Computer Networks

Project Report

Submitted by: Ali John Naqvi
Rabbiya Naeem
Imaan Shahid

Submitted to: Dr. Umar Farooq
Date : 29th January, 2021
Syndicate: B
Computer Engineering - 41

Project Report

Code

Server.cpp

```
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#include <pthread.h>
#include <iostream>
#include <arpa/inet.h>
#include <fstream>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

#define PORT 8080
#define size 1024

using namespace std;

int clients[3];
char send_filename[100];

void make_file(void *conn) {


        long connfd = (long) conn;      //connection handler
        char fileData[size];            //buffer to store incoming file data.
        char filename[100];
        memset(&filename,0,sizeof(filename));
        read(connfd,&filename,sizeof(filename));
        memset(&send_filename,0,sizeof(send_filename));
        strncpy(send_filename,filename,sizeof(filename));

        cout<<"RECIEVED FILE: "<<filename<<endl;

        //clear buffers
        memset(&fileData, 0, sizeof(fileData));
        //declaring variables
        int file_size = 0, read_size = 0, recv_size = 0, write_size = 0, current_stat = 0;

        //get size of incoming file
        do {
                current_stat = read(connfd, &file_size, sizeof(file_size));
        }
        while (current_stat < 0);

        //creating handle for file
        FILE * fp;
        fp = fopen(filename, "w");      //open file in write/read mode
        if (fp)   {
                while (recv_size < file_size) {
                        //read a chunk of data
                        do {
                                read_size = read(connfd, fileData, sizeof(fileData));
                                //write chunk of data to file
                                write_size = fwrite(fileData, 1, read_size, fp);
                                recv_size += read_size;
                        }
                        while (read_size < 0);
                }

                fclose(fp);     //close file
        } else {
                cout << "ERROR OPENING FILE AT SERVER SIDE" << endl;
        }
}

void send_file(void *conn) {


      char filename[100];
      memset(&filename,0,sizeof(filename));
      strncpy(filename,send_filename,sizeof(send_filename));

        long connfd = (long) conn;
        char buffer[size];
        char file[20] = "file";

        for (int i = 0; i < 2; i++) {
                if (clients[i] == connfd) {
                        continue;
                } else {
                        connfd = clients[i];
                        //replacing connfd with clients[i]
                        write(clients[i], file, strlen(file));
                        usleep(10000);
                        write(clients[i],filename,strlen(filename));
                        usleep(100000);

                        FILE * fp;
                        fp = fopen(filename, "r");

                        if (fp) {
                                fseek(fp, 0, SEEK_END);
                                int file_size = ftell(fp);
                                fseek(fp, 0, SEEK_SET);
                                int stat, read_size, x;

                                //send data size
                                write(clients[i], &file_size, sizeof(int));

                                usleep(100000);

                                //read file into large buffer.
                                memset(&buffer, 0, sizeof(buffer));

                                while (!feof(fp)) {
                                        read_size = fread(buffer, 1, sizeof(buffer) - 1, fp);
                                        stat = write(clients[i], buffer, read_size);
                                        usleep(100);

                                        memset(&buffer, 0, sizeof(buffer));
                                }
                        }

                        fclose(fp);

                }
        }
}

void *handle_recv(void *conn) {

        long connfd = (long) conn;
        char buffer[1024];
```

```
	int n;
	char check[1024];	//checks if incoming message is going to be file or not

	while (true) {
		memset(&check, 0, sizeof(check));
		//recv data to check if file is coming or not
		n = recv(connfd, (char*) &check, sizeof(check), 0);

		if (n <= 0) break;

		//file is key word sent by client in order to start sending file
		if (strcmp(check, "file") == 0) {
			make_file(conn);
			usleep(1000);
			send_file(conn);
		} else {
			//if check does not contain key word 'file' than its a message
			cout << "[Client]: " << check << endl;
			//each client response should be send to all other  clients.

			for (int i = 0; i < 2; i++) {
				if (clients[i] == connfd) continue;
				else send(clients[i], (char*) &check, strlen(check), 0);
			}
		}
	}

	return NULL;
}

void *handle_send(void *conn) {
	char buffer[1024];

	while (true) {
		string data;
		getline(cin, data);
		// clear buffer
		memset(&buffer, 0, sizeof(buffer));
		strcpy(buffer, data.c_str());
		//send_file(conn);
		for (int i = 0; i < 2; i++) {
			send(clients[i], (char*) &buffer, strlen(buffer), 0);

		}
```

```
        }
}

int main(int argc, char const *argv[]) {
        int server_fd;
        intptr_t connfd;
        struct sockaddr_in address, cliaddr;
        int cli_addrlen = sizeof(address);
        char buffer[1024];

        server_fd = socket(AF_INET, SOCK_STREAM, 0);  //CREATES A TCP SOCKET

        if (server_fd <= 0)
        {
                perror("SOCKET CREATION FAILED");
                exit(1);
        }

        address.sin_family = AF_INET;
        address.sin_addr.s_addr = INADDR_ANY;
        address.sin_port = htons(PORT);       //BIND TO PORT 8080

        if (bind(server_fd, (struct sockaddr *) &address, sizeof(address)) < 0)
        {
                perror("BIND FAILED");
                exit(1);
        }

        if (listen(server_fd, 2) < 0)
        {
                perror("LISTEN FAILED");
                exit(1);
        }

        pthread_t send_th;
        pthread_create(&send_th, NULL, handle_send, NULL);

        int i = 0;

        while (true)
        {

                connfd = accept(server_fd, (struct sockaddr *) &cliaddr, (socklen_t*)
&cli_addrlen);
```

```
		if (connfd < 0)
		{
			perror("SOCKET RECVIEVE FAILURE");
			exit(1);
		}

		char *dot_ip = inet_ntoa(cliaddr.sin_addr);
		cout << "NEW CLIENT: IP- " << dot_ip << " PORT-: " << cliaddr.sin_port << endl;

		clients[i] = connfd;

		pthread_t recv_th;
		pthread_create(&recv_th, NULL, handle_recv, (void*) connfd);

		i++;
	}

	return 0;
}
```

