PROGRAM CODE:

/* fserver.c */

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
int main()
{
   FILE *fp;
   char buff[40],msg[150],buf1[40];
   struct sockaddr_in server,client;
   int sock,rt1,rt2,rt3,rt4,len,rt5;
   sock=socket(AF_INET,SOCK_STREAM,0);
   if(sock<0)
   {
      perror("Error in creating the socket");
      exit(0);
   }
   printf("\nSocket created successfully");
   bzero(&server,sizeof(server));
   server.sin_family=AF_INET;
   server.sin_port=htons(43454);
   server.sin_addr.s_addr=htonl(INADDR_ANY);
   rt1=bind(sock,(struct sockaddr *)&server,sizeof(server));
   if(rt1<0)
   {
      perror("Error in binding the socket");
      exit(0);
   }
   printf("\nSocket binded successfully");
   if((rt3=listen(sock,4))<0)
   {
      perror("\nError in listening");
      exit(0);
   }
   printf("\nSocket listening successfully");
   len=sizeof(client);
   rt4=accept(sock,(struct sockaddr *)&client,&len);
   if(rt4<0)
   {
      perror("\nError in listening");
      exit(0);
   }
   rt5=recv(sock,buff,sizeof(buff),0);
   fp=fopen(buff,"r");
```

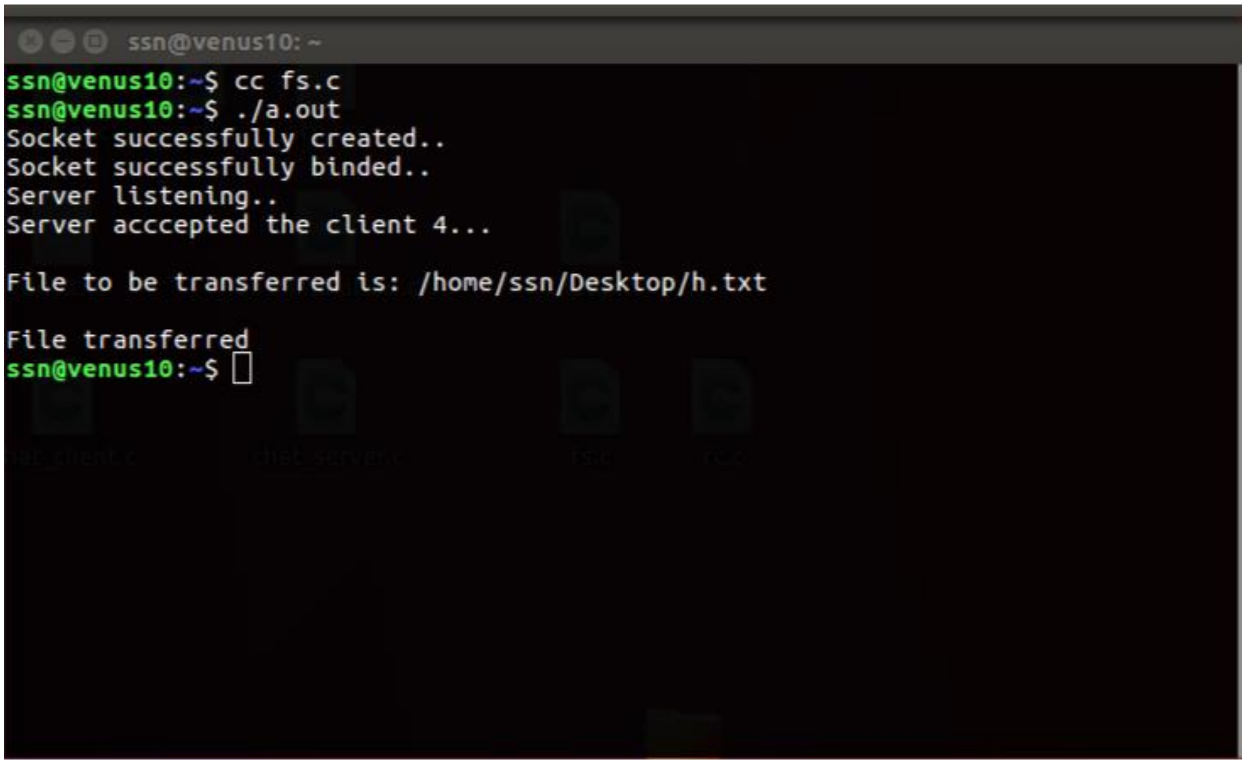
```c
   if(fp==NULL)
   {
      printf("\nError in opening the file");
      exit(0);
   }
   while(!feof(fp))
   {
      fscanf(fp,"%s",msg);
      send(sock,msg,sizeof(msg),0);
   }
   fclose(fp);
   close(sock);
   return 0;

}
```

/* fclient.c */

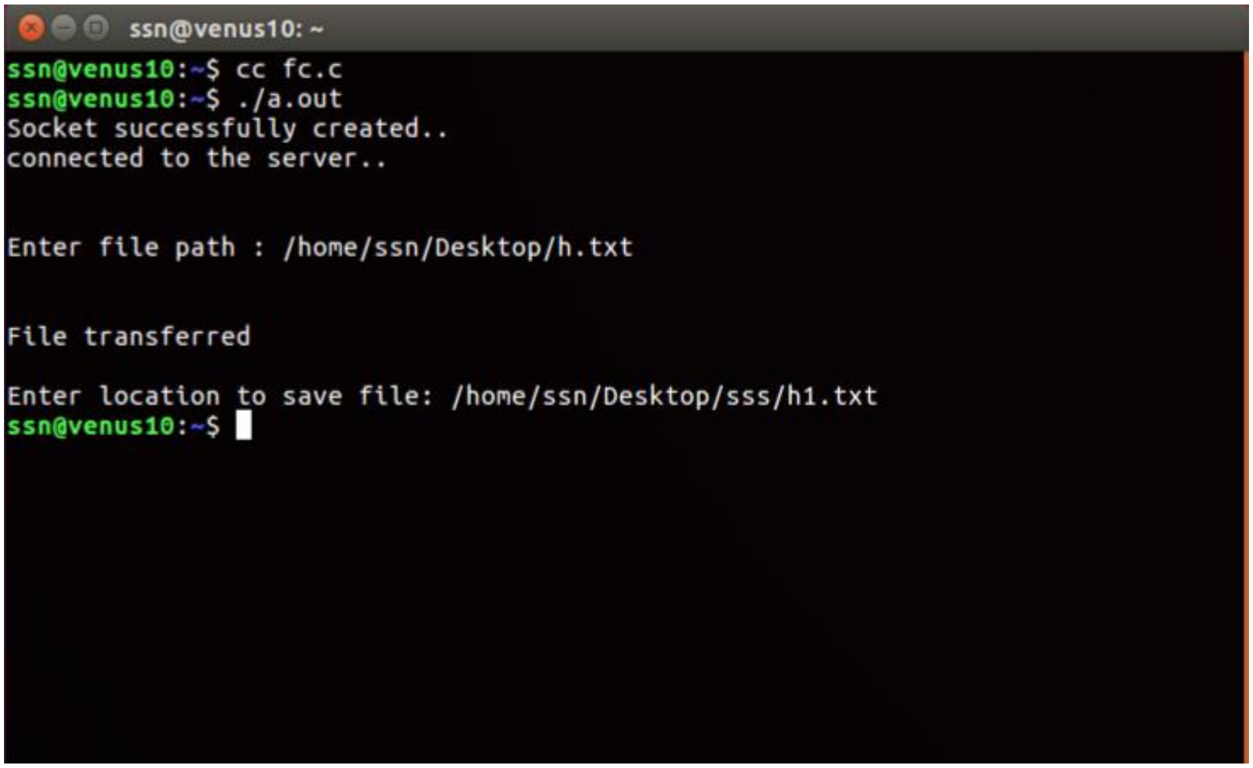
```c
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
int main()
{
   FILE *fp;
   char buff[40],msg[150],buf1[40];
   struct sockaddr_in server,client;
   int sock,rt1,rt2,rt3,rt4,len,rt5;
   sock=socket(AF_INET,SOCK_STREAM,0);
   if(sock<0)
   {
      perror("Error in creating the socket");
      exit(0);
   }
   printf("\nSocket created successfully");
   bzero(&server,sizeof(server));
   server.sin_family=AF_INET;
   server.sin_port=htons(43454);
   server.sin_addr.s_addr=htonl(INADDR_ANY);
   rt1=connect(sock,(struct sockaddr *)&server,sizeof(server));
   if(rt1<0)
   {
      perror("Error in connecting to the socket");
      exit(0);
   }
   printf("\nSocket connected to server successfully");
   printf("\nEnter the file name:");
   scanf("%s",buff);
   send(sock,buff,sizeof(buff),0);
```

