Computer Networks

Lab Manual

Submitted in the partial fulfillment
of the requirements for the award of Degree of

Bachelor of Engineering

in

Computer Science and Engineering

By

NAME: _____

ROLL NO: _____

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY COLLEGE OF ENGINEERING (A)
Osmania University, Hyderabad – 500 007
2023-2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY COLLEGE OF ENGINEERING (A)
Osmania University, Hyderabad – 500 007

CERTIFICATE

This is to certify that _____ bearing

Roll no:_____Studying B.E. VI Semester has

Successfully completed "**Computer Networks lab**" for the academic year 2023-24.

Internal Examiner **External Examiner**

Dr. L K Suresh Kumar

Index

1. Write a program for iterative server.

SERVER

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<string.h>
short portno;

int main(int argc,char *argv[])
{
	int sockmain,sockcli,i;
	int child;
	char buffer[512];
	struct sockaddr_in servaddr,cliaddr;
	if(argc!=2)
	{
		printf("Usage: server <portno>\n");
		exit(1);
	}
	if((sockmain=socket(AF_INET,SOCK_STREAM,0))<0)
	{
		perror("socket");
		exit(1);
	}
	servaddr.sin_family=AF_INET;
	servaddr.sin_port=htons(atoi(argv[1]));
```

```
servaddr.sin_addr.s_addr=htonl(0L);
if(bind(sockmain,(struct sockaddr *)&servaddr,sizeof(servaddr))<0)
{
	perror("Bind");
	exit(1);
}
if(listen(sockmain,5)<0)
{
	perror("LISTEN");
	exit(1);
}
for(;;)
{
	//char buffer[512];
	i=sizeof(cliaddr);
	if((sockcli=accept(sockmain,(struct sockaddr *)&cliaddr,&i))<0)
	{
		perror("ACCEPT");
		exit(1);
	}
	while((i=read(sockcli,buffer,sizeof( buffer)))!=0)
	{
		if(i<0)
		{
			perror("REad socket");
			exit(1);
		}
		if(write(sockcli,buffer,i)!=i)
		{
			perror("write socket");
```

```
                exit(1);
            }
            int a,b,temp;
            int l=strlen(buffer);
            for(a=0,b=l-1;a<b;a++,b--)
            {
                temp=buffer[a];
                buffer[a]=buffer[b];
                buffer[b]=temp;
            }
            write(1,buffer,i);          //or write(1,buffer,sizeof(buffer))
        }
    }
    return 0;
}
```

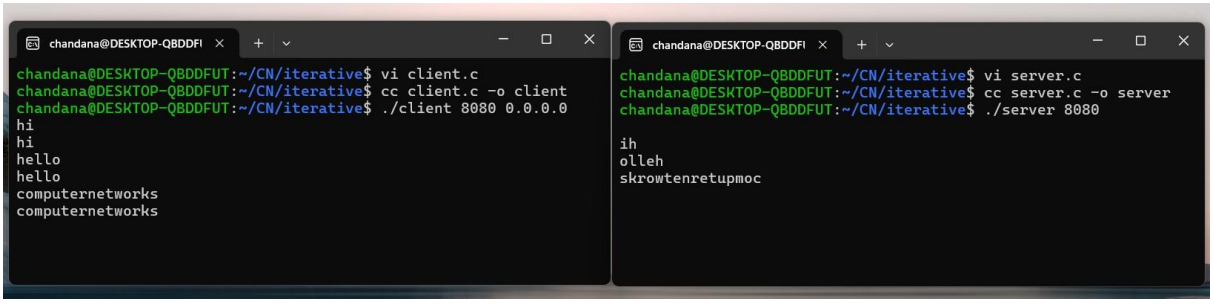
CLIENT

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
short portno;
int main(int argc,char *argv[])
{
  int sd,i;
  int rf_stdin,rf_sock;
  char buffer[512];
```

```
  struct sockaddr_in servaddr,cliaddr;
  if(argc!=3)
  {
    printf("client:usage <portno>");
    exit(1);
  }
  if((sd=socket(AF_INET,SOCK_STREAM,0))<0)
  {
    perror("client:socket");
    exit(1);
   }
   servaddr.sin_family=AF_INET;
   servaddr.sin_port=htons(atoi(argv[1]));
   servaddr.sin_addr.s_addr=inet_addr(argv[2]);
   if(connect(sd,(struct sockaddr *)&servaddr,sizeof(servaddr))<0)
 {
   perror("CLIENT:connect");
   exit(1);
 }
 for(;;)
{
  rf_stdin=read(0,buffer,sizeof(buffer));
  if(rf_stdin<0)
{
   perror("read stdin");
   exit(1);
 }
 if(write(sd,buffer,rf_stdin)!=rf_stdin)
{
  perror("write sock");
```

```
}
rf_sock=read(sd,buffer,sizeof(buffer));
if(rf_sock<0)
 {
    perror("read sock");
    exit(1);
  }
if(write(1,buffer,rf_sock)!=rf_sock)
{
  perror("write stdout");
  exit(1);
 }
}
exit(0);
return 0;
}
```

OUTPUT



2.Write a program for concurrent server.

SERVER

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/socket.h>
```

```
#include<netinet/in.h>
#include<arpa/inet.h>
#include<string.h>
short portno;

int main(int argc,char *argv[])
{
	int sockmain,sockcli,i;
	int childpid;
	char buffer[512];
	struct sockaddr_in servaddr,cliaddr;
	if(argc!=2)
	{
		printf("Usage: server <portno>\n");
		exit(1);
	}
	if((sockmain=socket(AF_INET,SOCK_STREAM,0))<0)
	{
		perror("socket");
		exit(1);
	}
	servaddr.sin_family=AF_INET;
	servaddr.sin_port=htons(atoi(argv[1]));
	servaddr.sin_addr.s_addr=htonl(0L);
	if(bind(sockmain,(struct sockaddr *)&servaddr,sizeof(servaddr))<0)
	{
		perror("Bind");
		exit(1);
	}
	if(listen(sockmain,5)<0)
```

```
{
        perror("LISTEN");
        exit(1);
}
for(;;)
{
        i=sizeof(cliaddr);
        if((sockcli=accept(sockmain,(struct sockaddr *)&cliaddr,&i))<0)
        {
                perror("ACCEPT");
                exit(1);
        }
        if((childpid=fork())==0)
        {
                close(sockmain);
                while((i=read(sockcli,buffer,sizeof( buffer)))!=0)
                {
                        if(i<0)
                        {
                                perror("REad socket");
                                exit(1);
                        }
                        if(write(sockcli,buffer,i)!=i)
                        {
                                perror("write socket");
                                exit(1);
                        }
                        int a,b,temp;
                        int l=strlen(buffer);
                        for(a=0,b=l-1;a<b;a++,b--)
```

```
                    {
                        temp=buffer[a];
                        buffer[a]=buffer[b];
                        buffer[b]=temp;
                    }
                    write(1,buffer,i);          //or write(1,buffer,sizeof(buffer))
                }
            }
            close(sockcli);
        }
        return 0;
}
```

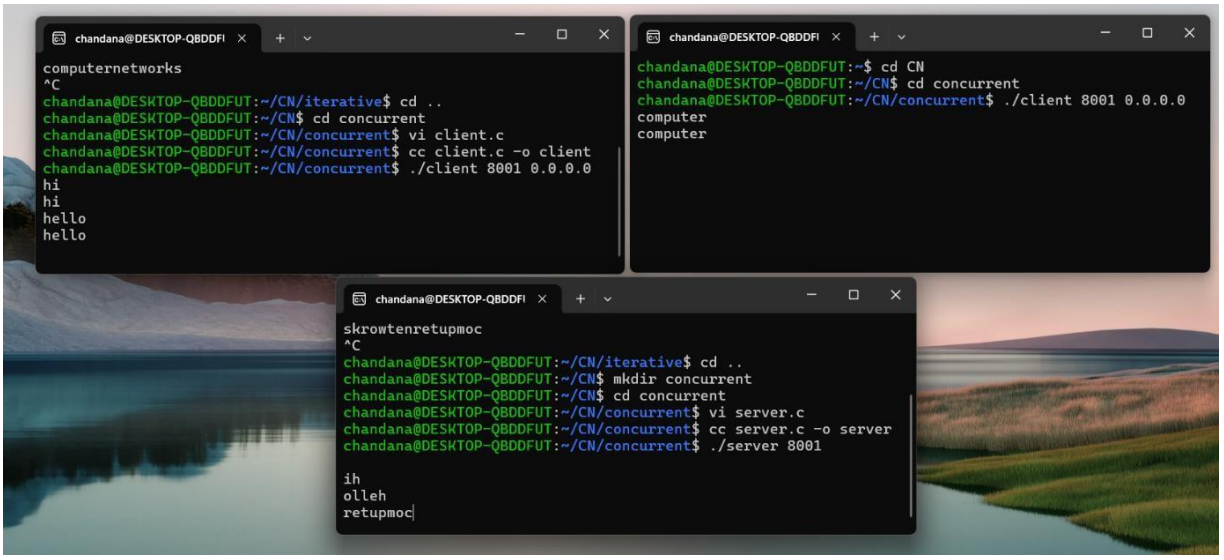
CLIENT

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
short portno;
int main(int argc,char *argv[])
{
  int sd,i;
  int rf_stdin,rf_sock;
  char buffer[512];
  struct sockaddr_in servaddr,cliaddr;
  if(argc!=3)
  {
    printf("client:usage <portno>");
```

```
    exit(1);
  }
  if((sd=socket(AF_INET,SOCK_STREAM,0))<0)
  {
    perror("client:socket");
    exit(1);
   }
   servaddr.sin_family=AF_INET;
   servaddr.sin_port=htons(atoi(argv[1]));
   servaddr.sin_addr.s_addr=inet_addr(argv[2]);
   if(connect(sd,(struct sockaddr *)&servaddr,sizeof(servaddr))<0)
 {
   perror("CLIENT:connect");
   exit(1);
 }
 for(;;)
{
  rf_stdin=read(0,buffer,sizeof(buffer));
  if(rf_stdin<0)
{
   perror("read stdin");
   exit(1);
 }
 if(write(sd,buffer,rf_stdin)!=rf_stdin)
{
  perror("write sock");
}
rf_sock=read(sd,buffer,sizeof(buffer));
if(rf_sock<0)
 {
```

```
    perror("read sock");
    exit(1);
  }
if(write(1,buffer,rf_sock)!=rf_sock)
{
  perror("write stdout");
  exit(1);
 }
}
exit(0);
return 0;
}
```