Client.cpp

```
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#include <pthread.h>
#include <iostream>
#include <arpa/inet.h>
#include <fstream>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

#define PORT 8080
#define size 1024
#define MAX_SIZE 10240
```

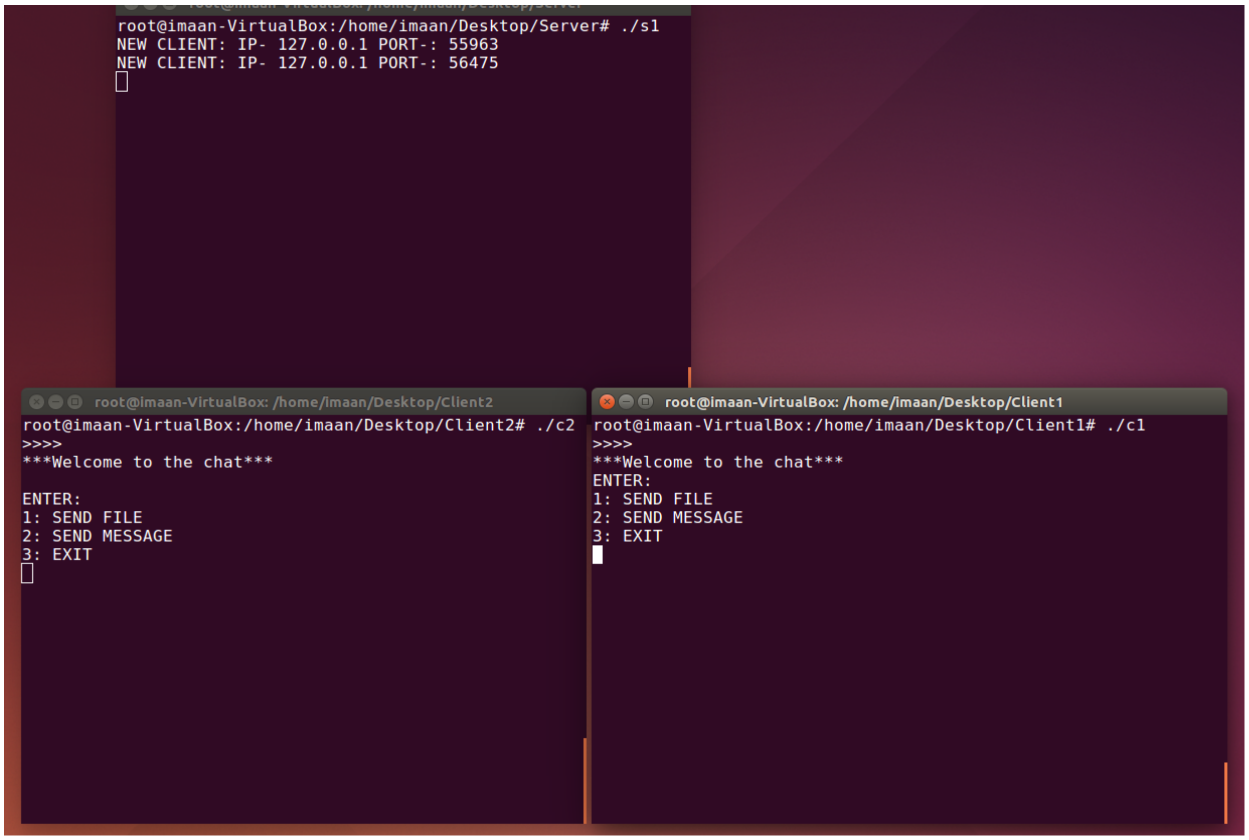
```
using namespace std;


void send_file(void *conn) {
        long sock = (long) conn;
        char file[20] = "file";
        char data_buffer[1024];
        char buffer[MAX_SIZE];
        memset(&data_buffer, 0, sizeof(data_buffer));
        char filename[100];
        send(sock, file, strlen(file), 0);
        cout << "ENTE FILE NAME:" << endl;
        memset(&filename, 0, sizeof(filename));
        scanf("%s", filename);
        write(sock,(void*)&filename,sizeof(filename));
        FILE * fp;
        fp = fopen(filename, "r");

        if (fp)   //if successful open
        {

                fseek(fp, 0, SEEK_END);
                int file_size = ftell(fp);
                fseek(fp, 0, SEEK_SET);
                int stat, read_size, x;
                //send data size
                write(sock, (void*) &file_size, sizeof(int));
                //read file into large buffer.
                memset(&buffer, 0, sizeof(buffer));

                while (!feof(fp))
                {
                        read_size = fread(buffer, 1, sizeof(buffer) - 1, fp);
                        do {
                                stat = write(sock, buffer, read_size);

                        }
                        while (stat < 0);
                        memset(&buffer, 0, sizeof(buffer));
                }
        }
        else
        {
                cout << "CAN NOT OPEN FILE" << endl;
        }
        fclose(fp);

}

void recv_file(void *conn)
{

        char filename[100];
        memset(&filename,0,sizeof(filename));
        long connfd = (long) conn;
        //get file name:
        read(connfd,filename,sizeof(filename));
        cout<<"RECIEVED FILE : "<<filename<<endl;
        char fileData[size];    //buffer to store incoming file data.
        //clear buffers
        memset(&fileData, 0, sizeof(fileData));
        //declaring variables
        int file_size = 0, read_size = 0, recv_size = 0, write_size = 0, current_stat = 0;
        //get size of incoming file
        //do
        //{

        current_stat = read(connfd, (int*) &file_size, sizeof(int));
        usleep(1000);
        //}
        //while(current_stat < 0);


        //creating handle for file
        FILE * fp;
        fp = fopen(filename, "w");      //open file in write/read mode
        if (fp)   //successful opening of file
        {

                while (recv_size < file_size)
                {
                        do {    //read a chunk of data
                                read_size = read(connfd, fileData, sizeof(fileData));
                                usleep(100);