```
    close(sock);
    return 0;
}
```

OUTPUT:



```
ssn@venus10:~$ cc fs.c
ssn@venus10:~$ ./a.out
Socket successfully created..
Socket successfully binded..
Server listening..
Server acccepted the client 4...

File to be transferred is: /home/ssn/Desktop/h.txt

File transferred
ssn@venus10:~$ 
```



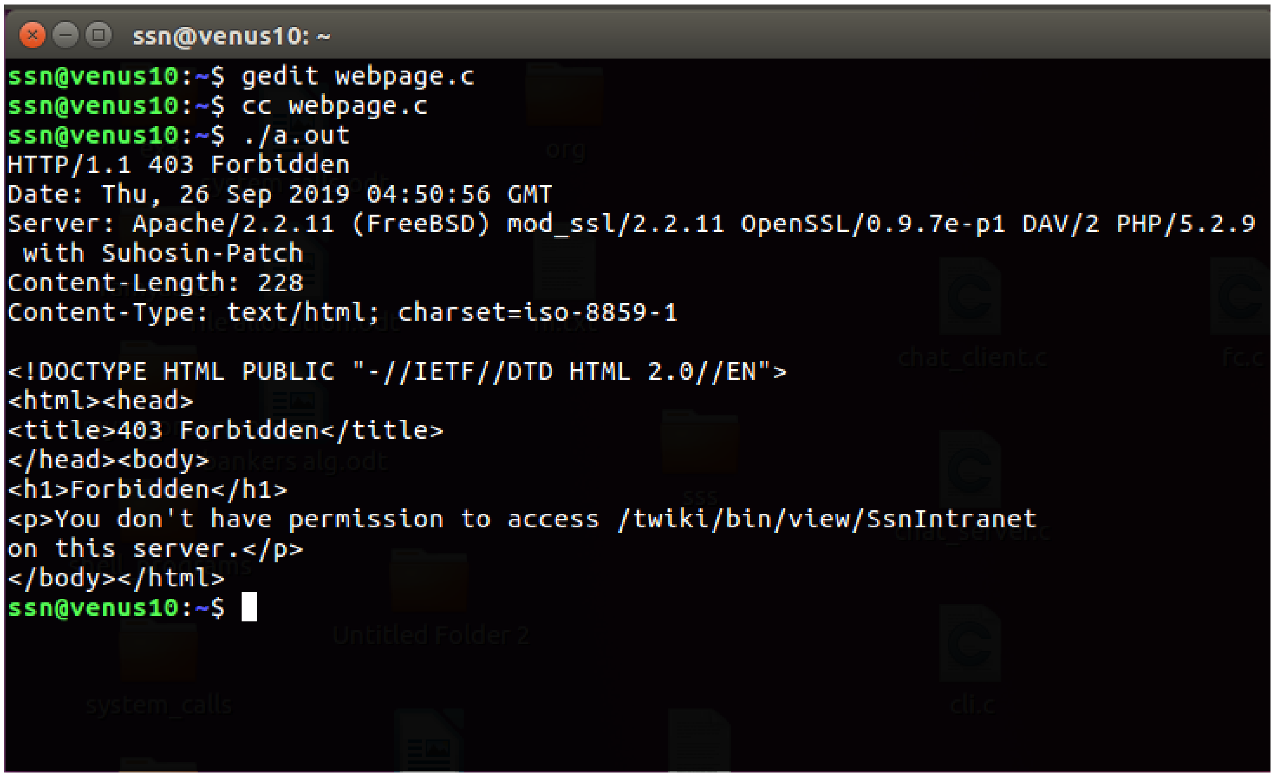
```
ssn@venus10:~$ cc fc.c
ssn@venus10:~$ ./a.out
Socket successfully created..
connected to the server..


Enter file path : /home/ssn/Desktop/h.txt


File transferred

Enter location to save file: /home/ssn/Desktop/sss/h1.txt
ssn@venus10:~$ 
```

PROGRAM CODE:

/* server */

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netdb.h>
int main()
{
    int sock;
    char host[] = "www.google.com";
    char port[] = "80";
    struct addrinfo hints, *res;
    char message[] = "GET / HTTP/1.1\nHost: www.google.com\n\n";
    unsigned int i;
    char buf[1024];
    int bytes_read;
    int status;
    memset(&hints, 0, sizeof hints);
    hints.ai_family = AF_INET;
    hints.ai_socktype = SOCK_STREAM;
    status = getaddrinfo(host, port, &hints, &res);
    if (status != 0) {
        perror("getaddrinfo");
        return 1;
    }
    sock = socket(res->ai_family, res->ai_socktype, res->ai_protocol);
    if (sock == -1) {
        perror("socket");
        return 1;
    }
    status = connect(sock, res->ai_addr, res->ai_addrlen);
    if (status == -1) {
        perror("connect");
        return 1;
    }
    freeaddrinfo(res);
    send(sock, message, strlen(message), 0);
    do {
            bytes_read = recv(sock, buf, 1024, 0);
            if (bytes_read == -1) {
                perror("recv");
            }
            else {
                printf("%.*s", bytes_read, buf);
            }
        } while (bytes_read > 0);
        close(sock);
        return 0;
}
```

OUTPUT:



ssn@venus10: ~

```
ssn@venus10:~$ gedit webpage.c
ssn@venus10:~$ cc webpage.c
ssn@venus10:~$ ./a.out
HTTP/1.1 403 Forbidden
Date: Thu, 26 Sep 2019 04:50:56 GMT
Server: Apache/2.2.11 (FreeBSD) mod_ssl/2.2.11 OpenSSL/0.9.7e-p1 DAV/2 PHP/5.2.9
 with Suhosin-Patch
Content-Length: 228
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /twiki/bin/view/SsnIntranet
on this server.</p>
</body></html>
ssn@venus10:~$ 
```

