

Pulakesh Bag

Roll No: 120CS0131

Computer Science & Engineering
(2020-24)

Data Communications & Computer
Network Laboratory
Assignment 5 (09-02-2023)

Q.1 Run the program given in UDP_Socket document (Example 6.1,6.2,6.3 and 6.4)

Answer:

//Client Side Code

```
#include "unp.h"
```

```
void dg_cli(FILE *fp, int sockfd, const SA *pservaddr, socklen_t  
servlen)
```

```
{  
    int n;  
    char sendline[MAXLINE], recvline[MAXLINE + 1];  
    while (fgets(sendline, MAXLINE, fp) != NULL) {  
        sendto(sockfd, sendline, strlen(sendline), 0, pservaddr,  
servlen);  
        n = recvfrom(sockfd, recvline, MAXLINE, 0, NULL, NULL);  
        recvline[n] = 0; /* null terminate */  
        fputs(recvline, stdout);  
        printf("\n");  
    }  
}
```

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

```
{  
    int sockfd;  
    struct sockaddr_in servaddr;  
    if (argc != 2)  
        exit(0);  
    bzero(&servaddr, sizeof(servaddr));  
    servaddr.sin_family = AF_INET;  
    servaddr.sin_port = htons(SERV_PORT);  
    inet_pton(AF_INET, argv[1], &servaddr.sin_addr);  
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);  
    dg_cli(stdin, sockfd, (SA*)&servaddr, sizeof(servaddr));  
    exit(0);  
}
```

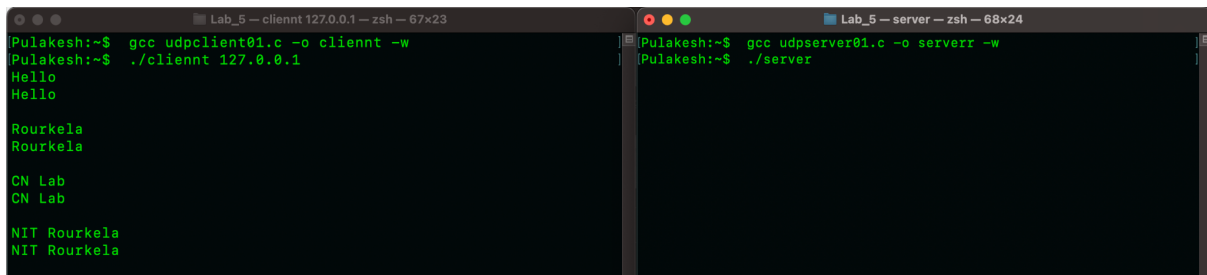
//Server Side Code

```
#include "unp.h"
```

```
void dg_echo(int sockfd, SA* pcliaddr, socklen_t clilen)
{
    int n;
    socklen_t len;
    char mesg[MAXLINE];
    for (; ; ) {
        len = clilen;
        n = recvfrom(sockfd, mesg, MAXLINE, 0, pcliaddr, &len);
        sendto(sockfd, mesg, n, 0, pcliaddr, len);
    }
}

int main(int argc, char** argv)
{
    int sockfd;
    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 = htonl(INADDR_ANY);
    servaddr.sin_port = htons(SERV_PORT);
    bind(sockfd, (SA*)&servaddr, sizeof(servaddr));
    dg_echo(sockfd, (SA*)&cliaddr, sizeof(cliaddr));
}
```

Output:



```
Lab_5 - cliennt 127.0.0.1 - zsh - 67x23
Pulakesh:~$ gcc udpclient01.c -o cliennt -w
Pulakesh:~$ ./cliennt 127.0.0.1
Hello
Hello

Rourkela
Rourkela

CN Lab
CN Lab

NIT Rourkela
NIT Rourkela

Lab_5 - server - zsh - 68x24
Pulakesh:~$ gcc udpserver01.c -o serverr -w
Pulakesh:~$ ./server
```

Q.2 Execute a client/server program using UDP service for adding a two integer numbers requested by the client and evaluated at server and get back result at the client