OUTPUT



3. Write an Echo server connection less program.

SERVER

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
```

```
#include<arpa/inet.h>
#include<string.h>
int main(int argc,char *argv[])
{
	int sockmain,i;
	char buffer[512];
	struct sockaddr_in servaddr,cliaddr;
	int cliaddrlength,msglength,no_of_bytes;
	int portno;
	if(argc != 2)
	{
		printf("USAGE : Server <portno>\n");
		exit(1);
	}
	if((sockmain = socket(AF_INET,SOCK_DGRAM,0)) < 0)
	{
		perror("SERVER : socket error\n");
		exit(1);
	}
	servaddr.sin_family = AF_INET;
	servaddr.sin_port = htons(atoi(argv[1]));
	servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
	if(bind(sockmain,(struct sockaddr *)&servaddr,sizeof(servaddr)) < 0)
	{
		perror("SERVER : bind error\n");
		exit(1);
	}
	cliaddrlength = sizeof(cliaddr);
	msglength = recvfrom(sockmain,(char *)buffer,sizeof(buffer),0,(struct sockaddr
*)&cliaddr,&cliaddrlength);
```

```
	no_of_bytes = sendto(sockmain,buffer,sizeof(buffer),0,(structsockaddr
*)&cliaddr,cliaddrlength);
	if(msglength < 0)
	{
		perror("SERVER : recvfrom error\n");
		exit(1);
	}
	printf("SERVER :\n");
	printf(" ------- \n");
	printf("Got data from %s\n",inet_ntoa(cliaddr.sin_addr));
	return 0;
}
```

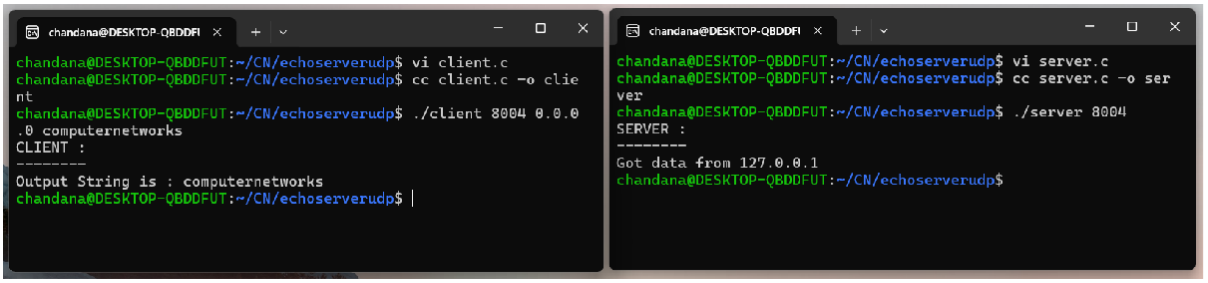
CLIENT

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<netdb.h>
#include<string.h>
int main(int argc,char *argv[])
{
	int sock,i;
	char buffer[512];
	struct sockaddr_in servaddr,cliaddr;
	int servaddrlength,msglength,no_of_bytes;
	if(argc != 4)
	{
		printf("USAGE : Client <portno> <hostname> <message>\n");
		exit(1);
```

```
}
if((sock = socket(AF_INET,SOCK_DGRAM,0)) < 0)
{
        perror("CLIENT : socket error\n");
        exit(1);
}
cliaddr.sin_family = AF_INET;
cliaddr.sin_port = htons(0);
cliaddr.sin_addr.s_addr = htonl(0L);
if(bind(sock,(struct sockaddr *)&cliaddr,sizeof(cliaddr)) < 0)
{
        perror("CLIENT : bind error\n");
        exit(1);
}
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(atoi(argv[1]));
servaddr.sin_addr.s_addr = inet_addr(argv[2]);
no_of_bytes = sendto(sock,argv[3],strlen(argv[3]),0,(struct
sockaddr*)&servaddr,sizeof(servaddr));
servaddrlength = sizeof(servaddr);
msglength= recvfrom(sock,buffer,sizeof(buffer),0,(struct
sockaddr*)&servaddr,&servaddrlength);
if(no_of_bytes < 0)
{
        perror("CLIENT : recvfrom error\n");
        exit(1);
}
printf("CLIENT :\n");
printf(" ------- \n");
printf("Output String is : %s\n",buffer);
return 0;
```

}

OUTPUT



4. Write a Listener talker program.

LISTENER

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>

short portno;
int main(int argc,char *argv[])
{
        int sockmain,i;
        char buffer[512];
        struct sockaddr_in servaddr,clinaddr;
        int addrlength,msglength;

        if(argc!=2)
        {
                printf("USAGE:Listener <portno>");
                exit(1);
        }
```

```
if((sockmain = socket(AF_INET,SOCK_DGRAM,0))<0)
{
        perror("server socket");
        exit(1);
}

portno=atoi(argv[1]);
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(portno);
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);

i=sizeof(servaddr);

if(bind(sockmain, (struct sockaddr*)&servaddr ,i)<0)
{
        perror("bind");
        exit(1);
}

addrlength=sizeof(clinaddr);
msglength=recvfrom(sockmain,(char*)buffer,sizeof(buffer),0,(struct
sockaddr*)&clinaddr,(socklen_t*)&(addrlength));

if(msglength<0)
{
        perror("receive from error\n");
        exit(1);
}

printf("\t\tLISTENER\n");
```

```
        printf("\t\t---------\n");
        printf("Got a packet from %s\n",inet_ntoa(clinaddr.sin_addr));
        printf("Packet is %d bytes long\n",msglength);
        printf("Packet contains %s\n",buffer);

        return 0;
}
```

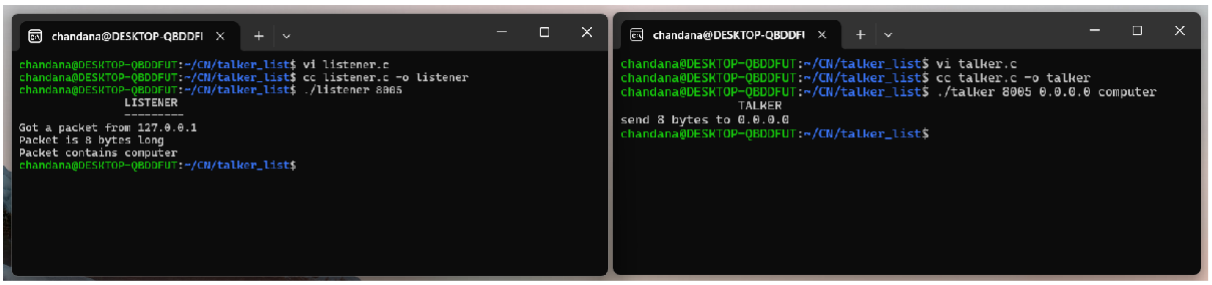
TALKER

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>

short portno;
int main(int argc,char *argv[])
{
        int sock,i;
        char buffer[512];
        int addrlength,msglength,msgrecv;
        int no_of_bytes;
        struct sockaddr_in servaddr,clinaddr;

        if(argc!=4)
        {
                printf("Usage: client <portno> <hostname> <message>\n");
                exit(1);
        }
```

```
if((sock=socket(AF_INET,SOCK_DGRAM,0))<0)
{
        perror("CLIENT: socket");
        exit(1);
}

clinaddr.sin_family=AF_INET;
clinaddr.sin_port=htons(0);
clinaddr.sin_addr.s_addr=htonl(INADDR_ANY);

if(bind(sock,(struct sockaddr*)&clinaddr,sizeof(clinaddr))<0)
{
        perror("CLIENT:bind");
        exit(1);
}

servaddr.sin_family=AF_INET;
portno=atoi(argv[1]);
servaddr.sin_port=htons(portno);
servaddr.sin_addr.s_addr=inet_addr(argv[2]);

no_of_bytes = sendto(sock,argv[3],strlen(argv[3]),0,(struct
sockaddr*)&servaddr,sizeof(servaddr));

if(no_of_bytes<0)
{
        perror("\nCLIENT: send to error\n");
        exit(1);
}
```

```
        printf("\t\tTALKER\n");
        printf("send %d bytes to %s\n",no_of_bytes,inet_ntoa(servaddr.sin_addr));

        return 0;
}
```