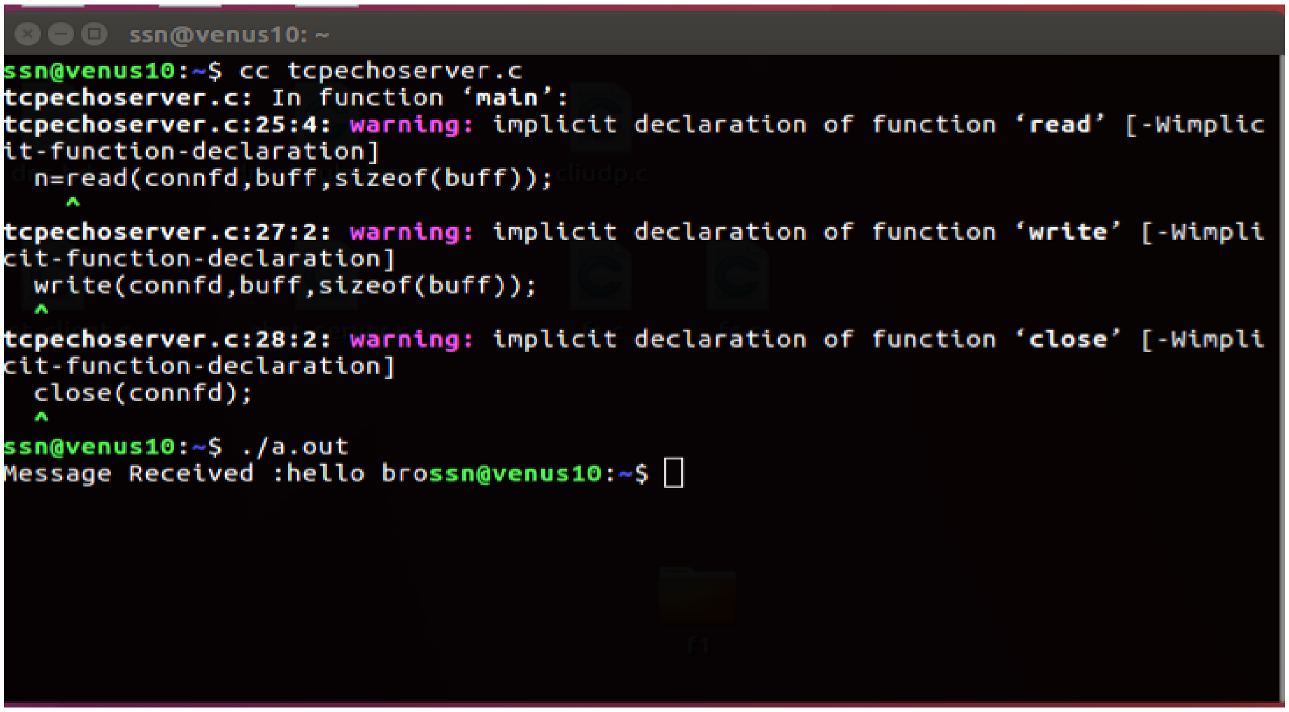
PROGRAM CODE:

```
/* client */
#include<stdio.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<stdlib.h>
#include<netinet/in.h>
#include<string.h>
#include<netdb.h>
void func(int sockfd)
{
   int n=0;
   char buff[100];
   bzero(buff,sizeof(buff));
   printf("\n Enter some text:");
   while((buff[n++]=getchar())!='\n');
   write(sockfd,buff,sizeof(buff));
}
int main()
{
  char msg[100];
  struct sockaddr_in ser,cli;
  int stat1,stat2,stat3,sockfd,len;
  sockfd=socket(AF_INET,SOCK_STREAM,0);
  if(sockfd<0)
  {
        perror("\n Error in creating a socket");
        exit(0);
  }
  else

     printf("\nSocket created successfully");
  bzero(&ser,sizeof(ser));
  ser.sin_family=AF_INET;
  ser.sin_port=htons(43454);
  ser.sin_addr.s_addr=inet_addr("127.0.0.1");
  len=sizeof(ser);
  stat2=connect(sockfd,(struct sockaddr *)&ser,len);
  if(stat2<0)
  {
    perror("\n Error in connecting to server");
  }
  else
    printf("\n Connected to the socket successfully");
  func(sockfd);
  close(sockfd);

}
```

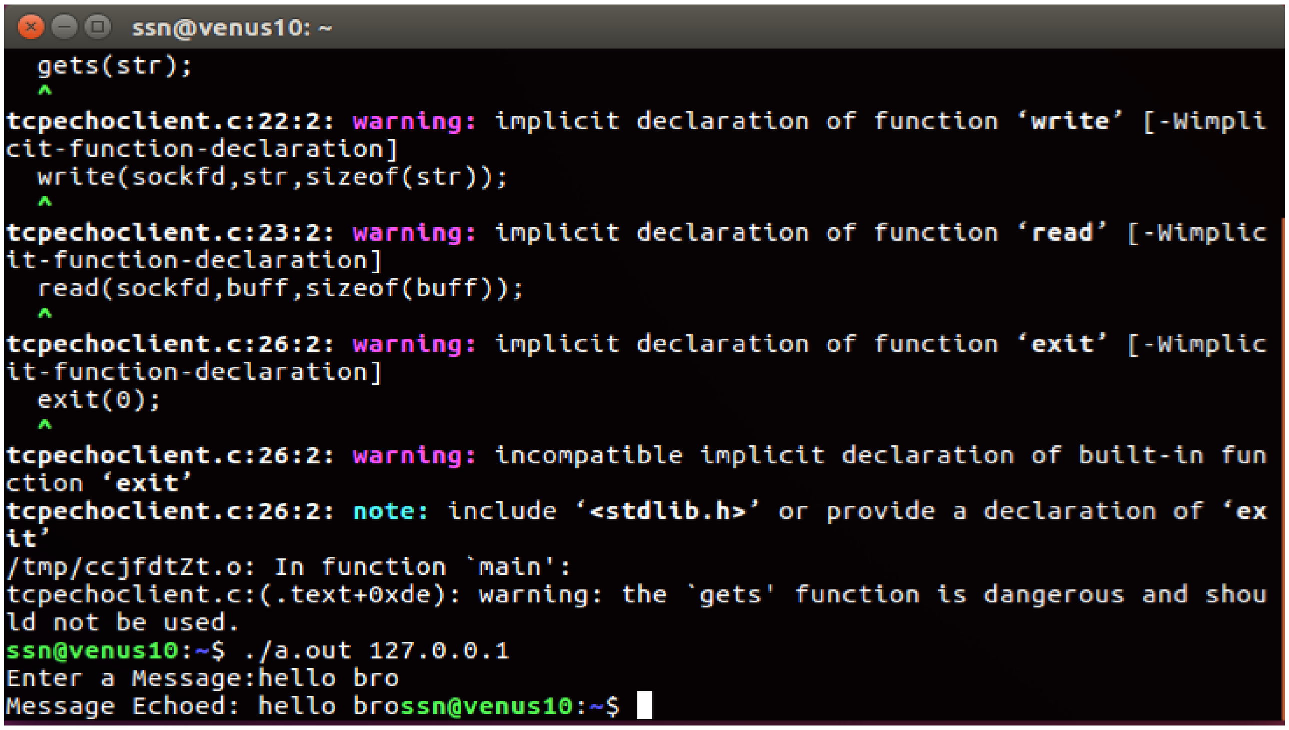
/* server */

```
#include<stdio.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<stdlib.h>
#include<netinet/in.h>
#include<string.h>
#include<netdb.h>
void func(int sockfd)
{
  int n=0;
  char buff[100];
  read(sockfd,buff,sizeof(buff));
  printf("\nFrom client:%s",buff);
}
int main()
{
  char buff[100];
  struct sockaddr_in ser,cli;
  int stat1,stat2,stat3,sockfd,len,sr;
  if((sockfd=socket(AF_INET,SOCK_STREAM,0))<0)
  {
       perror("\n Error in creating a socket");
       exit(0);
  }
  else

     printf("\nSocket created successfully");
  bzero(&ser,sizeof(ser));
  ser.sin_family=AF_INET;
  ser.sin_port=htons(43454);
  ser.sin_addr.s_addr=htonl(INADDR_ANY);
  stat1=bind(sockfd,(struct sockaddr *)&ser,sizeof(ser));
  if(stat1<0)
  {
    perror("\n Error in binding the socket");
    exit(0);
  }
  else
    printf("\n Successfully binded");
  sr=listen(sockfd,3);
  if(sr<0)
  {
    perror("\n Socket cannot listen error");
    exit(0);
  }
  else{
    printf("\n Listening mode on");
  }
  len=sizeof(cli);
  stat2=accept(sockfd,(struct sockaddr *)&cli,&len);
  if(stat2<0){
    perror("Error");
  }
  else
  {
    printf("\n Connection accepted");
  }
  func(stat2);
  close(sockfd);
}
```