                                //write chunk of data to file
                                write_size = fwrite(fileData, 1, read_size, fp);
                                recv_size += read_size;
                        }
                while (read_size <= 0);
            }

            fclose(fp);     //close file
        }
        else
        {
            cout << "ERROR OPENING FILE AT CLIENT SIDE" << endl;
        }
}

void *handle_recv(void *conn)
{

        long connfd = (long) conn;
        int n;
        char check[1024];
        while (true)
        {
            memset(&check, 0, sizeof(check));
            n = recv(connfd, (char*) &check, sizeof(check), 0);
            if (n <= 0) break;
            usleep(1000);
            check[n] = '\0';

            if (strncmp(check, "file", 4) == 0)
            {

                recv_file((void*) conn);
            }
            else
            {
                // print server response
                cout << "[SERVER]: " << check << endl;
            }
        }

        return NULL;
}

int main(int argc, char const *argv[])
{
        intptr_t sock;
        struct sockaddr_in serv_addr;
```

```
if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
{
        perror("SOCKET CREATION FAILED\n");
        exit(1);
}

serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(PORT);

if (inet_aton("127.0.0.1", &serv_addr.sin_addr) <= 0)
{
        printf("INVALID ADDRESS\n");
        exit(1);
}

if (connect(sock, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0)
{
        printf("CONNECTION FAILED\n");
        exit(1);
}

pthread_t th;
pthread_create(&th, NULL, handle_recv, (void*) sock);

cout << ">>>>" << endl;

int choice = 1;

while (choice != 3)
{

        int num;
        cout<< "***Welcome to the chat***"<<endl;
        cout<<"ENTER:"<<endl;
        cout<<"1: SEND FILE"<<endl;
        cout<<"2: SEND MESSAGE "<<endl;
        cout<<"3: EXIT "<<endl;
        cin >> num;
        choice = num;

        switch (num)
        {
                case 1:
```

```
                                send_file((void*) sock);
                                break;
                        case 2:
                        {
                                char buffer[1024];
                                cout << "ENTER MESSAGE: " << endl;
                                string data;
                                //getline(cin, data,'\0');
                                cin >> data;
                                memset(&buffer, 0, sizeof(buffer));
                                strcpy(buffer, data.c_str());
                                send(sock, (char*) &buffer, strlen(buffer), 0);
                            break;
                      }
                        case 3:
                                 cout<<"EXITING"<<endl;
                                 break;
                }
        }

        return 0;
}
```