Code:

```
//UDP Client side code
```

```
#include<sys/types.h>
```

```
#include<sys/socket.h>
```

```
#include<netinet/in.h>
```

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

```
#include<netdb.h>
```

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

```
#include<netdb.h>
```

```
#include<arpa/inet.h>
```

```
#include<strings.h>
```

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

```
#define maxline 4096
```

```
#include "sum.h"
```

```
void str_cli(FILE *,int);
```

```
/* code for str_cli() function */
```

```
void str_cli(FILE *fp,int sockfd)
```

```
{
```

```
    char sendline[maxline],recvline[maxline];
```

```
    struct args args;
```

```
    struct result result;
```

```

while(fgets(sendline,maxline,fp)!=NULL)
{
    if(sscanf(sendline,"%ld %ld", &args.arg1, &args.arg2)
!=2)
    {
        printf("\ninvlid input:%S\n",sendline);
        continue;
    }
    printf("the values are %ld\t%ld\n",args.arg1,args.arg2);
    write(sockfd,&args,sizeof(args));
    if(read(sockfd,&result,sizeof(result))==0)
        printf("error");
    //printf("socket desc read is %d\n",sockfd);
    printf("result is = %ld\n",result.sum_result);
}
}

```

```

int main(int argc,char **argv)
{
    int sockfd,n,cn;
    char recvline[maxline+1];
    struct sockaddr_in servaddr;
    if(argc !=2)
        printf("usage: a.out <IPaddress>");

    if((sockfd=socket(AF_INET,SOCK_STREAM,0))<0)

```

```

        printf("\nsocket error");

    bzero(&servaddr,sizeof(servaddr));

    servaddr.sin_family=AF_INET;

    servaddr.sin_addr.s_addr=inet_addr(argv[1]);
    //inet_addr("127.0.0.1");

    servaddr.sin_port=htons(4500);

    if(inet_pton(AF_INET,argv[1],&servaddr.sin_addr)<=0)
        printf("\ninet_pton error for %s", argv[1]);

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

    if(cn < 0)
    {
        printf("\nconnect error\n");
        exit(0);
    }
    else
        printf("\nconnection has been established\n");
    str_cli(stdin,sockfd);
    exit(0);
}

```

```
//UDP Server side code
```

```
#include<sys/types.h>
```

```
#include<sys/socket.h>
```

```
#include<netinet/in.h>
```

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

```
#include<netdb.h>
```

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

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

```
#include<time.h>
```

```
#include<strings.h>
```

```
#define maxline 4096
```

```
#define LISTENQ 5
```

```
#include "sum.h"
```

```
void str_echo(int);
```

```
/* code for str_echo function */
```

```
void str_echo(int sockfd)
```

```
{
```

```
    ssize_t n;
```

```
    struct args args;
```

```
    struct result result;
```

```
    char line[maxline];
```

```
    for(;;)
```

```
    {
```

```
        n=read(sockfd,&args,sizeof(args));
```



```

        if(n==0)
            return;

        printf("the values received from client are %ld\n",args.arg1,args.arg2);

        result.sum_result=args.arg1 + args.arg2;

        printf("the value returned by server is %ld\n",result.sum_result);

        //break;//result.sum = args.arg1 -args.arg2;

        //break;

        write(sockfd,&result,sizeof(result));

    }
}

```

```

int main(int argc,char **argv)
{
    int listenfd,connfd,bd;
    pid_t childpid;
    struct sockaddr_in servaddr;
    char buff[maxline];
    time_t ticks;
    if(( listenfd=socket(AF_INET,SOCK_STREAM,0))<0)
    {
        printf("error");
    }

    bzero(&servaddr,sizeof(servaddr));

    servaddr.sin_family=AF_INET;

    servaddr.sin_addr.s_addr=htonl(INADDR_ANY);

    servaddr.sin_port=htons(4500);
}