OUTPUT:



```
ssn@venus10: ~
ssn@venus10:~$ cc tcpechoserver.c
tcpechoserver.c: In function ‘main’:
tcpechoserver.c:25:4: warning: implicit declaration of function ‘read’ [-Wimplic
it-function-declaration]
  n=read(connfd,buff,sizeof(buff));
    ^
tcpechoserver.c:27:2: warning: implicit declaration of function ‘write’ [-Wimpli
cit-function-declaration]
  write(connfd,buff,sizeof(buff));
  ^
tcpechoserver.c:28:2: warning: implicit declaration of function ‘close’ [-Wimpli
cit-function-declaration]
  close(connfd);
  ^
ssn@venus10:~$ ./a.out
Message Received :hello brossn@venus10:~$ 
```



```
ssn@venus10: ~
  gets(str);
  ^
tcpechoclient.c:22:2: warning: implicit declaration of function ‘write’ [-Wimpli
cit-function-declaration]
  write(sockfd,str,sizeof(str));
  ^
tcpechoclient.c:23:2: warning: implicit declaration of function ‘read’ [-Wimplic
it-function-declaration]
  read(sockfd,buff,sizeof(buff));
  ^
tcpechoclient.c:26:2: warning: implicit declaration of function ‘exit’ [-Wimplic
it-function-declaration]
  exit(0);
  ^
tcpechoclient.c:26:2: warning: incompatible implicit declaration of built-in fun
ction ‘exit’
tcpechoclient.c:26:2: note: include ‘<stdlib.h>’ or provide a declaration of ‘ex
it’
/tmp/ccjfdtZt.o: In function `main':
tcpechoclient.c:(.text+0xde): warning: the `gets' function is dangerous and shou
ld not be used.
ssn@venus10:~$ ./a.out 127.0.0.1
Enter a Message:hello bro
Message Echoed: hello brossn@venus10:~$ 
```

PROGRAM CODE:

/* server */

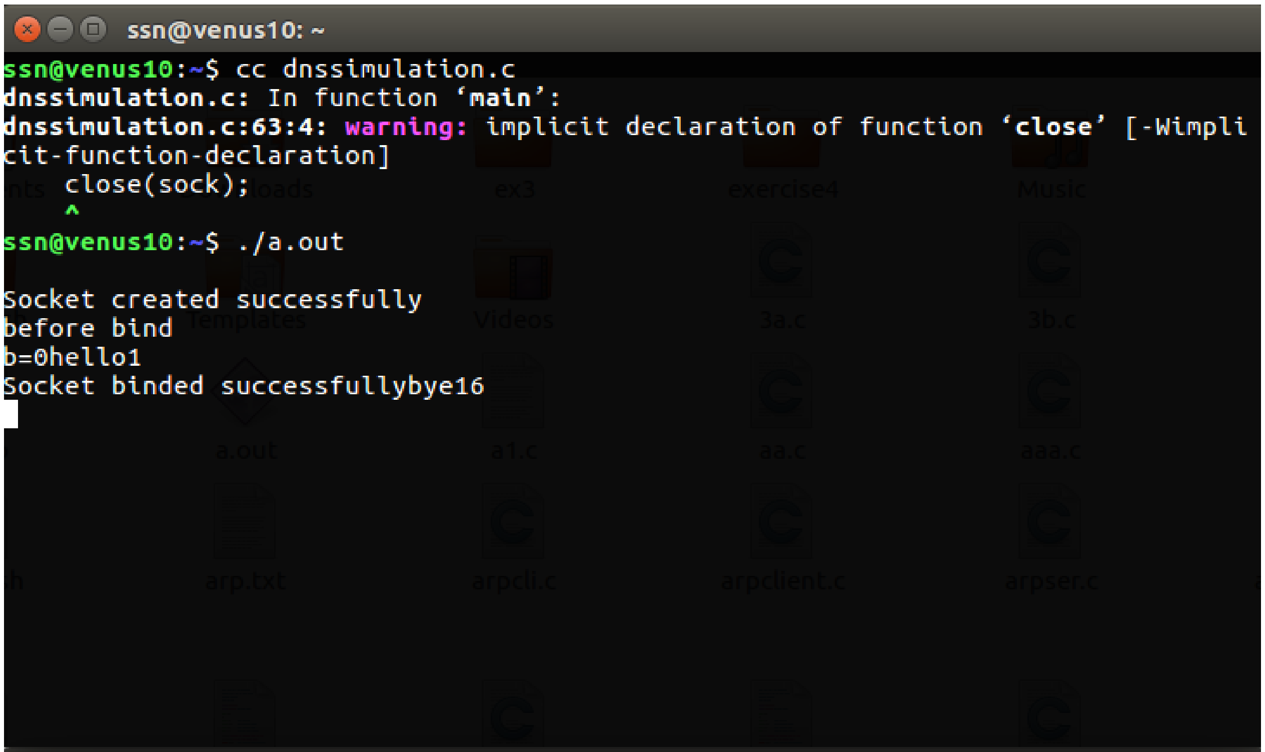
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<netinet/in.h>
#include<sys/socket.h>
#include<sys/types.h>
int main()
{
  FILE *fp;
  char buf[100],msg[100],buf1[100];
  struct sockaddr_in ser,cli;
  int sock,n,r,b;
  if((sock=socket(PF_INET,SOCK_DGRAM,0))<0)
  {
     perror("Error in socket creation");
     return 0;
  }
  printf("\nSocket created successfully");
  bzero(&ser,sizeof(ser));
  ser.sin_family=PF_INET;
  ser.sin_port=42523;
  ser.sin_addr.s_addr=inet_addr("127.0.0.1");
  b=bind(sock,(struct sockaddr *)&ser,sizeof(ser));
  if(b<0)
  {
     perror("\nSocket failed to bind");
     return 0;
  }
  printf("\nSocket binded successfully");
  n=sizeof(cli);
  while(1)
  {
       strcpy(buf1,"");
        fp=fopen("dns.txt","r");
        r=recvfrom(sock,buf,sizeof(buf),0,(struct sockaddr *)&cli,&n);
        while(!feof(fp))
        {
         fscanf(fp,"%s",msg);
         if(strcmp(msg,buf)==0)
          {
             fscanf(fp,"%s",buf1);
            break;
          }
        }
        if(strcmp(buf1,"")==0)
          strcpy(buf1,"Invalid address");
       fclose(fp);
       printf("%s",buf1);
```