OUTPUT



5. Write a program to display time using UDP.

SERVER

```
#include<stdio.h>
#include<stdlib.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<unistd.h>
#include<string.h>
#include<time.h>
int main(int argc,char *argv[])
{
        int sd;
        char buffer[512];
        struct sockaddr_in servaddr,cliaddr;  //in header netinet/in.h
        int cliaddrlen,msglen,no_of_bytes;
```

```
time_t t,curtime;
char *s;

if(argc!=2)
{
	printf("USAGE: SERVER <portno>\n");
	exit(1);
}

if((sd=socket(AF_INET,SOCK_DGRAM,0))<0)
{
	perror("SERVER: socket error\n");
	exit(1);
}

servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(atoi(argv[1]));
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);

if(bind(sd,(struct sockaddr*)&servaddr,sizeof(servaddr))<0)
{
	perror("SERVER: bind error\n");
	exit(1);
}

cliaddrlen=sizeof(cliaddr);
msglen=recvfrom(sd,(char *)buffer,sizeof(buffer),0,(struct
sockaddr*)&cliaddr,&cliaddrlen);
if(msglen<0)
{
	perror("SERVER: recvfrom error\n");
```

```
        exit(1);
    }

    time(&curtime);     //retrives current calendar time and stores it in curtime(ex:
00:00:00utc,jan 1,1970)
    s=ctime(&curtime);  //ctime converts to ascii string(ex: wed sep 15 14:35:20
2021)

    no_of_bytes=sendto(sd,(char*)s,sizeof(s),0,(struct
sockaddr*)&cliaddr,cliaddrlen);
    printf("%s\n",s);

    if(no_of_bytes<0)
    {
        perror("SERVER: sendto error\n");
        exit(1);
    }

    printf("SERVER\n");
    printf(" ------------- \n");
    printf("Requesting date and time from %s\n",inet_ntoa(cliaddr.sin_addr));
    printf("Sent response\n");
    return 0;
}
```

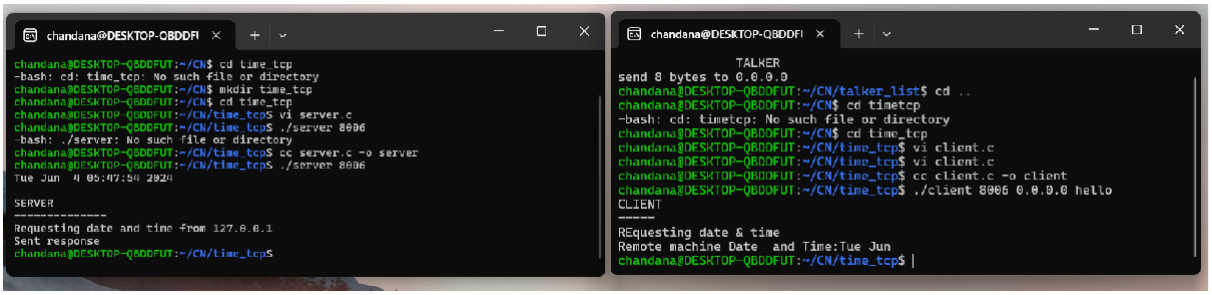
CLIENT

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
```

```
#include<arpa/inet.h>
#include<string.h>
#include<time.h>
#include<netdb.h>
int main(int argc,char *argv[])
{
  int sk;
  char buffer[512];
  struct sockaddr_in servaddr,cliaddr;
  int no_of_bytes;
  int servaddrlen,msglen;
  if(argc!=4)
  {
    printf("client:client socket error");
    exit(1);
  }
  if((sk=socket(AF_INET,SOCK_DGRAM,0))<0)
  {
    printf("client:client socket error");
    exit(1);
  }
   cliaddr.sin_family=AF_INET;
   cliaddr.sin_port=htons(0);
   cliaddr.sin_addr.s_addr=htonl(0L);
   if(bind(sk,(struct sockaddr *)&cliaddr,sizeof(cliaddr))<0)
   {
     perror("client:bind error");
     exit(1);
   }
    servaddr.sin_family=AF_INET;
```

```
  servaddr.sin_port=htons(atoi(argv[1]));
  servaddr.sin_addr.s_addr=inet_addr(argv[2]);
  no_of_bytes=sendto(sk,argv[3],strlen(argv[3]),0,(struct sockaddr
*)&servaddr,sizeof(servaddr));
  if(no_of_bytes<0)
  {
    perror("client:sendto error");
    exit(1);
  }
  servaddrlen=sizeof(servaddr);
  msglen=recvfrom(sk,buffer,sizeof(buffer),0,(struct sockaddr
*)&servaddr,&servaddrlen);
  if(msglen<0)
  {
    perror("client:recvfrom");
    exit(1);
  }
  printf("CLIENT\n");
  printf(" ---- \n");
  printf("REquesting date & time \n");
  printf("Remote machine Date  and Time:%s\n",buffer);
  return 0;
 }
```

OUTPUT



6. Write a program to display time using TCP.

SERVER

```
#include<stdio.h>
#include<stdlib.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<unistd.h>
#include<string.h>
#include<time.h>
int main(int argc,char *argv[])
{
	int sd,nsd;
	char buffer[512];
	struct sockaddr_in servaddr,cliaddr;  //in header netinet/in.h
	int cliaddrlen,msglen,no_of_bytes;
	time_t t,curtime;
	char *s;

	if(argc!=2)
	{
		printf("USAGE: SERVER <portno>\n");
		exit(1);
	}

	if((sd=socket(AF_INET,SOCK_STREAM,0))<0)
	{
		perror("SERVER: socket error\n");
		exit(1);
```

```
}

servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(atoi(argv[1]));
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);

if(bind(sd,(struct sockaddr*)&servaddr,sizeof(servaddr))<0)
{
	perror("SERVER: bind error\n");
	exit(1);
}

if(listen(sd,1)<0)
{
	perror("SERVER: listen error\n");
	exit(1);
}

cliaddrlen=sizeof(cliaddr);
if((nsd=accept(sd,(struct sockaddr*)&cliaddr,&cliaddrlen))<0)
{
	perror("SERVER: accept error\n");
	exit(1);
}

msglen=read(nsd,buffer,sizeof(buffer));
//msglen=recvfrom(sd,(char *)buffer,sizeof(buffer),0,(struct
sockaddr*)&cliaddr,&cliaddrlen);
if(msglen<0)
{
	perror("SERVER: read error\n");
```

```
        exit(1);
    }

    time(&curtime);     //retrives current calendar time and stores it in curtime(ex:
00:00:00utc,jan 1,1970)
    s=ctime(&curtime);  //ctime converts to ascii string(ex: wed sep 15 14:35:20
2021)

    //no_of_bytes=sendto(sd,(char*)s,sizeof(s),0,(struct
sockaddr*)&cliaddr,cliaddrlen);
    //printf("%s\n",s);

    if(write(nsd,s,strlen(s))!=strlen(s))
    {
        perror("SERVER: write error\n");
        exit(1);
    }

