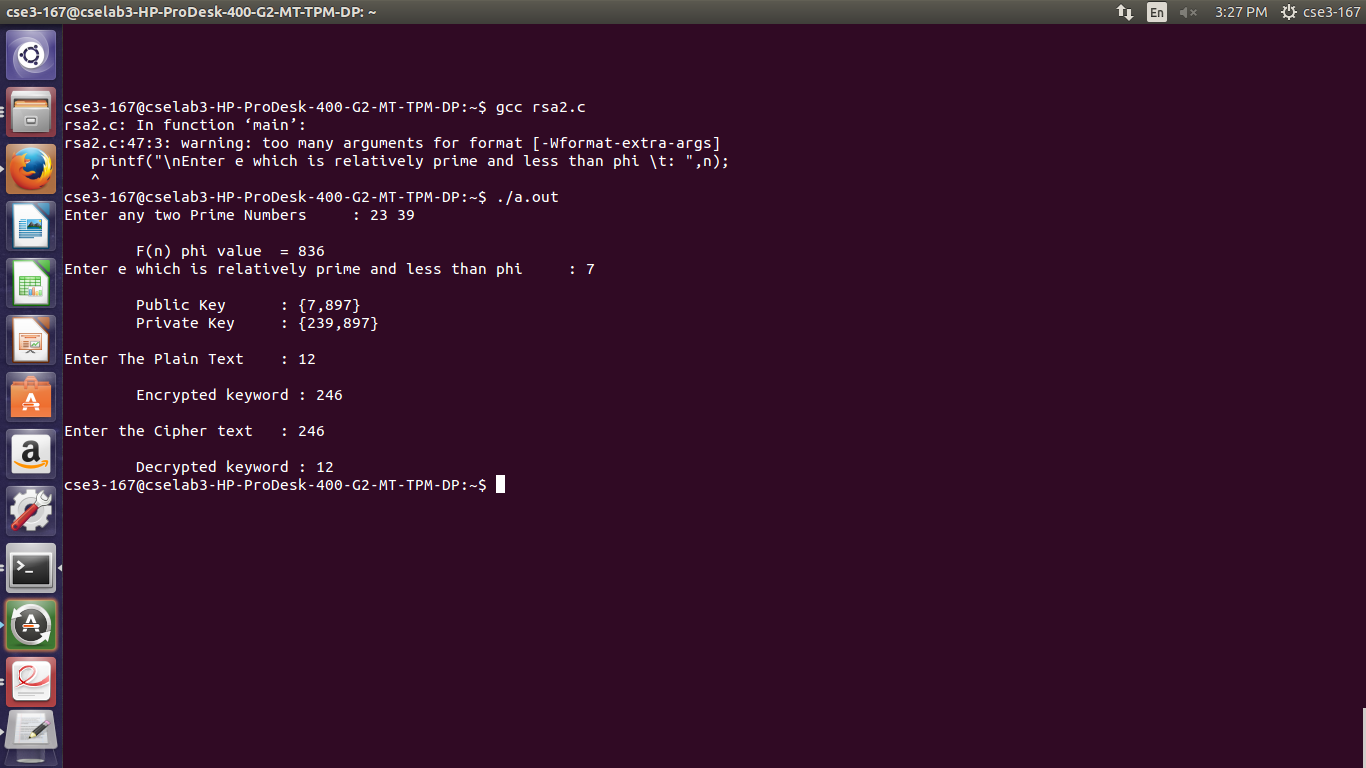
printf("\n\nEnter the Cipher text\t: ");

scanf("%d",&C);

decrypt();

}

**OUTPUT:**

****

**b)Encrypt and decrypt text messages**

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#include<string.h>

long int p,q,n,t,flag,e[100],d[100],temp[100],j,m[100],en[100],i;

char msg[100];

int prime(long int);

void ce();

long int cd(long int);

void encrypt();

void decrypt();

int main()

{

printf("\nENTER FIRST PRIME NUMBER\n");

scanf("%ld",&p);

flag=prime(p);

if(flag==0)

