Certificate

Name: M . C . SOHAN Class:

Roll No: Exam No:

Institution R . V . College of Engineering

This is certified to be the bonafide work of the student in the Network Programming & Security *Laboratory during the academic year 2020 / 2021 .*

No. of practicals certified _____ *out of* _____ *in the subject of* Network Programming & Security.

.....
Teacher In-charge

.....
Examiner's Signature

.....
Principal

Date: .....

Institution Rubber Stamp

(N.B: The candidate is expected to retain his/her journal till he/she passes in the subject.)

Output:

SERVER

```
$ ./filesrv
  Listen called
   Accepted
    text
    file sent
    Accepted
     notavailable.txt
```

CLIENT

```
$ ./filecli 127.0.0.1 text
   socket created : 3
   file text requested
    The contents of this file will be displayed
   l1
   l2
   End of file - there is a new line after this

$ ./filecli 127.0.0.1 notavailable.txt
    socket created : 3
    File notavailable.txt requested
    Unable to open the requested file.
```

Date

Expt. No. 1. Page No. 1

1. Implement a client and server communication using socket programming.

* File server

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

#define BUFSIZE 1024
#define PORT 5000
#define INADDR_ANY
#define BACKLOG 5

void servfunc (int sockfd);
```

// continued in the next page.

Teacher's Signature : _____

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name M.C. SOHAN

Dept./Lab _____ Class _____ Expt./No. 1

Title Client-server

```
int main()          //cout...
{
        struct sockaddr_in serv;
        int listenfd, acceptfd, fval;
        bzero(&serv, sizeof(serv));
        serv.sin_family = AF_INET
        serv.sin_port = htons(PORT)
        serv.sin_addr.s_addr = htonl(IP);

        // creating a concurrent server
        listenfd = socket(AF_INET, SOCK_STREAM, 0);
        if(listenfd < 0)
        {
                printf("Unable to create socket");
                exit(0);
        {
        // bind
if( ~~listen~~ bind( listenfd, (struct sockaddr*) &serv,
                                sizeof(serv)) == -1) {
```

Signature of
Teacher incharge

```
        printf("Bind error");
        exit(0);
}
if (listen(listenfd, BACKLOG) < 0)
  {
        printf("Unable to call listen");
          exit(0);
  }
printf("Listen called");
 while (1)
 {
        acceptfd = accept(listenfd, NULL, NULL)
        if (acceptfd == -1)
         {
            printf("Error accepting");
            exit(0);
         }
        printf("Accepted");
        fval = fork();
        if (fval == 0)
        {
              close(listenfd);
              // inside child
              serverfunc(acceptfd);
              close(acceptfd);
               exit(0);
                                    }
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name M. C. SOHAN

Dept./Lab _____ Class _____ Expt./No. _____

Title _____

```
        close (acclphfd);
    }
    return 0;
}
void servfunc (int sockfd)
{
    char buf[BUFSIZE];
    int cnt, fd;
    cnt = recv (sockfd, buf, BUFSIZE, 0);
    if (cnt<0)
    {   printf("error");
        exit(0);
    }
    else if (cnt==0)
    {   printf ("No request");
        exit(0);
    }
```

Signature of
Teacher incharge

```
// write the name of the file
// open the file
// read the contents & send them to the cl
write (1, buf, cnt);
printf ("\n");
fd = open (buf, O_RDONLY);
if (fd == -1)
{
    sprintf (buf, BUFSIZE, "Unable
            to open requested file");
    send (sockfd, buf, strlen(buf),0
    return 0;
}
// send the file
bzero (buf, BUFSIZE);
while ( cnt = read ( fd, buf, BUFSIZE))
{
        send (sockfd, buf, cnt, 0);
        bzero (buf, BUFSIZE);
}
printf ("file sent\n");
close(fd);
} // end.
```

Date .....

Expt. No. 1 Page No. 2

File client

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

#define BUFSIZE 1024
#define PORT 5500
#define IP "127.0.0.1"

int main (int argc, char *argv[])
{
        struct sockaddr_in serv;
        char buf[BUFSIZE];
        int sockfd, cnt;
        if (argc != 3)
        {
            printf ("Format : ./a.out <IP> <file>\n");
            exit(0);
        }
        bzero (&serv, sizeof (serv));
        serv.sin_family = AF_INET;
        serv.sin_port = htons (PORT); // cont...
```

Teacher's Signature : _____

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name M.C. SOHAN

Dept./Lab _____ Class _____ Expt./No. 4

Title client-server (client cont...)

```
if (inet_pton(AF_INET, IP, (void *)&serv.sin_addr)
        == 0)
{
    printf("Invalid address");
    exit(0);
}
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0)
{
    printf("Cant create socket");
    exit(0);
}
printf("File %s requested \n", argv[2]);
// send(sockfd, argv[2]
if (connect(sockfd, (struct sockaddr *)&serv,
        sizeof(serv)) == -1)
{
    printf("Cannot connect");
    exit(0);
}
```

Signature of
Teacher incharge

```
    send( sockfd, argv[2], strlen(argv[2]), 0);
    while( cnt = recv( sockfd, buf, BUFSIZE, 0) )
        write (1, buf, cnt);
    close (sockfd);
    return 0;
} // end of client code
```

Text file that was used:

The contents of this file will be displayed.
l_1
l_2
End of file - there is a newline after this.