    printf("SERVER\n");
    printf("------------- \n");
    printf("Requesting date and time from %s\n",inet_ntoa(cliaddr.sin_addr));
    printf("Sent response\n");
    return 0;
}
```

CLIENT

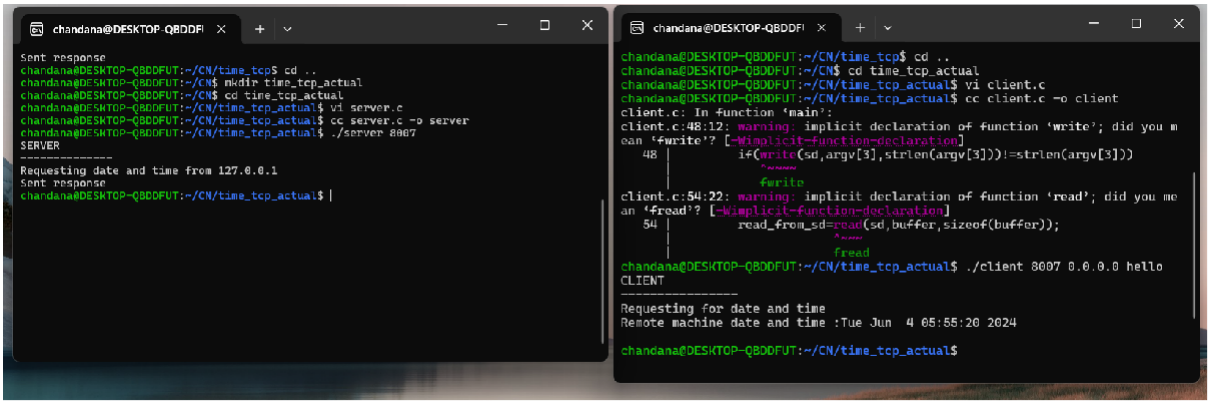
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
```

```
#include<arpa/inet.h>
#include<netdb.h>
int main(int argc,char *argv[])
{
	int sd,i;
	char buffer[512];
	struct sockaddr_in servaddr,cliaddr;
	int read_from_stdin,read_from_sd;

	if(argc!=4)
	{
		printf("USAGE: client <portno><hostname><message>\n");
		exit(1);
	}

	if((sd=socket(AF_INET,SOCK_STREAM,0))<0)
	{
		perror("CLIENT: socket error\n");
		exit(1);
	}

	cliaddr.sin_family=AF_INET;
	cliaddr.sin_port=htons(0);
	cliaddr.sin_addr.s_addr=htonl(0L);

	if(bind(sd,(struct sockaddr*)&cliaddr,sizeof(cliaddr))<0)
	{
		perror("CLIENT: bind error\n");
		exit(1);
	}
```

```
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(atoi(argv[1]));
servaddr.sin_addr.s_addr=inet_addr(argv[2]);

if(connect(sd,(struct sockaddr*)&servaddr,sizeof(servaddr))<0)
{
        perror("CLIENT: connect error\n");
        exit(1);
}

if(write(sd,argv[3],strlen(argv[3]))!=strlen(argv[3]))
{
        perror("CLIENT: write sd error\n");
        exit(1);
}

read_from_sd=read(sd,buffer,sizeof(buffer));
if(read_from_sd<0)
{
        perror("CLIENT: read sd error\n");
        exit(1);
}


printf("CLIENT\n");
printf("---------\n");
printf("Requesting for date and time\n");
printf("Remote machine date and time :%s\n",buffer);
```

```
        return 0;
}
```

OUTPUT

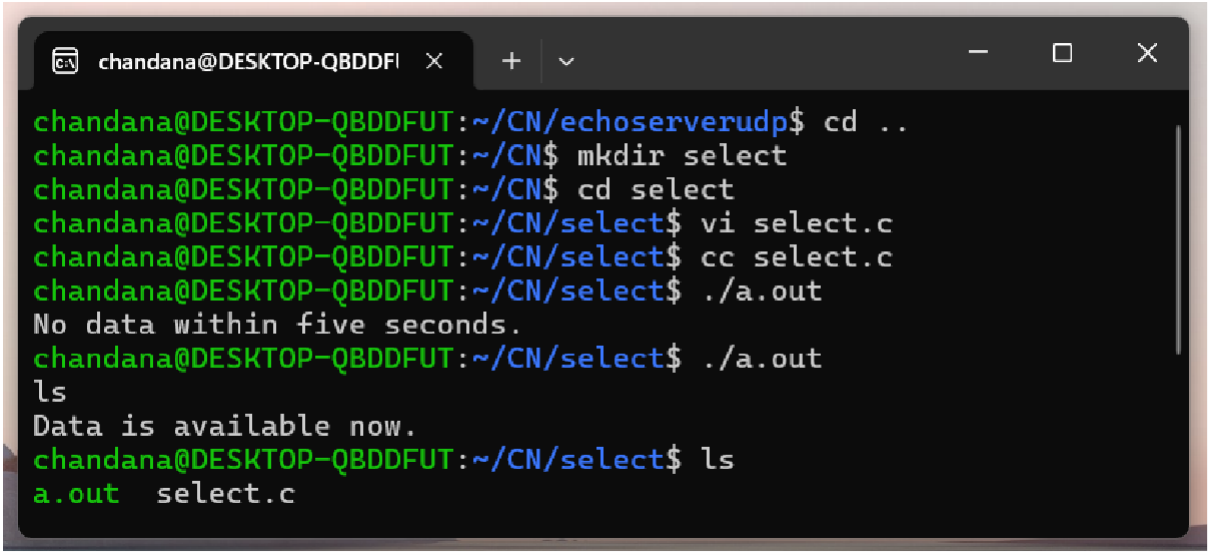


7. Write a program using select system call.

```
#include <stdio.h>
#include <sys/time.h>
#include <sys/types.h>
#include <unistd.h>
int main(void)
{
        fd_set rfds;
        struct timeval tv;
        int retval;
        /* Watch stdin (fd 0) to see when it has input. */
        FD_ZERO(&rfds);
        FD_SET(0, &rfds);
        /* Wait up to five seconds. */
        tv.tv_sec = 5;
        tv.tv_usec = 0;
        retval = select(1, &rfds, NULL, NULL, &tv);
        /* Don't rely on the value of tv now! */
        if (retval == -1)
                perror("select()");
```

```
        else if (retval)
                printf("Data is available now.\n");
        /* FD_ISSET(0, &rfds) will be true. */
        else
                printf("No data within five seconds.\n");
        return 0;
}
```

OUTPUT



8. DNS program.

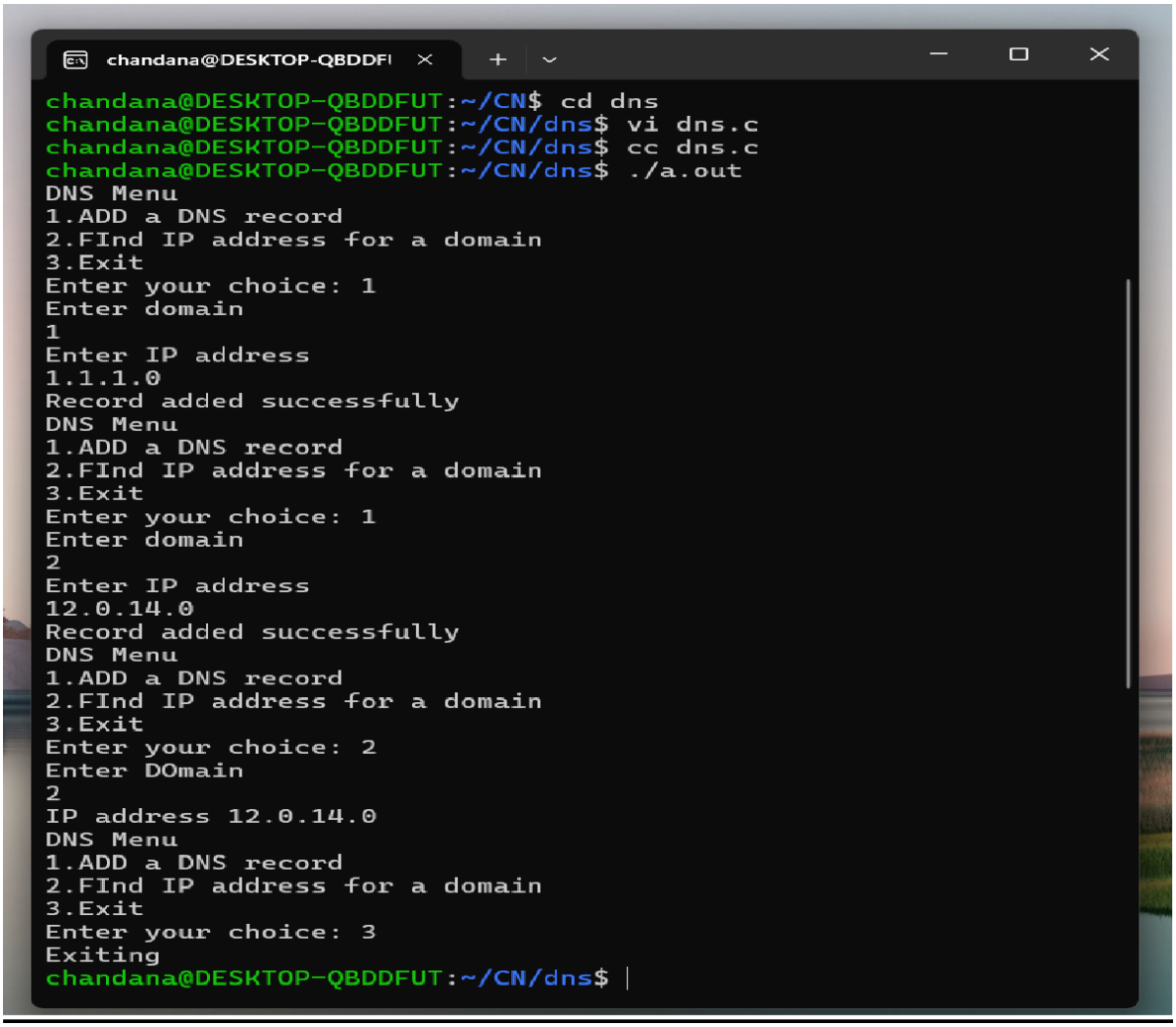
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX_SZ 10
typedef struct{
        char domain[MAX_SZ];
        char ip_add[MAX_SZ];
}DNS_rec;
DNS_rec records[MAX_SZ];
int num_rec=0;
void addRecord(char *domain,char *ip_add)
```