{

printf("\nWRONG INPUT\n");

exit(1);

}

printf("\nENTER ANOTHER PRIME NUMBER\n");

scanf("%ld",&q);

flag=prime(q);

if(flag==0||p==q)

{

printf("\nWRONG INPUT\n");

exit(1);

}

printf("\nENTER MESSAGE\n");

fflush(stdin);

scanf("%s",msg);

for(i=0;msg[i]!=NULL;i++)

m[i]=msg[i];

n=p\*q;

t=(p-1)\*(q-1);

ce();

printf("\nPOSSIBLE VALUES OF e AND d ARE\n");

for(i=0;i<j-1;i++)

printf("\n%ld\t%ld",e[i],d[i]);

encrypt();

decrypt();

return 0;

}

int prime(long int pr)

{

int i;

j=sqrt(pr);

for(i=2;i<=j;i++)

{

if(pr%i==0)

return 0;

}

return 1;

}

void ce()

{

int k;

k=0;

for(i=2;i<t;i++)

{

if(t%i==0)

continue;

flag=prime(i);

if(flag==1&&i!=p&&i!=q)

{

e[k]=i; flag=cd(e[k]);

if(flag>0)

{

d[k]=flag;

k++;

}

if(k==99)

break;

}

}

}

long int cd(long int x)

{

long int k=1;

while(1)

{

k=k+t;

if(k%x==0)

return(k/x);

}

}

void encrypt()

{

long int pt,ct,key=e[0],k,len;

i=0;

len=strlen(msg);

while(i!=len)

{

pt=m[i];

pt=pt-96;

k=1;

for(j=0;j<key;j++)

{

k=k\*pt;

k=k%n;

}

temp[i]=k;

ct=k+96;

en[i]=ct;

i++;

}

en[i]=-1;

printf("\nTHE ENCRYPTED MESSAGE IS\n");

for(i=0;en[i]!=-1;i++)

printf("%c",en[i]);

}

void decrypt()

{

long int pt,ct,key=d[0],k;

i=0;

while(en[i]!=-1)

{

ct=temp[i];

k=1;

for(j=0;j<key;j++)

{

k=k\*ct;

k=k%n;

}

pt=k+96;

m[i]=pt;

i++;

}

m[i]=-1;

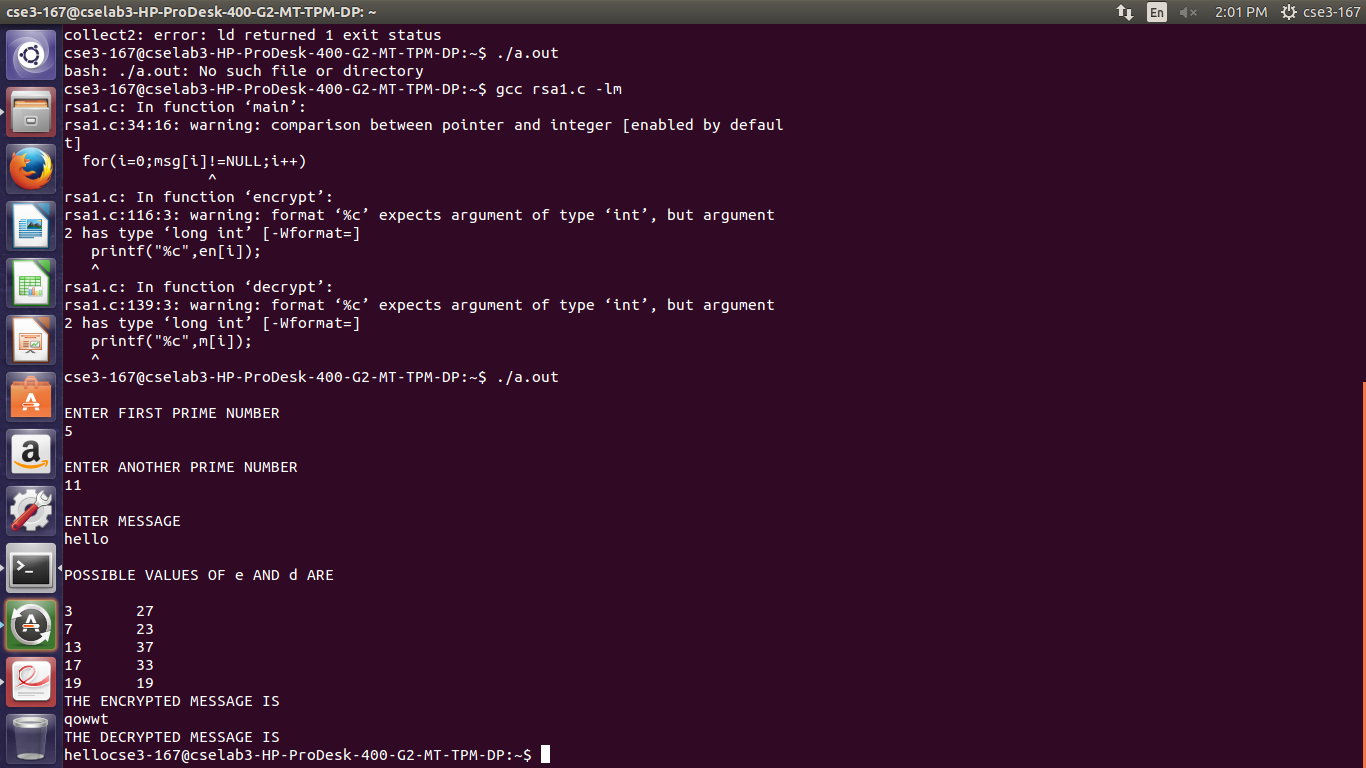
printf("\nTHE DECRYPTED MESSAGE IS\n");