```
        sendto(sock,buf1,sizeof(buf1),0,(struct sockaddr *)&cli,n);
        if((strcmp(buf,"exit"))==0)
        {
               printf("sever Exit...\n");
               break;
        }
    }
    close(sock);
    return 0;
}
```

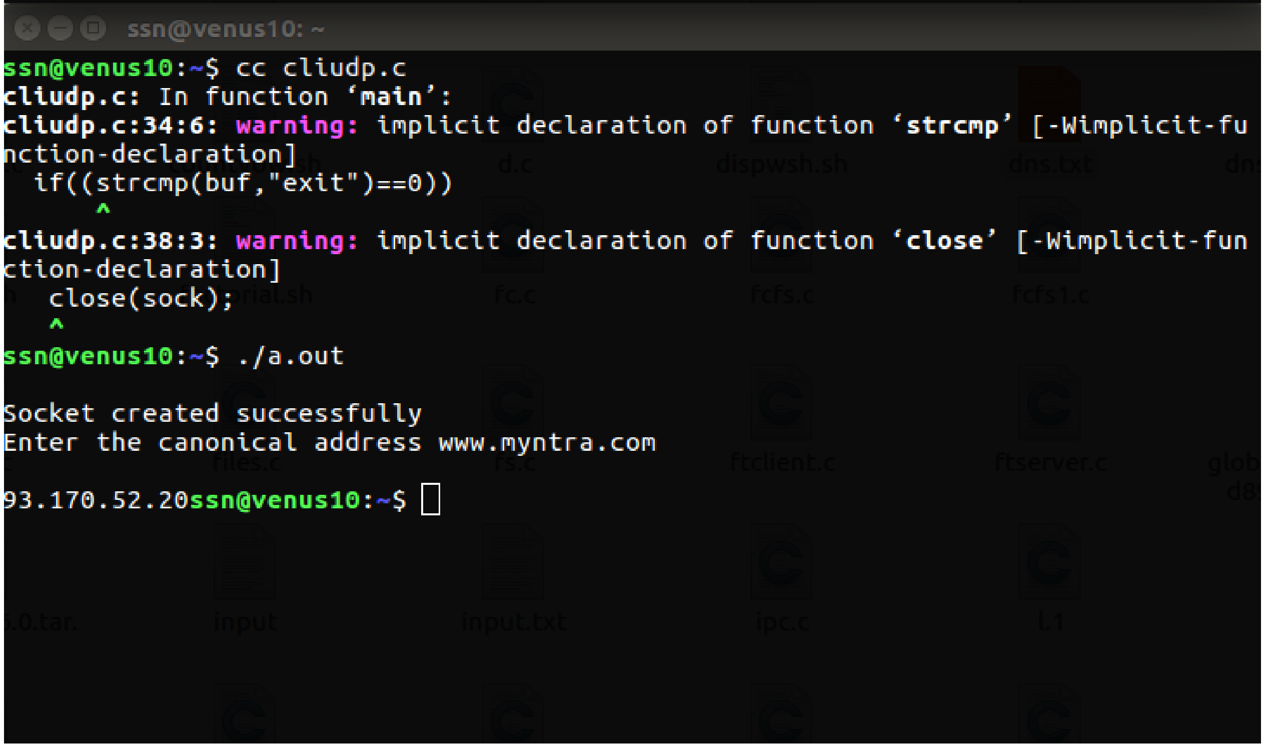
/* client */

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<sys/socket.h>
void main()
{
        struct sockaddr_in ser,cli;
        int sock,n;
        char buf[100],buf1[100];
        if((sock=socket(PF_INET,SOCK_DGRAM,0))<0)
        {
               perror("Error in creating sockets");
               exit(0);
        }
        printf("\nSocket created successfully");
        ser.sin_family=PF_INET;
        ser.sin_port=42523;
        ser.sin_addr.s_addr=inet_addr("127.0.0.1");
        n=sizeof(ser);
        printf("\nEnter the canonical address");
        scanf("%s",buf);
        sendto(sock,buf,sizeof(buf),0,(struct sockaddr *)&ser,n);
        recvfrom(sock,buf1,sizeof(buf1),0,(struct sockaddr *)&ser,&n);
        printf("\n%s",buf1);
        if((strcmp(buf,"exit")==0))
        {
               printf("Client exit");
        }
        close(sock);
}
```

OUTPUT:



```
ssn@venus10: ~
ssn@venus10:~$ cc dnssimulation.c
dnssimulation.c: In function ‘main’:
dnssimulation.c:63:4: warning: implicit declaration of function ‘close’ [-Wimpli
cit-function-declaration]
    close(sock);
    ^
ssn@venus10:~$ ./a.out

Socket created successfully
before bind
b=0hello1
Socket binded successfullybye16
```



```
ssn@venus10: ~
ssn@venus10:~$ cc cliudp.c
cliudp.c: In function ‘main’:
cliudp.c:34:6: warning: implicit declaration of function ‘strcmp’ [-Wimplicit-fu
nction-declaration]
  if((strcmp(buf,"exit")==0))
      ^
cliudp.c:38:3: warning: implicit declaration of function ‘close’ [-Wimplicit-fun
ction-declaration]
   close(sock);
   ^
ssn@venus10:~$ ./a.out

Socket created successfully
Enter the canonical address www.myntra.com

93.170.52.20ssn@venus10:~$ 
```

PROGRAM CODE:

/* server */

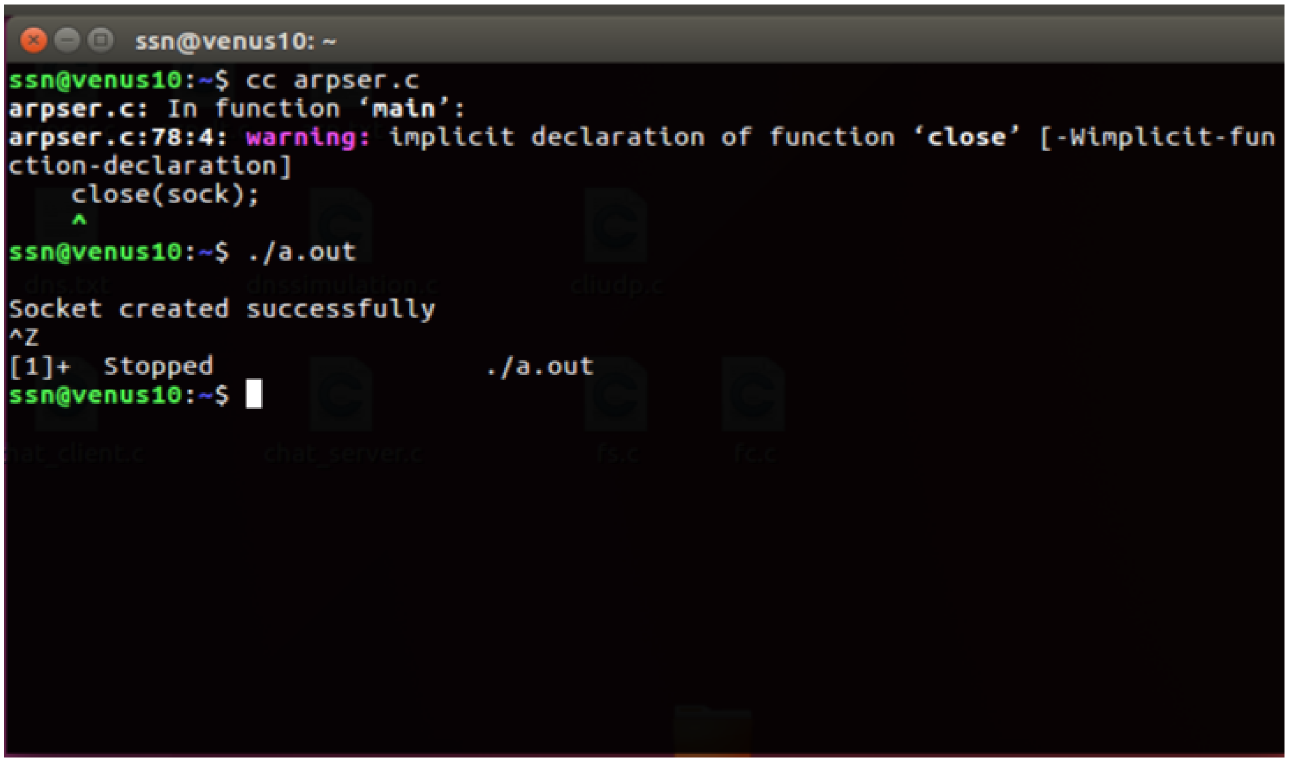
```
#include<stdio.h>
#include<netinet/in.h>
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<sys/socket.h>
void main()
{
  FILE *fp;
   char buf[100],msg[100],buf1[100],choice[5];
  struct sockaddr_in ser,cli;
  int sock,b,status,len,r,n,v;
  int pos,p;
  if((sock=socket(PF_INET,SOCK_DGRAM,0))<0)
  {
     perror("Error in socket creation");
     exit(0);
  }
  printf("\nSocket created successfully");
  bzero(&ser,sizeof(ser));
  ser.sin_family=PF_INET;
  ser.sin_port=42523;
  ser.sin_addr.s_addr=htonl(INADDR_ANY);
   b=bind(sock,(struct sockaddr *)&ser,sizeof(ser));
  if(b<0)
  {
     perror("\nSocket failed to bind");
     exit(0);
  }
  printf("\nSocket binded successfully");
  n=sizeof(cli);
  while(1)
  {
       strcpy(buf1,"");
       fp=fopen("arp.txt","r");
       v=recvfrom(sock,choice,sizeof(choice),0,(struct sockaddr*)&cli,&n);
       r=recvfrom(sock,buf,sizeof(buf),0,(struct sockaddr *)&cli,&n);
       if((strcmp(choice,"a"))==0)
       {
         while(!feof(fp))
         {
         fscanf(fp,"%s",msg);
         if(strcmp(msg,buf)==0)
          {
             fscanf(fp,"%s",buf1);
            break;
          }
         }
```

```
          if(strcmp(buf1,"")==0)
           strcpy(buf1,"Invalid address");
        }
        else
        {
         while(!feof(fp))
         {
           pos=ftell(fp);
           fscanf(fp,"%s",msg);
           if(strcmp(msg,buf)==0)
           {
              p=pos-13;
             fseek(fp,p,SEEK_SET);
             fscanf(fp,"%s",buf1);
             break;
           }
         }
         if(strcmp(buf1,"")==0)
            strcpy(buf1,"Invalid address");
        }
        fclose(fp);
        printf("%s",buf1);
        sendto(sock,buf1,sizeof(buf1),0,(struct sockaddr *)&cli,n);
        if((strcmp(buf,"exit"))==0)
        {
               printf("sever Exit...\n");
               break;
        }
    }
    close(sock);
}
```