```
{
	if(num_rec>=MAX_SZ)
	{
		printf("Maximum no.of records reached\n");
		return;
	}
	strncpy(records[num_rec].domain,domain,MAX_SZ);
	strncpy(records[num_rec].ip_add,ip_add,MAX_SZ);
	num_rec++;
}
const char *findIP(const char *domain)
{
	int i=0;
	for(i=0;i<num_rec;i++)
	{
		if(strcmp(records[i].domain,domain)==0)
		{
			return records[i].ip_add;
		}
	}
	return NULL;
}
int main()
{
	int choice;
	char domain[MAX_SZ];
	char  ip_add[MAX_SZ];
	while(1)
	{
		printf("DNS Menu\n");
```

```
printf("1.ADD a DNS record\n");
printf("2.FInd IP address for a domain \n");
printf("3.Exit\n");
printf("Enter your choice: ");
scanf("%d",&choice);

switch(choice)
{
        case 1:{
                printf("Enter domain\n");
                scanf("%s",domain);
                printf("Enter IP address\n");
                scanf("%s",ip_add);
                addRecord(domain,ip_add);
                printf("Record added successfully\n");
                break;
                }
        case 2:{
                printf("Enter DOmain\n");
                scanf("%s",domain);
                const char *ip=findIP(domain);
                if(ip!=NULL)
                {
                        printf("IP address %s\n",ip);
                }
                else
                {
                        printf("Domain not found\n");
                }
                break;
```

```
                }
            case 3:
                {
                printf("Exiting\n");
                exit(0);
                }
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}
```

OUTPUT



9. Write a program to demonstrate scatter read and gather write.

SERVER

```
#include<stdio.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<sys/uio.h>
#include<unistd.h>
int main(int argc,char *argv[])
{
	int sockmain,sockcli,i,msglen,j;
	struct sockaddr_in server,client;
	struct iovec iv[3];
	char buffer1[]="Computer";
	char buffer2[]="Networks";
	char buffer3[]="Laboratory";
	iv[0].iov_base=buffer1;
	iv[0].iov_len=sizeof(buffer1)-1;
	iv[1].iov_base=buffer2;
	iv[1].iov_len=sizeof(buffer2)-1;
	iv[2].iov_base=buffer3;
	iv[2].iov_len=sizeof(buffer3)-1;
	if(argc!=2)
	{
		printf("USAGE: server <portno>\n");
		exit(1);
	}
	if((sockmain=socket(AF_INET,SOCK_STREAM,0))<0)
	{
```

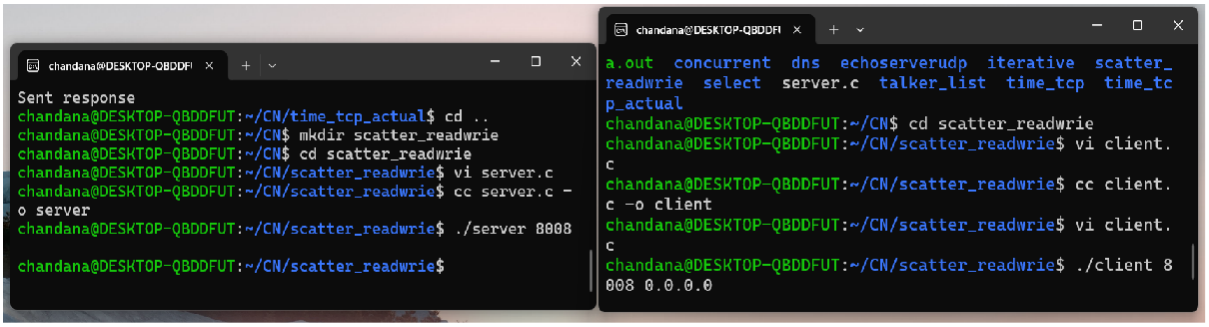
```
        perror("Socket error");
        exit(1);
    }
    server.sin_family=AF_INET;
    server.sin_port=htons(atoi(argv[1]));
    server.sin_addr.s_addr=htonl(INADDR_ANY);
    if((bind(sockmain,(struct sockaddr*)&server,sizeof(server)))<0)
    {
        perror("Listen error");
        exit(1);
    }
    if(listen(sockmain,5)<0)
    {
        perror("Listen Error");
        exit(1);
    }
    i=sizeof(client);
    if((sockcli=accept(sockmain,(struct  sockaddr*)&client,&i))<0)
    {
        perror("Server:accept error");
        exit(1);
    }
    if((writev(sockcli,iv,3))<0)
    {
        perror("Server:write error");
        exit(1);
    }
    printf("\n");
    return 0;
}
```

CLIENT

```
#include<stdio.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<sys/uio.h>
#include<unistd.h>
int main(int argc,char *argv[])
{
	int sockmain,msglen,i;
	struct sockaddr_in server,client;
	struct iovec iv[3];
	char buffer1[512];
	char buffer2[512];
	char buffer3[512];
	iv[0].iov_base=buffer1;
	iv[0].iov_len=sizeof(buffer1);
	iv[1].iov_base=buffer2;
	iv[1].iov_len=sizeof(buffer2);
	iv[2].iov_base=buffer3;
	iv[2].iov_len=sizeof(buffer3);
	if(argc!=3)
	{
		printf("Client <port no> <hostname>");
		exit(1);
	}
	if((sockmain=socket(AF_INET,SOCK_STREAM,0))<0)
	{
```

```
        perror("socket error");
        exit(1);
    }
    client.sin_family=AF_INET;
    client.sin_port=htons(0);
    client.sin_addr.s_addr=htonl(0L);
    server.sin_family=AF_INET;
    server.sin_port=htons(atoi(argv[1]));
    server.sin_addr.s_addr=inet_addr(argv[2]);
    if((bind(sockmain,(struct sockaddr*)&client,sizeof(client)))<0)
    {
        perror("Client bind error");
        exit(1);
    }
    if((connect(sockmain,(struct sockaddr*)&server,sizeof(server)))<0)
    {
        perror("Client connection error");
        exit(1);
    }
    if((readv(sockmain,iv,3))<0)
    {
        perror("server read");
        exit(1);
    }
    printf("%s",buffer1);
    printf("%s",buffer2);
    printf("%s",buffer3);
    printf("\n");
    return 0;
}
```

OUTPUT



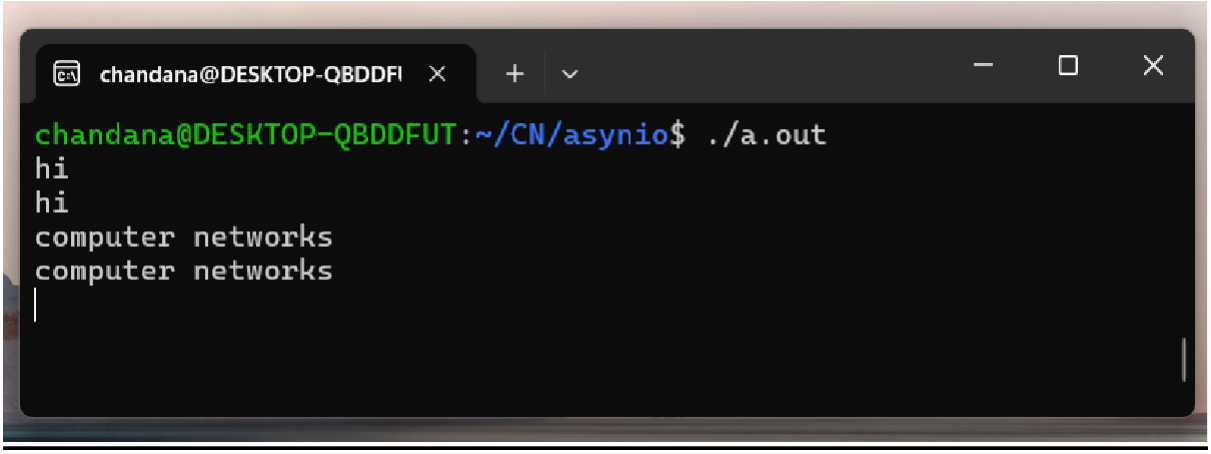
10. Write a program for Asynchronous I/O.