```

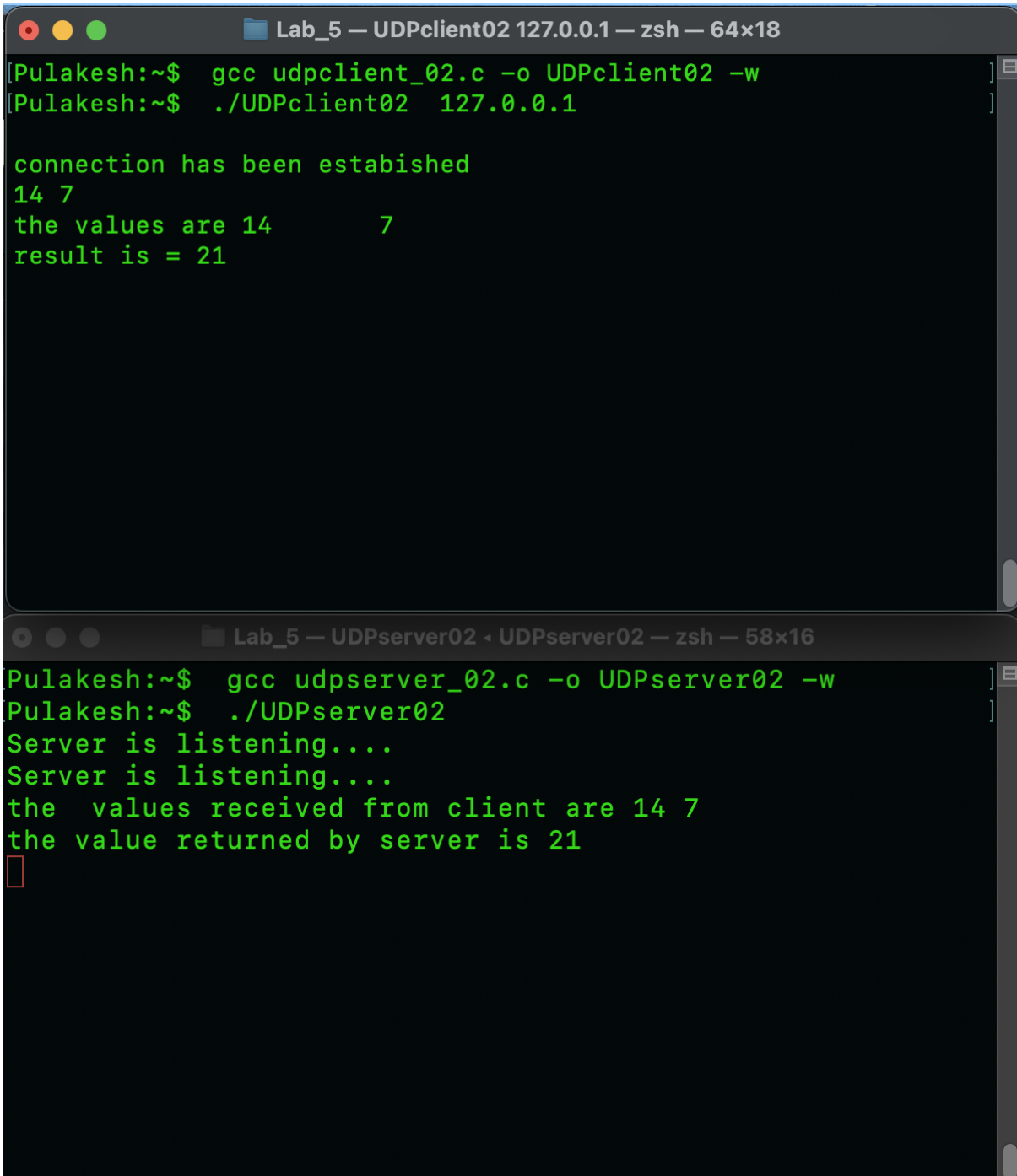
```

        if((bd=bind(listenfd,(struct
sockaddr*)&servaddr,sizeof(servaddr)))<0)
        {
            printf("\nbind error");
            exit(0);
        }

listen(listenfd,LISTENQ);
for(;;)
{
    printf("Server is listening....\n");
    connfd=accept(listenfd,(struct sockaddr*)NULL,NULL);
    if((childpid=fork())==0)
    {
        close(listenfd);
        str_echo(connfd);
        exit(0);
    }
    close(connfd);
}
}