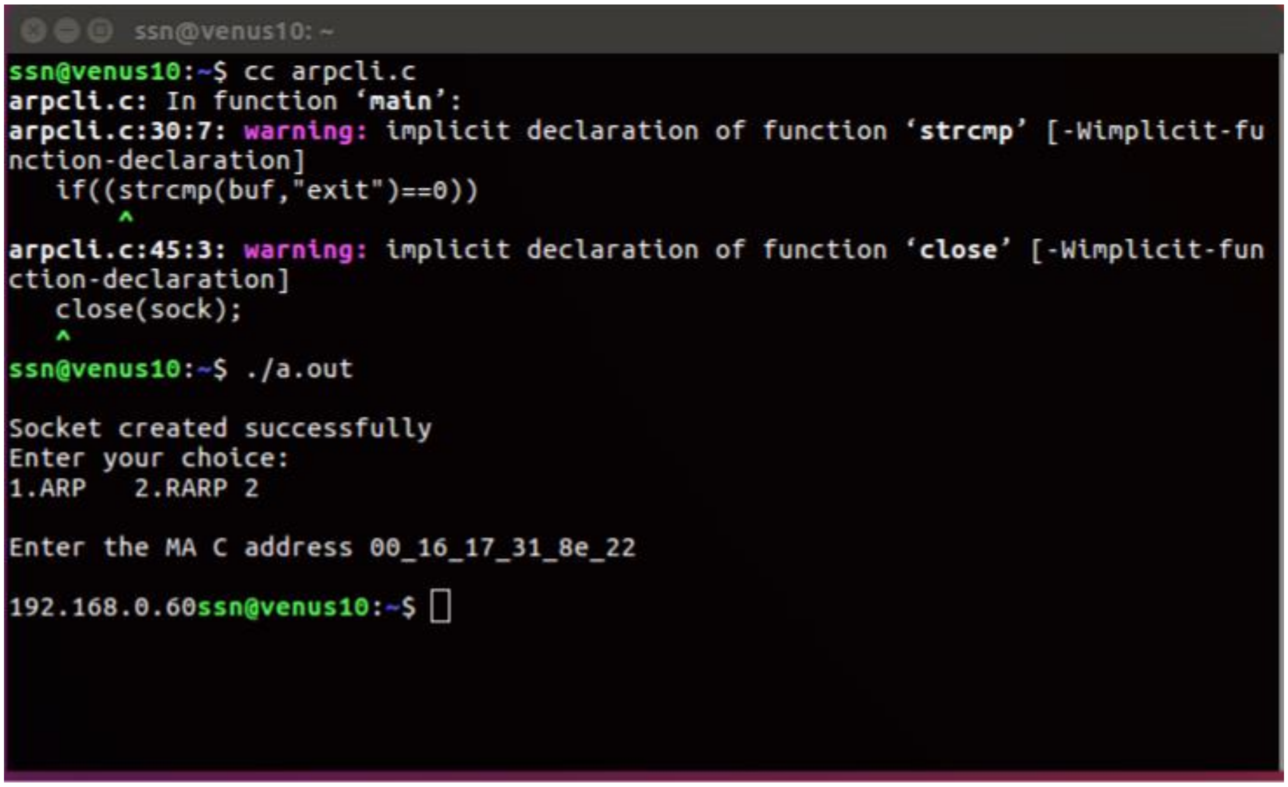
/* client */

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<sys/socket.h>
void main()
{
        struct sockaddr_in ser,cli;
        int sock,n,choice,r;
        char buf[100],buf1[100],c1[]="a",c2[]="b";
        if((sock=socket(PF_INET,SOCK_DGRAM,0))<0)
        {
               perror("Error in creating sockets");
               exit(0);
        }
        printf("\nSocket created successfully");
        ser.sin_family=PF_INET;
        ser.sin_port=42523;
        ser.sin_addr.s_addr=htonl(INADDR_ANY);
        n=sizeof(ser);
         printf("\nEnter your choice:\n1.ARP\t2.RARP");
        scanf("%d",&choice);
         if(choice==1){
         printf("\nEnter the IP address");
         scanf("%s",buf);
         sendto(sock,c1,sizeof(c1),0,(struct sockaddr *)&ser,n);
         sendto(sock,buf,sizeof(buf),0,(struct sockaddr *)&ser,n);
         r=recvfrom(sock,buf1,sizeof(buf1),0,(struct sockaddr *)&ser,&n);
         printf("\n%s",buf1);
         if((strcmp(buf,"exit")==0))
         {
                printf("Client exit");
         }
         }
        else{
         printf("\nEnter the MAC address");
         scanf("%s",buf);
         sendto(sock,c2,sizeof(c2),0,(struct sockaddr *)&ser,n);
         sendto(sock,buf,sizeof(buf),0,(struct sockaddr *)&ser,n);
         r=recvfrom(sock,buf1,sizeof(buf1),0,(struct sockaddr *)&ser,&n);
         printf("\n%s",buf1);
         if((strcmp(buf,"exit")==0))
                printf("\n Client exit");
        }
        close(sock);
}
```

OUTPUT:



```
ssn@venus10: ~
ssn@venus10:~$ cc arpser.c
arpser.c: In function ‘main’:
arpser.c:78:4: warning: implicit declaration of function ‘close’ [-Wimplicit-fun
ction-declaration]
    close(sock);
    ^
ssn@venus10:~$ ./a.out

Socket created successfully
^Z
[1]+  Stopped                 ./a.out
ssn@venus10:~$ 
```



```
ssn@venus10: ~
ssn@venus10:~$ cc arpcli.c
arpcli.c: In function 'main':
arpcli.c:30:7: warning: implicit declaration of function 'strcmp' [-Wimplicit-fu
nction-declaration]
   if((strcmp(buf,"exit")==0))
       ^
arpcli.c:45:3: warning: implicit declaration of function 'close' [-Wimplicit-fun
ction-declaration]
   close(sock);
   ^
ssn@venus10:~$ ./a.out

Socket created successfully
Enter your choice:
1.ARP   2.RARP 2

Enter the MA C address 00_16_17_31_8e_22

192.168.0.60ssn@venus10:~$ 
```

PROGRAM CODE:

/* server */

```
#include<stdio.h>
#include<netinet/in.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#define MAX 80
#define PORT 43454
#define SA struct sockaddr
void func(int sockfd)
{
	char buff[MAX];
	int n;
	for(;;)
	{
		bzero(buff,MAX);
		read(sockfd,buff,sizeof(buff));
		printf("From client: %s\t To client : ",buff);
		bzero(buff,MAX);
		n=0;
		while((buff[n++]=getchar())!='\n');
		write(sockfd,buff,sizeof(buff));
		if(strncmp("exit",buff,4)==0)
		{
			printf("Server Exit...\n");
			break;
		}
	}
}
int main()
{
	int sockfd,connfd,len;
	struct sockaddr_in servaddr,cli;
	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));
	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)
```

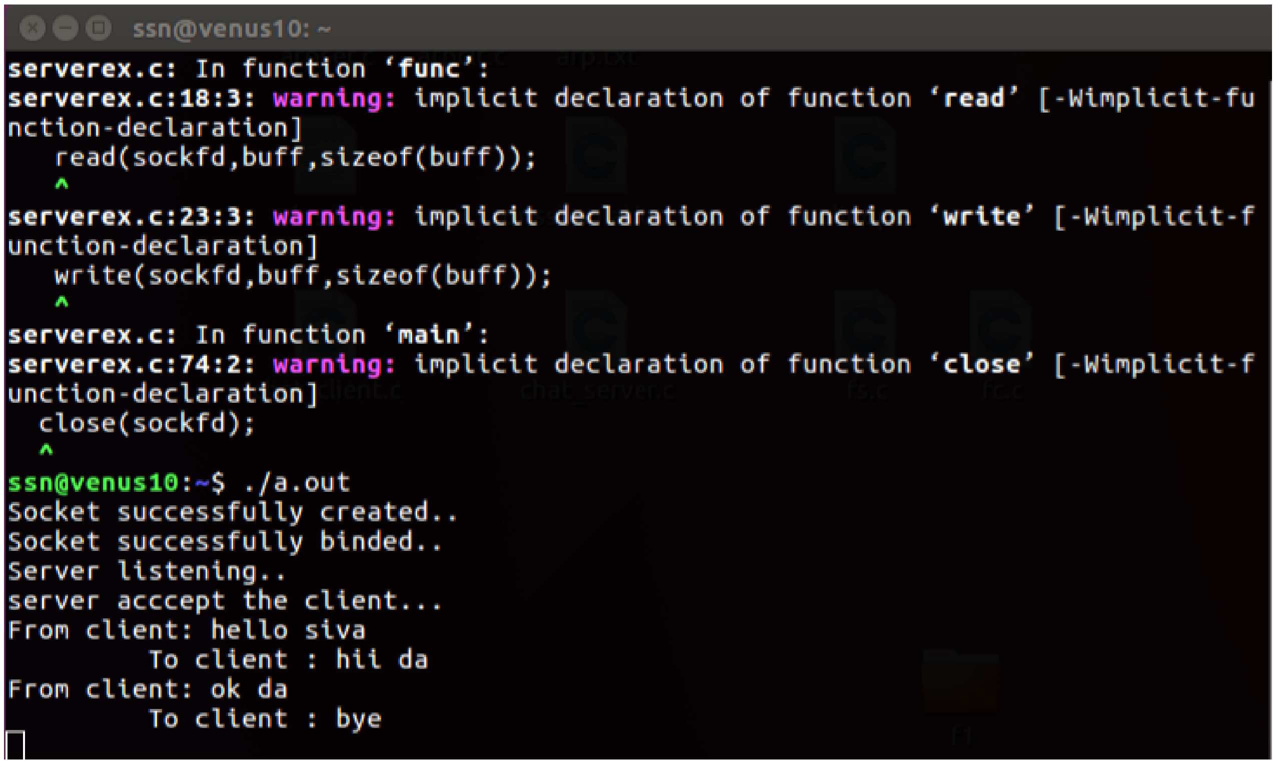
```
        {
               printf("socket bind failed...\n");
               exit(0);
        }
        else
               printf("Socket successfully binded..\n");