```
//copies given input and gives the same output


//#define FASYNC 0x1000
//#define F_SETOWN 400
//#define _POSIX_SOURCE
//#include<stdio.h>
//#include<stdlib.h>
#include<signal.h>
#include<fcntl.h>
//#include <sys/file.h>
//#define buffer_size 4096
int sigflag;
/*void sigio_func(int signum)
{
        sigflag=1;
}*/
int main()
{
        int n;
        char buff[100];
        int sigio_func();
```

```
signal(SIGIO,sigio_func);
if(fcntl(0,F_SETOWN,getpid())<0)
{
        printf("F_SETOWN error ");
}
if(fcntl(0,F_SETFL,FASYNC)<0)
{
        printf("F_SETFL FASYNC error");
}
for( ; ; )
{
        sigblock(sigmask(SIGIO));
        while(sigflag==0)
        {
                sigpause(0);
        }
        if((n=read(0,buff,100))>0)
        {
                if(write(1,buff,n)!=n)
                {
                        printf("Write error");
                }
        }
        else if(n<0)
        {
                printf("Write error");
        }
        else if(n==0)
        {
                exit(0);
```

```
            }
            sigflag=0;
            sigsetmask(0);
        }
        return 0;
}
int sigio_func()
{
        sigflag=1;
}
```

OUTPUT



11. Write a program for ioctl.

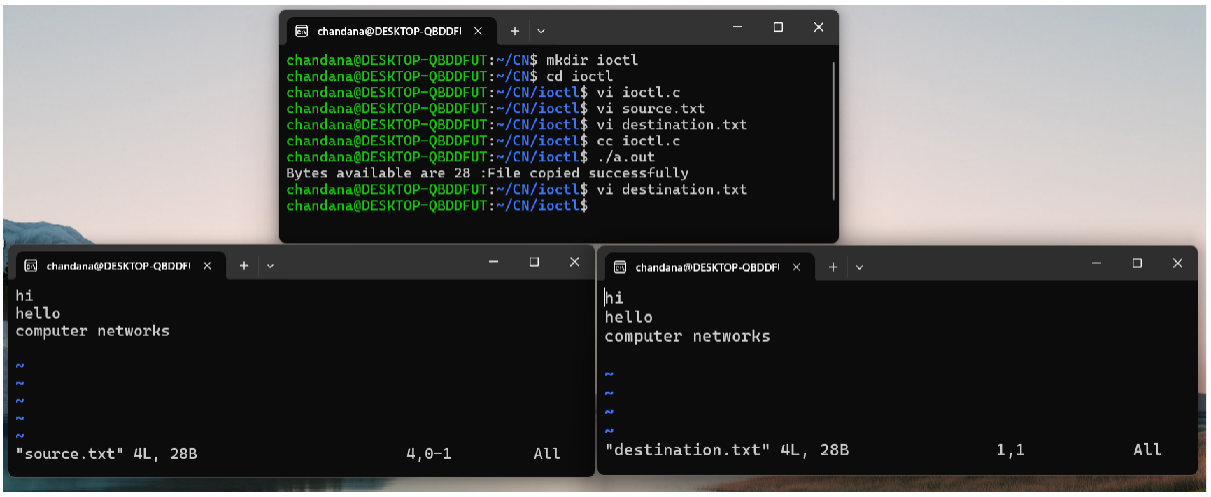
```
/*source.txt
This is to demonstrate ioctl
In this example we are copying file contents to another file
--destination.txt
This is to demonstrate ioctl
In this example we are copying file contents to another file*/

#include<stdio.h>
#include<stdlib.h>
```

```
#include<unistd.h>
#include<fcntl.h>
#include<sys/ioctl.h>
#define SOURCE_FILE "source.txt"
#define DESTINATION_FILE "destination.txt"
#define BUFFER_SIZE 1024
int main ()
{
	int source_fd,dest_fd;
	ssize_t bytes_read,bytes_written;
	char buffer[BUFFER_SIZE];
	int bytes_available;
	source_fd=open(SOURCE_FILE,O_RDONLY);
	if(source_fd==-1)
	{
		perror("open source file");
		exit(EXIT_FAILURE);
	}
	dest_fd=open(DESTINATION_FILE,O_WRONLY|O_CREAT|O_TRUNC,0644);
	if(dest_fd==-1)
	{
		perror("open destination file");
		close(source_fd);
		exit(EXIT_FAILURE);
	}
	if (ioctl (source_fd,FIONREAD,&bytes_available) == -1)
	{
		perror ("ioctl");
		close(source_fd);
		close(dest_fd);
```

```
        exit (EXIT_FAILURE);
}
printf("Bytes available are %d :",bytes_available);
while(bytes_available>0)
{
        bytes_read=read(source_fd,buffer,BUFFER_SIZE);
        if(bytes_read==-1)
        {
                perror("read");
                close(source_fd);
                close(dest_fd);
                exit (EXIT_FAILURE);
        }
        bytes_written=write(dest_fd,buffer,bytes_read);
        if(bytes_written==-1)
        {
                perror("write");
                close(source_fd);
                close(dest_fd);
                exit (EXIT_FAILURE);
        }
        if (ioctl (source_fd,FIONREAD,&bytes_available) == -1)
        {
                perror ("ioctl");
                close(source_fd);
                close(dest_fd);
                exit (EXIT_FAILURE);
        }
}
printf ("File copied successfully\n");
```

```
        close(source_fd);
        close(dest_fd);
        return 0;
}
```

OUTPUT



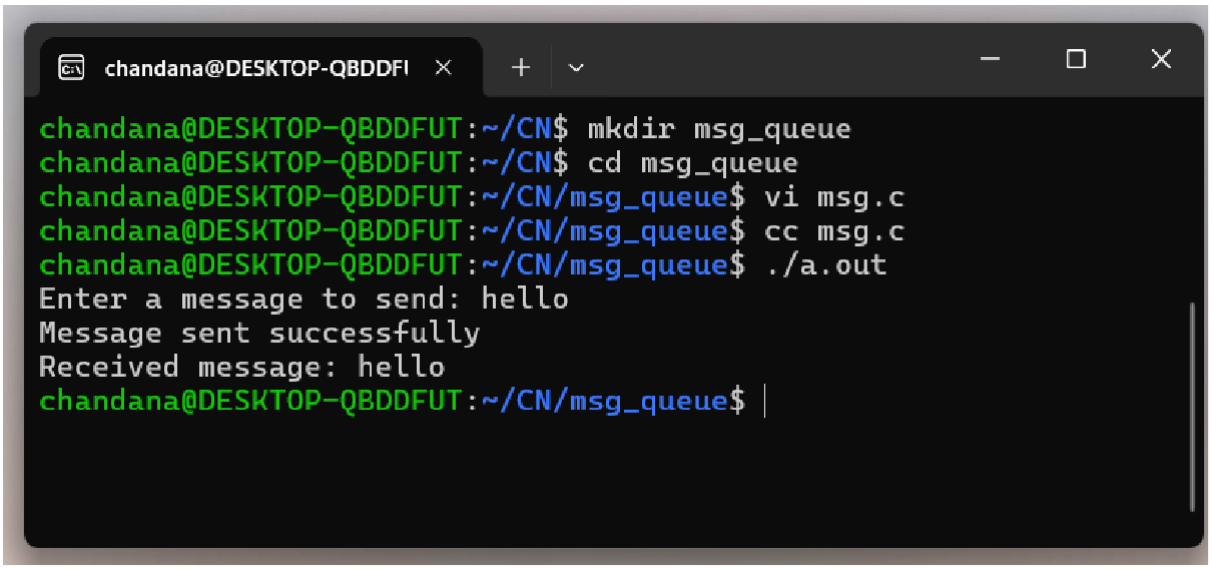
12. Write a program for IPC using message queues.

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<string.h>
#include<unistd.h>
#include<sys/msg.h>
#define MAX_MESSAGE_LENGTH 100
struct message
{
        long mtype;
        char mtext[MAX_MESSAGE_LENGTH];
};
int main()
{
```

```
key_t key=ftok("message_queue",65);
int msgid=msgget(key,0666|IPC_CREAT);
if(msgid==-1)
{
	perror("msgget");
	exit(1);
}
pid_t pid=fork();
if(pid==-1)
{
	perror("fork");
	exit(1);
}
if(pid==0)
{
	struct message received_msg;
	if(msgrcv(msgid,&received_msg,sizeof(received_msg.mtext),1,0)==-1)
	{
		perror("msgrcv");
		exit(1);
	}
	printf("Received message: %s\n",received_msg.mtext);
}
else
{
	struct message msg;
	msg.mtype=1;
	printf("Enter a message to send: ");
	fgets(msg.mtext,sizeof(msg.mtext),stdin);
	msg.mtext[strcspn(msg.mtext,"\n")]='\0';
```

```
        if(msgsnd(msgid,&msg,sizeof(msg.mtext),0)==-1)
        {
            perror("msgsnd");
            exit(1);
        }
        printf("Message sent successfully\n");
    }
    msgctl(msgid,IPC_RMID,NULL);
    return 0;
}
```