Output

1) Multiple Clients connecting to server:



```
root@imaan-VirtualBox:/home/imaan/Desktop/Server# ./s1
NEW CLIENT: IP- 127.0.0.1 PORT-: 55963
NEW CLIENT: IP- 127.0.0.1 PORT-: 56475
```

root@imaan-VirtualBox: /home/imaan/Desktop/Client2

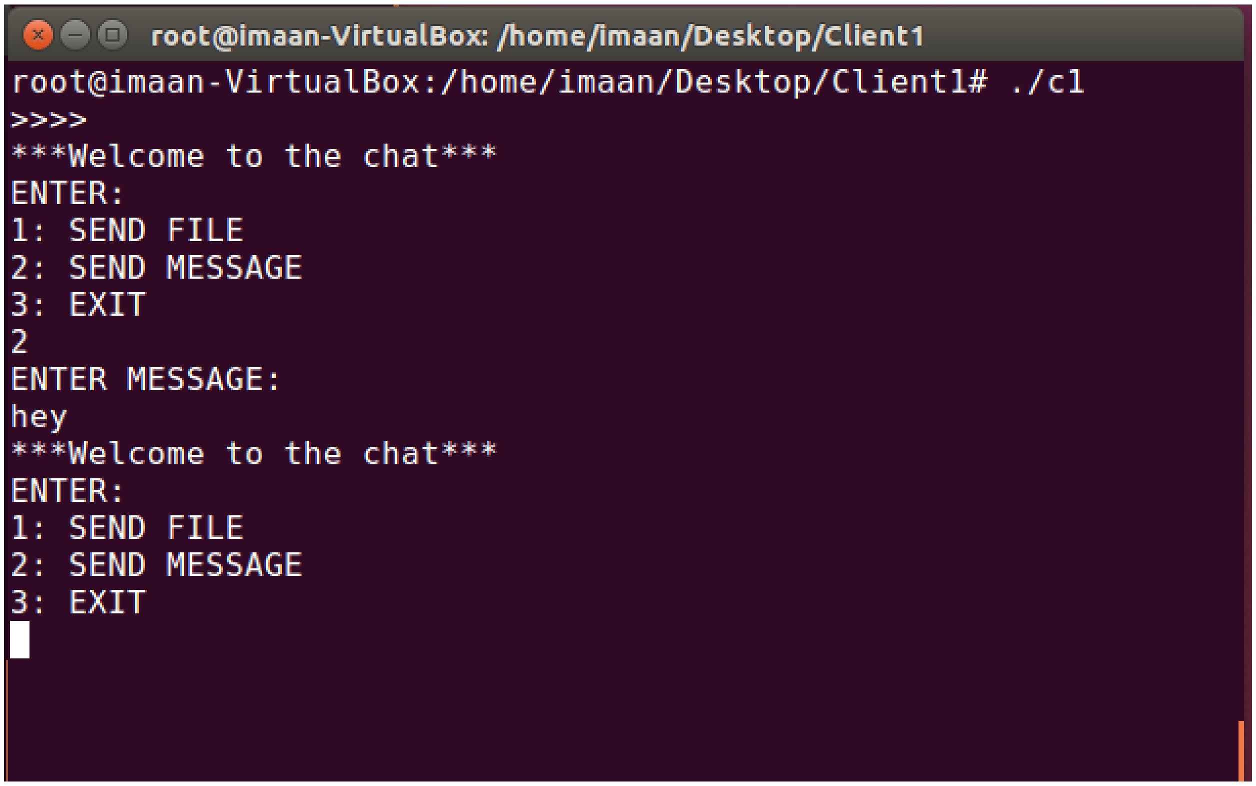
```
root@imaan-VirtualBox:/home/imaan/Desktop/Client2# ./c2
>>>>
***Welcome to the chat***

ENTER:
1: SEND FILE
2: SEND MESSAGE
3: EXIT
```

root@imaan-VirtualBox: /home/imaan/Desktop/Client1

```
root@imaan-VirtualBox:/home/imaan/Desktop/Client1# ./c1
>>>>
***Welcome to the chat***
ENTER:
1: SEND FILE
2: SEND MESSAGE
3: EXIT
```