        if((listen(sockfd,5))!=0)
        {
               printf("Listen failed...\n");
               exit(0);
        }
        else
               printf("Server listening..\n");

        len=sizeof(cli);
        connfd=accept(sockfd,(SA *)&cli,&len);
        if(connfd<0)
        {
               printf("server acccept failed...\n");
               exit(0);
        }
        else
               printf("server acccept the client...\n");
        func(connfd);
        close(sockfd);
}
```

/* client */

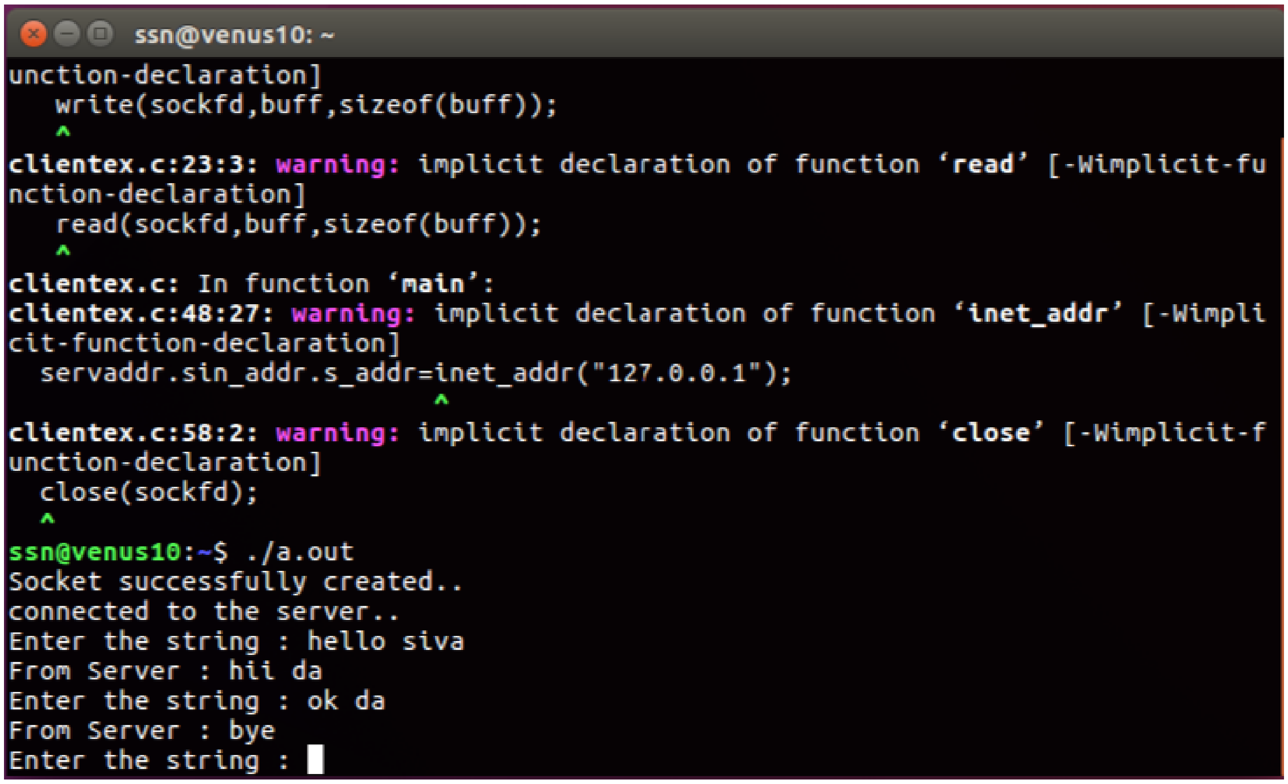
```
#include<stdio.h>
#include<netinet/in.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netdb.h>
#include<string.h>
#include<stdlib.h>
#define MAX 80
#define PORT 43454
#define SA struct sockaddr
void func(int sockfd)
{
        char buff[MAX];
        int n;
        for(;;)
        {
               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;
        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));
        servaddr.sin_family=AF_INET;
        servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
        servaddr.sin_port=htons(PORT);
        if(connect(sockfd,(SA *)&servaddr,sizeof(servaddr))!=0)
        {
                printf("connection with the server failed...\n");
                exit(0);
        }
        else
                printf("connected to the server..\n");
        func(sockfd);
        close(sockfd);
}
```

OUTPUT:



```
ssn@venus10: ~
serverex.c: In function ‘func’:
serverex.c:18:3: warning: implicit declaration of function ‘read’ [-Wimplicit-fu
nction-declaration]
   read(sockfd,buff,sizeof(buff));
   ^
serverex.c:23:3: warning: implicit declaration of function ‘write’ [-Wimplicit-f
unction-declaration]
   write(sockfd,buff,sizeof(buff));
   ^
serverex.c: In function ‘main’:
serverex.c:74:2: warning: implicit declaration of function ‘close’ [-Wimplicit-f
unction-declaration]
  close(sockfd);
  ^
ssn@venus10:~$ ./a.out
Socket successfully created..
Socket successfully binded..
Server listening..
server acccept the client...
From client: hello siva
         To client : hii da
From client: ok da
         To client : bye
```



```
ssn@venus10: ~
unction-declaration]
   write(sockfd,buff,sizeof(buff));
   ^
clientex.c:23:3: warning: implicit declaration of function ‘read’ [-Wimplicit-fu
nction-declaration]
   read(sockfd,buff,sizeof(buff));
   ^
clientex.c: In function ‘main’:
clientex.c:48:27: warning: implicit declaration of function ‘inet_addr’ [-Wimpli
cit-function-declaration]
  servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
                           ^
clientex.c:58:2: warning: implicit declaration of function ‘close’ [-Wimplicit-f
unction-declaration]
  close(sockfd);
  ^
ssn@venus10:~$ ./a.out
Socket successfully created..
connected to the server..
Enter the string : hello siva
From Server : hii da
Enter the string : ok da
From Server : bye
Enter the string :
```