```

Output:



The image shows two terminal windows. The top window, titled 'Lab_5 — UDPclient02 127.0.0.1 — zsh — 64x18', shows the compilation of 'udpclient_02.c' into 'UDPclient02' and its execution with arguments '127.0.0.1'. The output indicates a successful connection and the calculation of the sum of 14 and 7, resulting in 21. The bottom window, titled 'Lab_5 — UDPserver02 • UDPserver02 — zsh — 58x16', shows the compilation of 'udpserver_02.c' into 'UDPserver02' and its execution. The server output shows it is listening and receives the values 14 and 7 from the client, returning the sum 21. A red cursor is visible at the end of the last line in the bottom terminal.

```
Lab_5 — UDPclient02 127.0.0.1 — zsh — 64x18
[Pulakesh:~$ gcc udpclient_02.c -o UDPclient02 -w
[Pulakesh:~$ ./UDPclient02 127.0.0.1

connection has been established
14 7
the values are 14      7
result is = 21

Lab_5 — UDPserver02 • UDPserver02 — zsh — 58x16
Pulakesh:~$ gcc udpserver_02.c -o UDPserver02 -w
Pulakesh:~$ ./UDPserver02
Server is listening....
Server is listening....
the values received from client are 14 7
the value returned by server is 21
█
```

Q.3 Execute a client/server program for simple calculator having following operations addition, subtraction, multiplication, division, and detecting prime numbers. Input entered at the client side and evaluated at server machine and get back result at the client machine. Try this question for both TCP and UDP socket programming

Code:

```
//UDP Client side code
#include "unp.h"
#include "sum.h"
void dg_cli(FILE *fp, int sockfd, const SA *pservaddr, socklen_t
servlen)
{
    int n=0;
    char sendline[MAXLINE], recvline[MAXLINE + 1];
    struct args args;
    struct result result;

    printf("Connected by UDP...\n\n");

    while (fgets(sendline, MAXLINE, fp) != NULL) {

        if(sscanf(sendline,"%ld %ld", &args.arg1, &args.arg2) !=2)
        {
            printf("\ninvalid input:%S\n",sendline);
            continue;
        }

        sendto(sockfd, &args, sizeof(args), 0, pservaddr, servlen);

        printf("the input numbers are
%ld\t%ld\n",args.arg1,args.arg2);

        n = recvfrom(sockfd, &result, sizeof(result), 0, pservaddr,
&servlen);

        //printf("n = %d, error is %d\n",n,errno);
        //printf("received message size %d\n", n);
        //result[n] = 0; /* null terminate */
        //fputs(result, stdout);
    }
}
```

```

        printf("\nreceived sum result from server =
%d\n",result.sum_result);
        printf("received subtract result from server =
%d\n",result.subtract_result);
        printf("received multiply result from server =
%d\n",result.multiply_result);
        printf("received division result from server =
%d\n",result.division_result);
        printf("received modulo result from server =
%d\n",result.modulo_result);
        printf("received arg1 is_prime from server =
%d\n",result.isprime_arg1);
        printf("received arg2 is_prime from server =
%d\n\n",result.isprime_arg2);
    }
}

int main(int argc, char** argv)
{
    int sockfd;
    struct sockaddr_in servaddr;
    if (argc != 2)
        exit(0);
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(SERV_PORT);
    inet_pton(AF_INET, argv[1], &servaddr.sin_addr);
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    dg_cli(stdin, sockfd, (SA*)&servaddr, sizeof(servaddr));
    exit(0);
}