for(i=0;m[i]!=-1;i++)

printf("%c",m[i]);

}

**OUTPUT:**

****

**4.Simulation of ARP/RARP.**

**Server:**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<arpa/inet.h>

#include<netinet/in.h>

#define SA structsockaddr

structIPmac

{

charip[100];

char mac[100];

};

int main()

{

intsockfd,len,i;

structsockaddr\_inservaddr;

char buff[30],temp[30],ip[30],mac[30];

int flag=0;

structIPmac in[3]={

{"10.1.1.8","44:dd:22:11:33"},{"127.0.0.1","33:aa:fe:4e:2d"},{"10.1.8.5","23:a3:5d:33:9d"} };

printf("ip\t\tmac\n");

for(i=0;i<3;i++)

{

printf("%s\t%s\n",in[i].ip,in[i].mac);

}

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

servaddr.sin\_family = AF\_INET;

servaddr.sin\_port = htons(9999);

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

bind(sockfd,(SA\*)&servaddr,sizeof(servaddr));

len=sizeof(servaddr);

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

for(i=0;i<strlen(ip)-1;i++)

{

temp[i]=ip[i];

}

temp[i]='\0';

printf("received IP :%s\n",temp);

for(i=0;i<3;i++)

{

if(strcmp(temp,in[i].ip)==0)

{

strcpy(mac,in[i].mac);

break;

}

}

printf("mac address is %s\n",mac);

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

bzero(mac,sizeof(mac));

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

printf("received mac address :%s",mac);

bzero(temp,sizeof(temp));

for(i=0;i<strlen(mac)-1;i++)

{

temp[i]=mac[i];

}

temp[i]='\0';

bzero(ip,sizeof(ip));

//check in table

for(i=0;i<3;i++)

{

if(strcmp(temp,in[i].mac)==0)

{

strcpy(ip,in[i].ip);

break;

}

}

printf("ip address :%s\n",ip);

sendto(sockfd,ip,sizeo

f(ip),0,(SA\*)&servaddr,len);

return 0;

}

**Client:**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<arpa/inet.h>

#include<netinet/in.h>

#define SA structsockaddr

int main()

{

intsockfd,len;

charip[30],mac[30];

structsockaddr\_inservaddr;