OUTPUT

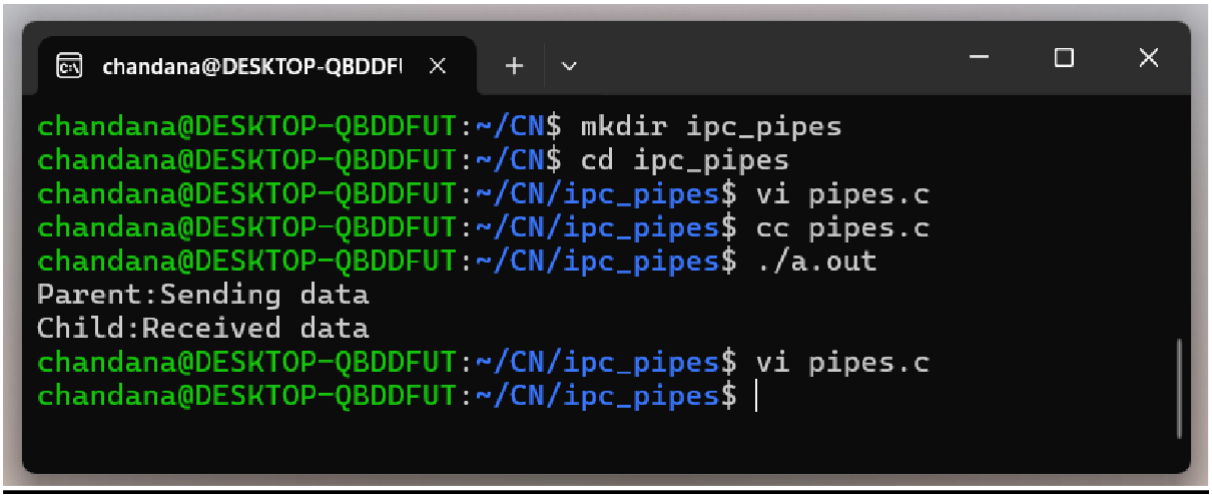


13. Write a program for IPC using pipes.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main()
{
    int fd[2],n;
    char buffer[100];
```

```
    pid_t p;
    pipe(fd);
    p=fork();
    if(p>0)
    {
        //printf("Parent:sending value to child\n");
        write(1,"Parent:Sending data\n",20);
        write(fd[1],"InterProcessCommunication",25);
    }
    else
    {
        //printf("\nChild:received data\n");
        write(1,"Child:Received data\n",20);
        n=read(fd[0],buffer,100);
        write(1,buffer,n);
    }
    return 0;
}
```

OUTPUT



14. Write an echo server program using threads.

SERVER

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<pthread.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
void * func(void * arg)
{
	int sd=*((int *)arg);
	char buf[512];
	int readbytes;
	while((readbytes=read(sd,buf,sizeof(buf)))!=0)
		{
			if(readbytes<0)
			{
				perror("SERVER:READ");
				exit(1);
			}
			if(write(sd,buf,readbytes)!=readbytes)
			{
				perror("SERVER:WRITE");
				exit(1);
			}
		printf("Written to client:%s\n",buf);
		memset(buf,'\0', sizeof(buf));
		}
	//printf("Written to client:%s\n",buf);
```

```
        close(sd);
        pthread_exit(NULL);

}
int main(int argc,char* argv[])
{
        int sd,len,nsd,readbytes;
        struct sockaddr_in servaddr,cliaddr;
        pthread_t tid[5];
        int i=0;
        if(argc!=2)
        {
                printf("<SERVER> portno");
        }
        if((sd=socket(AF_INET,SOCK_STREAM,0))<0)
        {
                perror("SERVER:SOCKET");
                exit(1);
        }
        servaddr.sin_family=AF_INET;
        servaddr.sin_port=htons(atoi(argv[1]));
        servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
        if(bind(sd,(struct sockaddr*)&servaddr,sizeof(servaddr))<0)
        {
                perror("SERVER:BIND");
                exit(1);
        }
        if(listen(sd,5)<0)
        {
                perror("SERVER:LISTEN");
```

```
            exit(1);
        }
        while(1)
        {
            len=sizeof(cliaddr);
            if((nsd=accept(sd,(struct sockaddr *)&cliaddr,&len))<0)
            {
                perror("SERVER:ACCEPT");
                exit(1);
            }
            pthread_create(&tid[i],NULL,func,(void *)&nsd);
            i=(i+1)%5;
        }
        close(sd);
        return 0;
}
```

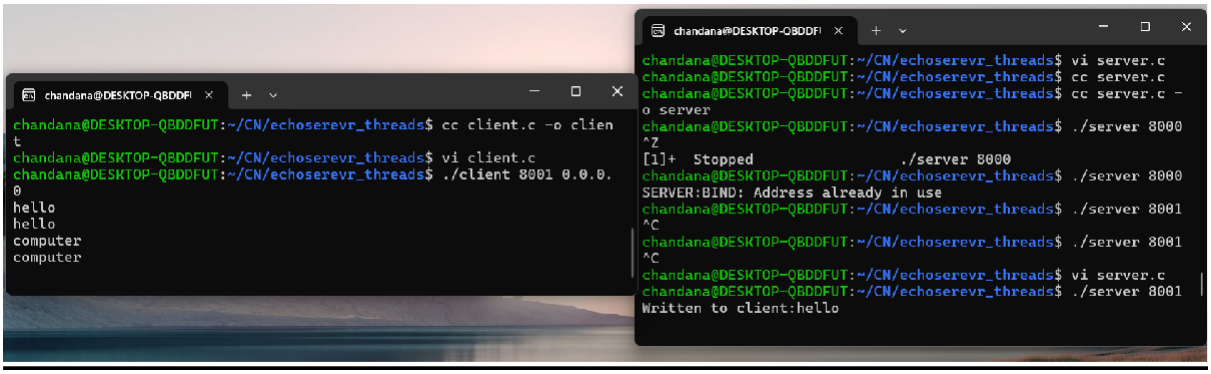
CLIENT

```
#include<stdlib.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<unistd.h>
int main(int argc,char* argv[])
{
        int sd,writtenbytes,readbytes,i;
        struct sockaddr_in servaddr,cliaddr;
        char buf[512];
        if(argc!=3)
        {
```

```
        perror("<CLIENT> PORT IPADDR");
        exit(1);
}
if((sd=socket(AF_INET,SOCK_STREAM,0))<0)
{
        perror("CLIENT:SOCKET");
        exit(1);
}
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(atoi(argv[1]));
servaddr.sin_addr.s_addr=inet_addr(argv[2]);
/*if(bind(sd,(struct sockaddr *)&cliaddr,sizeof(cliaddr))<0)
{
        perror("CLIENT:BIND");
        exit(1);
}*/
if((connect(sd,(struct sockaddr *)&servaddr,sizeof(servaddr)))<0)
{
        perror("CLIENT:CONNECT");
        exit(1);
}
for(;;)
{
        if((i=read(0,buf,sizeof(buf)))<0)
        {
                perror("CLIENT:READ STDIN");
                exit(1);
        }
        if((writtenbytes=write(sd,buf,i))<0)
        {
```

```
            perror("CLIENT:WRITE");
            exit(1);
        }
        if((readbytes=read(sd,buf,writtenbytes))<0)
        {
            perror("CLIENT:READ");
            exit(1);
        }
        if(write(1,buf,readbytes)!=readbytes)
        {
            perror("CLIENT:WRITE STDOUT");
            exit(1);
        }
    }
    return 0;
}
```

OUTPUT



15. Write the program to display the file contents on sending the file name to the server.

SERVER

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
```

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<arpa/inet.h>
short portno;
int main(int argc,char *argv[])
{
        int sd,i;
        char buffer[512];
        struct sockaddr_in servaddr,cliaddr;
        int addrlength,msglength;
        if(argc!=2)
        {
                printf("SERVER:<portno>\n");
                exit(1);
        }
        if((sd=socket(AF_INET,SOCK_DGRAM,0))<0)
        {
                perror("SERVER:socket error\n");
                exit(1);
        }
        portno=atoi(argv[1]);
        servaddr.sin_family=AF_INET;
        servaddr.sin_port=htons(portno);
        servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
        i=sizeof(servaddr);
        if(bind(sd,(struct sockaddr*)&servaddr,i)<0)
        {
                perror("SERVER:bind  errror");
```

```
            exit(1);
        }
        addrlength=sizeof(cliaddr);
        msglength=recvfrom(sd,(char*)buffer,sizeof(buffer),0,(struct
sockaddr*)&cliaddr,(socklen_t*)&(addrlength));
        if(msglength<0)
        {
            perror("SERVER:recvfrom error\n");
            exit(1);
        }
        printf("\tFILE TRANSFER\n");
        printf("Received file name %s\n",buffer);
        FILE *file=fopen(buffer,"r");
        if(file==NULL)
        {
            perror("Error openig file");
            exit(1);
        }
        printf("Contents of %s\n",buffer);
        while(fgets(buffer,sizeof(buffer),file)!=NULL)
        {
            printf("%s",buffer);
        }
        fclose(file);
        close(sd);
        return 0;
}
```