```

```
//UDP Server side code
```

```
#include "unp.h"
```

```
#include "sum.h"
```

```
long isprime(long n)
```

```
{
```

```
    long i;
```

```
    // Corner case
```

```
    if (n <= 1)
```

```
        return 0;
```

```
    // Check from 2 to n-1
```

```
    for (i = 2; i < n; i++)
```

```
        if (n % i == 0)
```

```
            return 0;
```

```
    return 1;
```

```
}
```

```
void dg_echo(int sockfd, SA* pcliaddr, socklen_t clilen)
```

```
{
```

```
    ssize_t n;
```

```
    struct args args;
```

```
    socklen_t len;
```

```
    struct result result;
```

```
    printf("Connected by UDP...\n\n");
```

```
    for(;;)
```

```
    {
```

```
        //n=read(sockfd,&args,sizeof(args));
```

```
        len = clilen;
```

```
        n = recvfrom(sockfd, &args, sizeof(args), 0, pcliaddr,  
&len);
```

```
        if(n==0)
```

```
            return;
```

```
        printf("the values received from client are %ld %ld\n",  
args.arg1, args.arg2);
```

```
        result.sum_result=args.arg1 + args.arg2;
```

```
        result.multiply_result = (args.arg1 * args.arg2);
```

```
        result.division_result = (args.arg1 / args.arg2);
```

```
        result.subtract_result = (args.arg1 - args.arg2);
```

```
        result.division_result = (args.arg1 / args.arg2);
```

```
        result.modulo_result = (args.arg1) % (args.arg2);
```

```

        result.isprime_arg2 = isprime(args.arg2);
        result.isprime_arg1 = isprime(args.arg1);

        sendto(sockfd, &result, sizeof(result), 0, pcliaddr, len);

        //printf("the value returned by server is %ld\n
",result.sum);
        //write(sockfd,&result,sizeof(result));

    }
}
int main(int argc, char** argv)
{
    int sockfd;
    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 = htonl(INADDR_ANY);
    servaddr.sin_port = htons(SERV_PORT);
    bind(sockfd, (SA*)&servaddr, sizeof(servaddr));
    dg_echo(sockfd, (SA*)&cliaddr, sizeof(cliaddr));
}

```

```
//TCP Client side code
```

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

```
#define maxline 4096
#include "sum.h"
```

```
void str_cli(FILE *,int);
```

```
/* code for str_cli() function */
```

```
void str_cli(FILE *fp,int sockfd)
{
    char sendline[maxline],recvline[maxline];
    struct args args;
    struct result result;

    printf("Connected by TCP...\n\n");

    while(fgets(sendline,maxline,fp)!=NULL)
    {
        if(sscanf(sendline,"%ld %ld", &args.arg1, &args.arg2)
!=2)
        {
            printf("\ninvalid input:%S\n",sendline);
            continue;
        }

        printf("the values are %ld\t%ld\n",args.arg1,args.arg2);

        write(sockfd,&args,sizeof(args));

        if(read(sockfd,&result,sizeof(result))==0)
            printf("error");
    }
}
```



```

        printf("\nreceived sum result from server =
%d\n",result.sum_result);
        printf("received subtract result from server =
%d\n",result.subtract_result);
        printf("received multiply result from server =
%d\n",result.multiply_result);
        printf("received division result from server =
%d\n",result.division_result);
        printf("received modulo result from server =
%d\n",result.modulo_result);
        printf("received arg1 is_prime from server =
%d\n",result.isprime_arg1);
        printf("received arg2 is_prime from server =
%d\n\n",result.isprime_arg2);
    }
}

```

```

int main(int argc,char **argv)
{
    int sockfd,n,cn;
    char recvline[maxline+1];
    struct sockaddr_in servaddr;

    //printf("%d\n%s %s",argc,argv[1],argv[2]);

    if(argc !=2)
        printf("usage: a.out <IPaddress>");

    if((sockfd=socket(AF_INET,SOCK_STREAM,0))<0)
        printf("\nsocket error");

    bzero(&servaddr,sizeof(servaddr));

    servaddr.sin_family=AF_INET;
    servaddr.sin_addr.s_addr=inet_addr(argv[1]);
    //inet_addr("127.0.0.1");
    servaddr.sin_port=htons(4500);

    if(inet_pton(AF_INET,argv[1],&servaddr.sin_addr)<=0)
        printf("\ninet_pton error for %s", argv[1]);
}