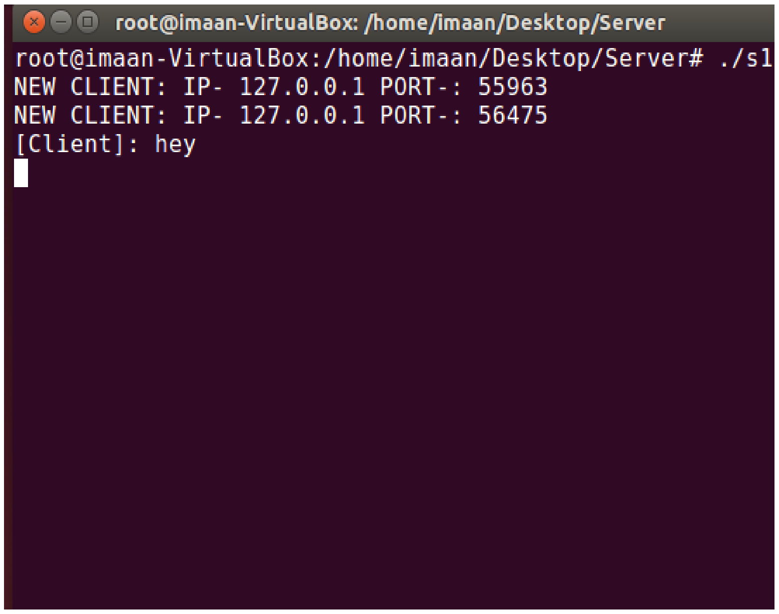
2) **<u>Sending Message from client, received to server and other client</u>**

<u>-Client 1: sending message</u>

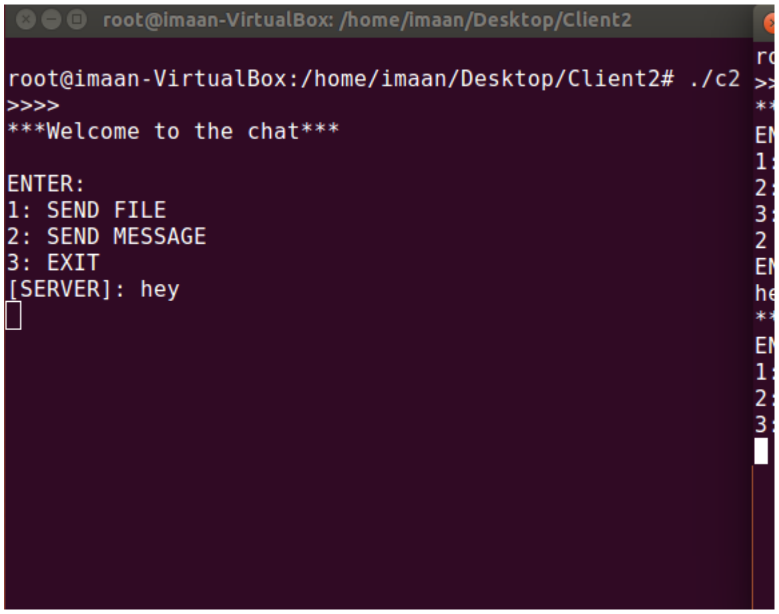


```
root@imaan-VirtualBox: /home/imaan/Desktop/Client1
root@imaan-VirtualBox:/home/imaan/Desktop/Client1# ./c1
>>>>
***Welcome to the chat***
ENTER:
1: SEND FILE
2: SEND MESSAGE
3: EXIT
2
ENTER MESSAGE:
hey
***Welcome to the chat***
ENTER:
1: SEND FILE
2: SEND MESSAGE
3: EXIT
```

<u>-Server & Client 2: receiving message</u>



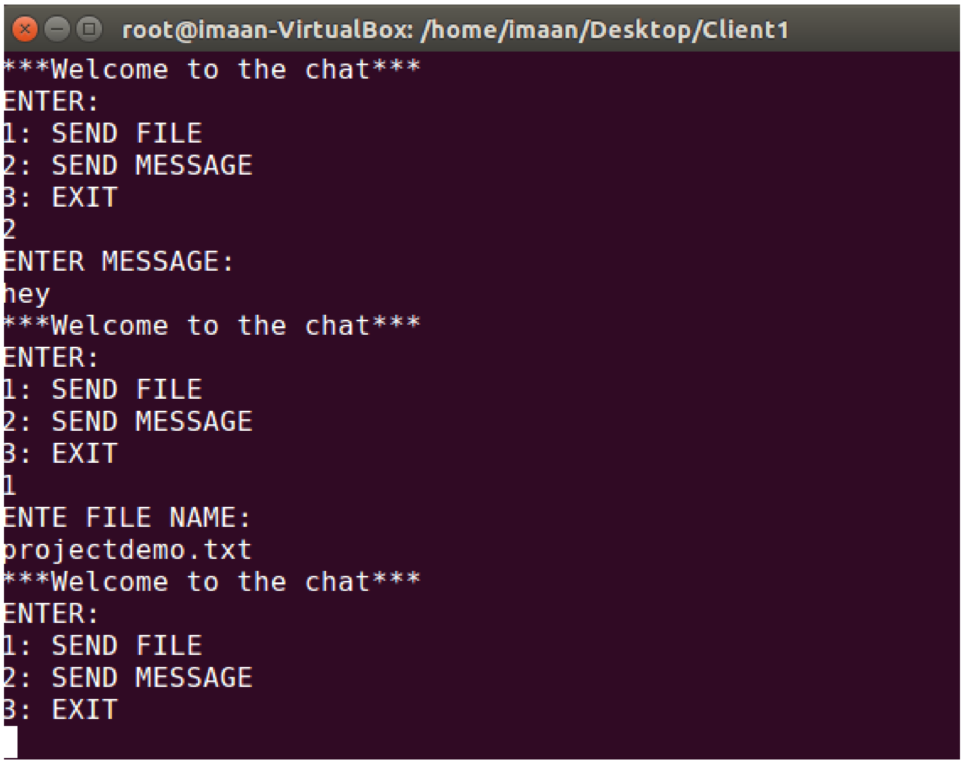
```
root@imaan-VirtualBox: /home/imaan/Desktop/Server
root@imaan-VirtualBox:/home/imaan/Desktop/Server# ./s1
NEW CLIENT: IP- 127.0.0.1 PORT-: 55963
NEW CLIENT: IP- 127.0.0.1 PORT-: 56475
[Client]: hey
```



```
root@imaan-VirtualBox: /home/imaan/Desktop/Client2

root@imaan-VirtualBox:/home/imaan/Desktop/Client2# ./c2
>>>>
***Welcome to the chat***

ENTER:
1: SEND FILE
2: SEND MESSAGE
3: EXIT
[SERVER]: hey
```