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

servaddr.sin\_family = AF\_INET;

servaddr.sin\_port = htons(9999);

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

printf("ARP SIMULATION\n");

printf("enter ip address :");

fgets(ip,sizeof(ip),stdin);

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

len=sizeof(servaddr);

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

printf("MAC address is: %s\n",mac);

printf("RARP simulation\n");

printf("enter mac address :");

bzero(mac,sizeof(mac));

fgets(mac,sizeof(mac),stdin);

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

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

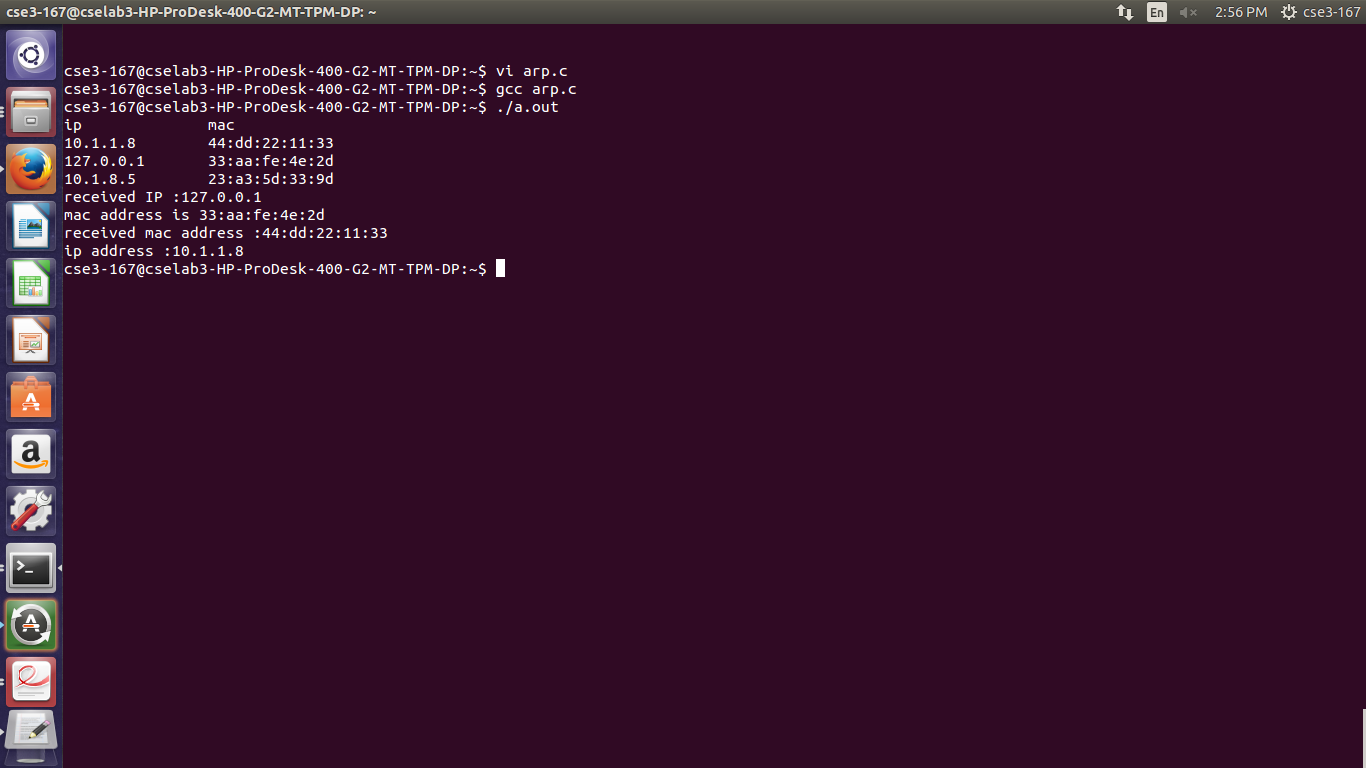
printf("IP address is: %s\n",ip);

return 0;

}

**OUTPUT:**

**Server**

****

**Client**