```

```
    cn=connect(sockfd,(struct
sockaddr*)&servaddr,sizeof(servaddr));

    if(cn < 0)
    {
        printf("\nconnection error\n");
        exit(0);
    }
    else
        printf("\nconnection has been established\n");

    str_cli(stdin,sockfd);

    exit(0);
}
```

//TCP Server side code

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

```
#define maxline 4096
#define LISTENQ 5
#include "sum.h"
```

```
void str_echo(int);
```

```
long isprime(long n)
{
```

```
    long i;
    // Corner case
    if (n <= 1)
        return 0;
```

```
    // Check from 2 to n-1
    for (i = 2; i < n; i++)
        if (n % i == 0)
            return 0;
```

```
    return 1;
```

```
}
```

```
/* code for str_echo function */
```

```
void str_echo(int sockfd)
```

```
{
```

```
    ssize_t n;
    struct args args;
```

```

    struct result result;
    char line[maxline];
    printf("Connected by TCP...\n\n");

    for(;;)
    {
        n=read(sockfd,&args,sizeof(args));
        if(n==0)
            return;
        printf("the values received from client are %ld
%ld\n",args.arg1,args.arg2);

        result.sum_result=args.arg1 + args.arg2;
        result.multiply_result = (args.arg1 * args.arg2);
        result.division_result = (args.arg1 / args.arg2);
        result.subtract_result = (args.arg1 - args.arg2);
        result.division_result = (args.arg1 / args.arg2);
        result.modulo_result = (args.arg1) % (args.arg2);

        result.isprime_arg2 = isprime(args.arg2);
        result.isprime_arg1 = isprime(args.arg1);

        // printf("the value returned by server is %ld\n ",result.sum);
        write(sockfd,&result,sizeof(result));
    }
}

int main(int argc,char **argv)
{
    int listenfd,connfd,bd;
    pid_t childpid;
    struct sockaddr_in servaddr;
    char buff[maxline];
    time_t ticks;
    if((listenfd=socket(AF_INET,SOCK_STREAM,0))<0)

```

```

{
    printf("error");
}
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(4500);

if((bd=bind(listenfd,(struct sockaddr*)&servaddr,sizeof(servaddr)))<0)
{
    printf("\nbind error");
    exit(0);
}

listen(listenfd,LISTENQ);
for(;;)
{
    printf("Server is listening....\n");
    connfd=accept(listenfd,(struct sockaddr*)NULL,NULL);
    if((childpid=fork())==0)
    {
        close(listenfd);
        str_echo(connfd);
        exit(0);
    }
    close(connfd);
}
}

```

Output:

```
Lab_5 — TCPclient 127.0.0.1 — zsh — 64x18
[Pulakesh:~$ gcc udpclient_03_2.c -o TCPclient -w
[Pulakesh:~$ ./TCPclient 127.0.0.1

connection has been established
Connected by TCP...

14 7
the values are 14      7

received sum result from server = 21
received subtract result from server = 7
received multiply result from server = 98
received division result from server = 2
received modulo result from server = 0
received arg1 is_prime from server = 0
received arg2 is_prime from server = 1

Lab_5 — TCPserver < TCPserver — zsh — 58x16
Pulakesh:~$ gcc udpserver_03_2.c -o TCPserver -w
Pulakesh:~$ ./TCPserver
Server is listening....
Server is listening....
Connected by TCP...

the values received from client are 14 7

```

```
Lab_5 — UDPserver — zsh — 74x14
[Pulakesh:~$ gcc udpserver02.c -o UDPserver -w
[Pulakesh:~$ ./UDPserver
Connected by UDP...

the values received from client are 14 7
█
```

```
Lab_5 — UDPclient 127.0.0.1 — zsh — 67x16
Pulakesh:~$ gcc udpclient02.c -o UDPclient -w
Pulakesh:~$ ./UDPclient 127.0.0.1
Connected by UDP...

14 7
the input numbers are 14      7

received sum result from server = 21
received subtract result from server = 7
received multiply result from server = 98
received division result from server = 2
received modulo result from server = 0
received arg1 is_prime from server = 0
received arg2 is_prime from server = 1
```