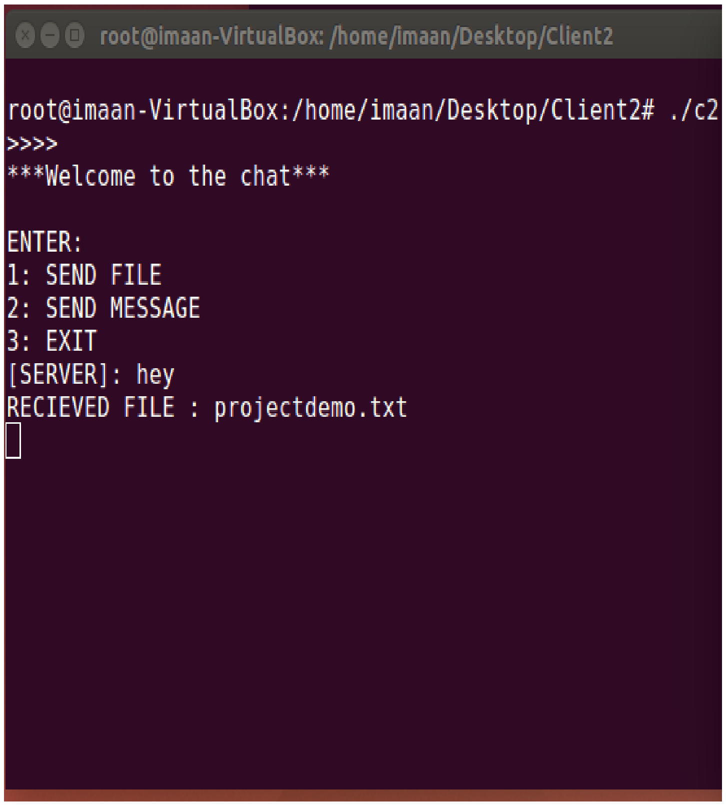
3) **Sending file from client, received to server and other client**

-Client 1: sending file

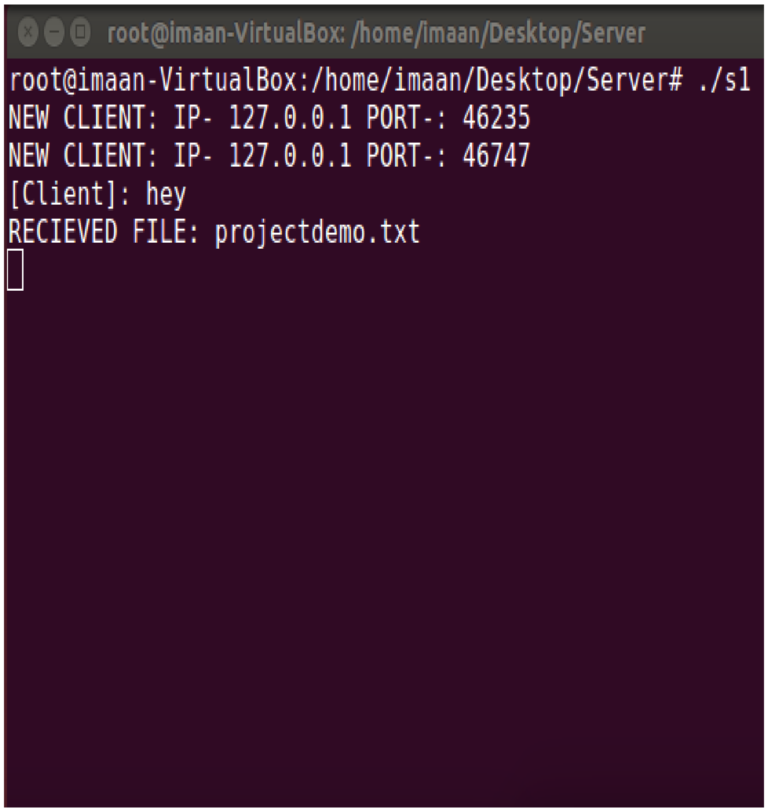


```
***Welcome to the chat***
ENTER:
1: SEND FILE
2: SEND MESSAGE
3: EXIT
2
ENTER MESSAGE:
hey
***Welcome to the chat***
ENTER:
1: SEND FILE
2: SEND MESSAGE
3: EXIT
1
ENTE FILE NAME:
projectdemo.txt
***Welcome to the chat***
ENTER:
1: SEND FILE
2: SEND MESSAGE
3: EXIT
```