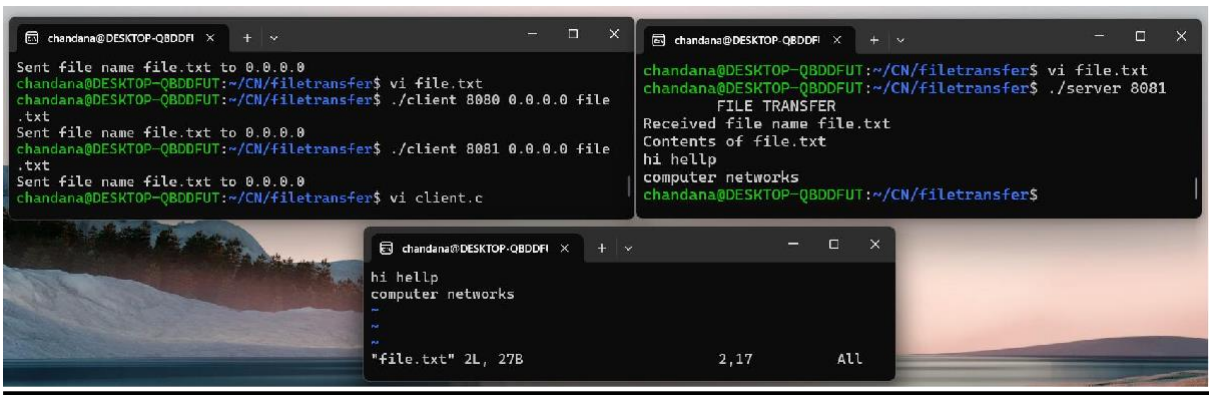
CLIENT

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
```

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<string.h>
#include<arpa/inet.h>
int main(int argc,char *argv[])
{
	int sd,i;
	char buffer[512];
	int addelength,no_of_bytes;
	struct sockaddr_in servaddr,cliaddr;
	if(argc!=4)
	{
		printf("CLIENT: <portno> <hostname> <filename>\n");
		exit(1);
	}
	if((sd=socket(AF_INET,SOCK_DGRAM,0))<0)
	{
		perror("CLIENT: socket error\n");
		exit(1);
	}
	cliaddr.sin_family=AF_INET;
	cliaddr.sin_port=htons(0);
	cliaddr.sin_addr.s_addr=htonl(0L);
	if(bind(sd,(struct sockaddr*)&cliaddr,sizeof(cliaddr))<0)
	{
		perror("CLIENT: bind error\n");
		exit(1);
	}
```

```
        servaddr.sin_family=AF_INET;
        servaddr.sin_port=htons(atoi(argv[1]));
        servaddr.sin_addr.s_addr=inet_addr(argv[2]);
        no_of_bytes=sendto(sd,argv[3],strlen(argv[3]),0,(struct
sockaddr*)&servaddr,sizeof(servaddr));
        if(no_of_bytes<0)
        {
                perror("CLIENT: sento error\n");
                exit(1);
        }
        printf("Sent file name %s to %s\n",argv[3],inet_ntoa(servaddr.sin_addr));
        close(sd);
        return 0;
}
```

OUTPUT



16. Write a program to perform the program execution /Command execution on sending the appropriate command to the server.

SERVER

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<arpa/inet.h>
```

```
#include<netinet/in.h>
#include<netdb.h>
int main(int argc,char *argv[])
{
	int sockfd;
	char buffer[512];
	struct sockaddr_in servaddr,cliaddr;
	int cliaddrlength,msglength;
	if(argc!=2)
	{
		printf("SERVER: <port no>");
		exit(1);
	}
	if((sockfd=socket(AF_INET,SOCK_DGRAM,0))<0)
	{
		perror("SERVER: socket error");
		exit(1);
	}
	servaddr.sin_family=AF_INET;
	servaddr.sin_port=htons(atoi(argv[1]));
	servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
	if(bind(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr))<0)
	{
		perror("SERVER:bind error");
		exit(1);
	}
	cliaddrlength=sizeof(cliaddr);
	msglength=recvfrom(sockfd,(char*)buffer,sizeof(buffer),0,(struct
sockaddr*)&cliaddr,&cliaddrlength);
	if(msglength<0)
	{
```

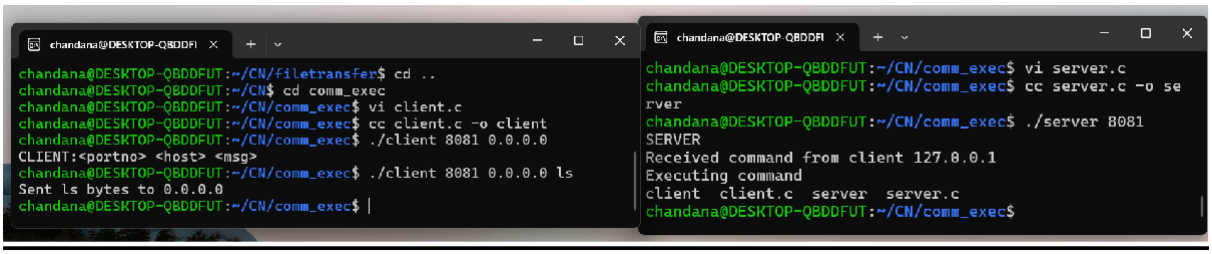
```
        perror("SERVER:recvfrom error");
        exit(1);
    }
    printf("SERVER\n");
    printf("Received command from client %s\n",inet_ntoa(cliaddr.sin_addr));
    printf("Executing command\n");
    system(buffer);
    return 0;
}
```

CLIENT

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<string.h>
int main(int argc,char *argv[])
{
    int sockfd;
    char buffer[512];
    struct sockaddr_in servaddr,cliaddr;
    int portno;
    int no_of_bytes,addrlength;
    if(argc!=4)
    {
        printf("CLIENT:<portno> <host> <msg>\n");
        exit(1);
    }
    if((sockfd=socket(AF_INET,SOCK_DGRAM,0))<0)
```

```
    {
        perror("CLIENT:socket error\n");
        exit(1);
    }
    cliaddr.sin_family=AF_INET;
    cliaddr.sin_port=htons(0);
    cliaddr.sin_addr.s_addr=htonl(0L);
    if(bind(sockfd,(struct sockaddr*)&cliaddr,sizeof(cliaddr))<0)
    {
        perror("CLIENT:bind error\n");
        exit(1);
    }
    servaddr.sin_family=AF_INET;
    portno=atoi(argv[1]);
    servaddr.sin_port=htons(portno);
    servaddr.sin_addr.s_addr=inet_addr(argv[2]);
    no_of_bytes=sendto(sockfd,argv[3],strlen(argv[3]),0,(struct
sockaddr*)&servaddr,sizeof(servaddr));
    if(no_of_bytes<0)
    {
        perror("CLIENT:sendto error");
        exit(1);
    }
    printf("Sent %s bytes to %s\n",argv[3],inet_ntoa(servaddr.sin_addr));
    return 0;
}
```

OUTPUT



17. Write an Echo server using TCP.

SERVER

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<unistd.h>
int main(int argc,char* argv[])
{
	int sd,len,nsd,readbytes;
	char buf[512];
	struct sockaddr_in servaddr,cliaddr;
	if(argc!=2)
	{
		printf("<SERVER> portno");
	}
	if((sd=socket(AF_INET,SOCK_STREAM,0))<0)
	{
		perror("SERVER:SOCKET");
		exit(1);
	}
	servaddr.sin_family=AF_INET;
	servaddr.sin_port=htons(atoi(argv[1]));
```

```
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
if(bind(sd,(struct sockaddr*)&servaddr,sizeof(servaddr))<0)
{
	perror("SERVER:BIND");
	exit(1);
}
if(listen(sd,5)<0)
{
	perror("SERVER:LISTEN");
	exit(1);
}
while(1)
{
	len=sizeof(cliaddr);
	if((nsd=accept(sd,(struct sockaddr *)&cliaddr,&len))<0)
	{
		perror("SERVER:ACCEPT");
		exit(1);
	}
	while((readbytes=read(nsd,buf,sizeof(buf)))!=0)
	{
		if(readbytes<0)
		{
			perror("SERVER:READ");
			exit(1);
		}
		if(write(nsd,buf,readbytes)!=readbytes)
		{
			perror("SERVER:WRITE");
			exit(1);
```

```
                }
                write(1,buf,readbytes);
            }
        }
        return 0;
}
```

CLIENT

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>

#include<unistd.h>
int main(int argc,char* argv[])
{
        int sd,writtenbytes,readbytes,i;
        struct sockaddr_in servaddr,cliaddr;
        char buf[512];
        if(argc!=3)
        {
            perror("<CLIENT> PORT IPADDR");
            exit(1);
        }
        if((sd=socket(AF_INET,SOCK_STREAM,0))<0)
        {
            perror("CLIENT:SOCKET");
            exit(1);
        }
```

```
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(atoi(argv[1]));
servaddr.sin_addr.s_addr=inet_addr(argv[2]);
/*if(bind(sd,(struct sockaddr *)&cliaddr,sizeof(cliaddr))<0)
{
        perror("CLIENT:BIND");
        exit(1);
}*/
if((connect(sd,(struct sockaddr *)&servaddr,sizeof(servaddr)))<0)
{
        perror("CLIENT:CONNECT");
        exit(1);
}
for(;;)
{
        if((i=read(0,buf,sizeof(buf)))<0)
        {
                perror("CLIENT:READ STDIN");
                exit(1);
        }
        if((writtenbytes=write(sd,buf,i))<0)
        {
                perror("CLIENT:WRITE");
                exit(1);
        }
        if((readbytes=read(sd,buf,writtenbytes))<0)
        {
                perror("CLIENT:READ");
                exit(1);
        }
```

```
            if(write(1,buf,readbytes)!=readbytes)
            {
                perror("CLIENT:WRITE STDOUT");
                exit(1);
            }
        }
        return 0;
}
```

OUTPUT