-Server & Client 2: receiving file

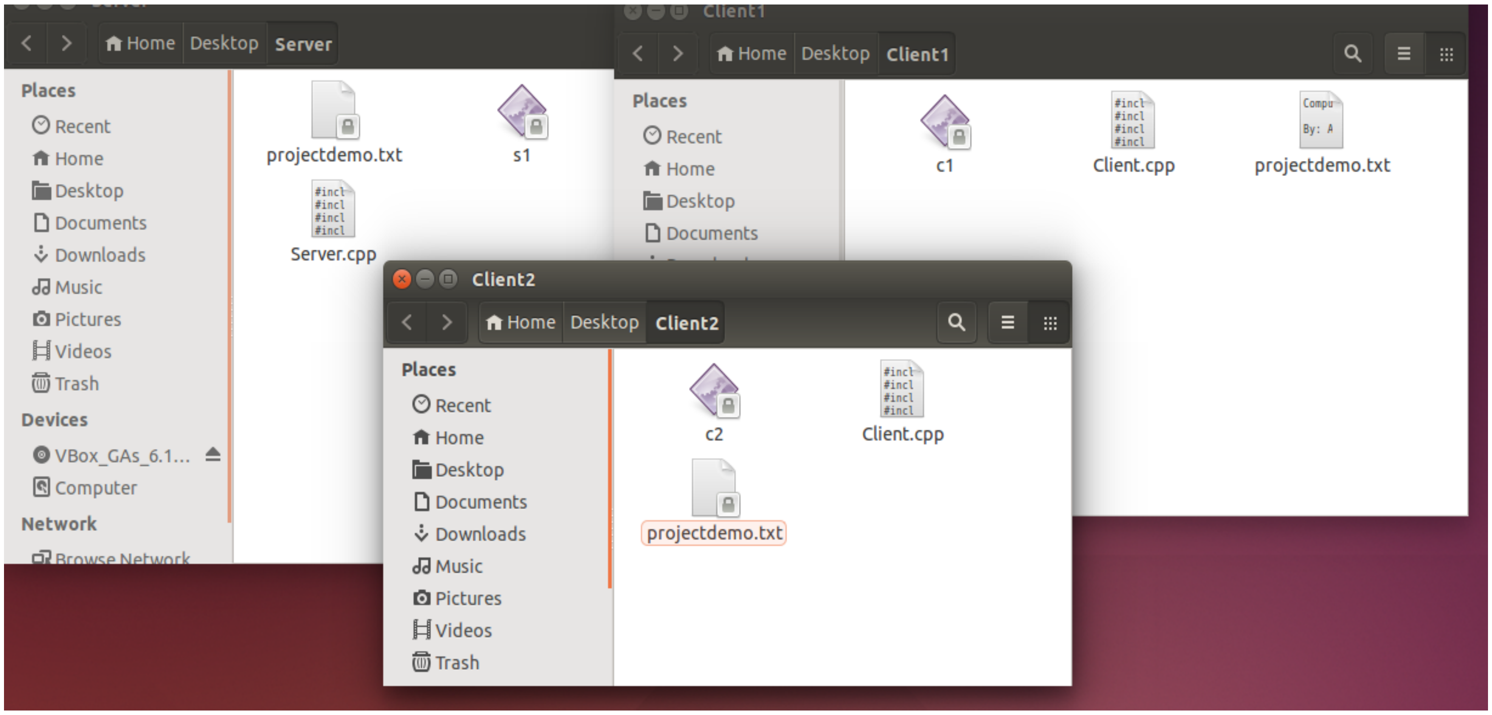


```
root@imaan-VirtualBox:/home/imaan/Desktop/Client2# ./c2
>>>>
***Welcome to the chat***

ENTER:
1: SEND FILE
2: SEND MESSAGE
3: EXIT
[SERVER]: hey
RECIEVED FILE : projectdemo.txt
```

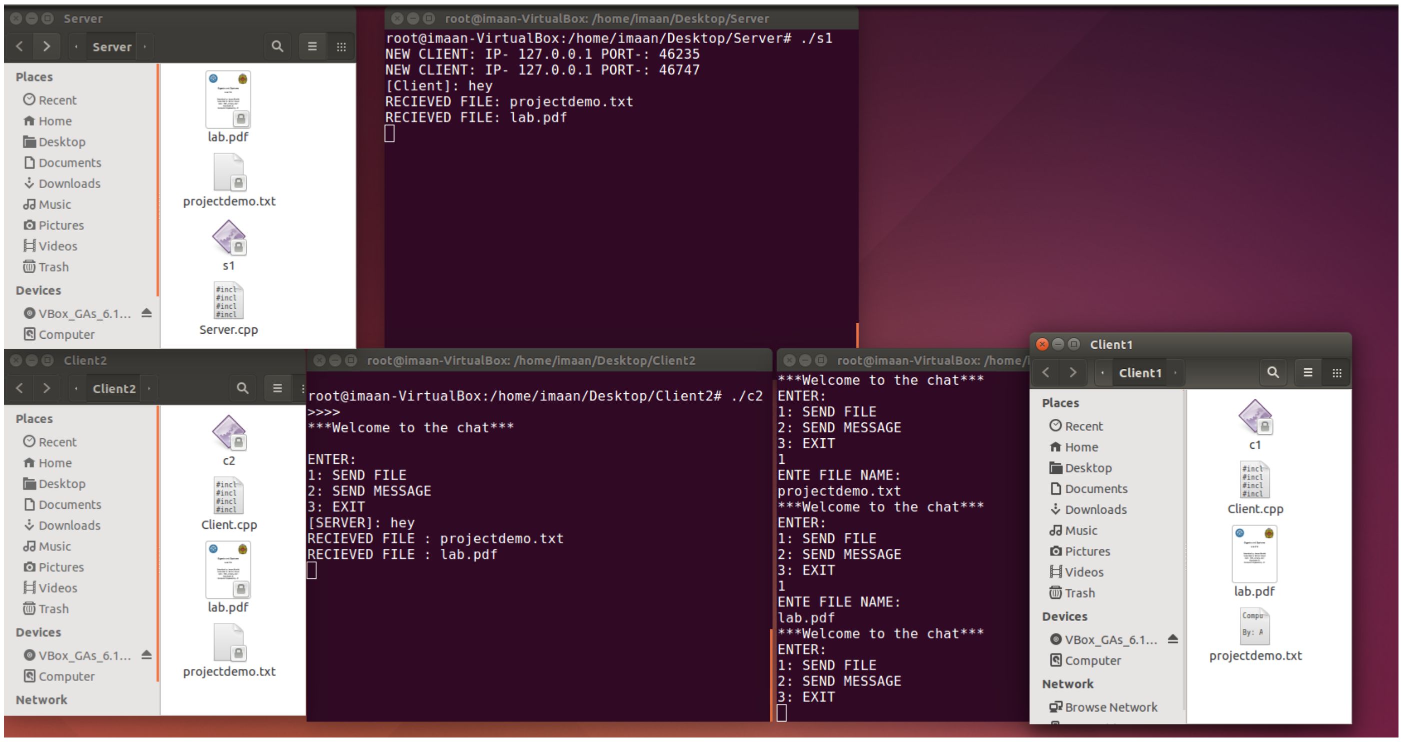


```
root@imaan-VirtualBox:/home/imaan/Desktop/Server# ./s1
NEW CLIENT: IP- 127.0.0.1 PORT-: 46235
NEW CLIENT: IP- 127.0.0.1 PORT-: 46747
[Client]: hey
RECIEVED FILE: projectdemo.txt
```

-Transferred file in respective directories:



4) Sending PDF from client, received to server and other client



```
root@imaan-VirtualBox:/home/imaan/Desktop/Server# ./s1
NEW CLIENT: IP- 127.0.0.1 PORT-: 46235
NEW CLIENT: IP- 127.0.0.1 PORT-: 46747
[Client]: hey
RECIEVED FILE: projectdemo.txt
RECIEVED FILE: lab.pdf
```

```
root@imaan-VirtualBox:/home/imaan/Desktop/Client2# ./c2
>>>>
***Welcome to the chat***

ENTER:
1: SEND FILE
2: SEND MESSAGE
3: EXIT
[SERVER]: hey
RECIEVED FILE : projectdemo.txt
RECIEVED FILE : lab.pdf
```

```
***Welcome to the chat***
ENTER:
1: SEND FILE
2: SEND MESSAGE
3: EXIT
1
ENTE FILE NAME:
projectdemo.txt
***Welcome to the chat***
ENTER:
1: SEND FILE
2: SEND MESSAGE
3: EXIT
1
ENTE FILE NAME:
lab.pdf
***Welcome to the chat***
ENTER:
1: SEND FILE
2: SEND MESSAGE
3: EXIT
```