NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR

Cachar, Assam

B.Tech. Vth Sem

Subject Code: CS-311

Subject Name: Computer Network Laboratory

Submitted By:

Name : Subhojit Ghimire

Sch. Id. : 1912160

Branch : CSE – B

Q.6. Write a Client-Server socket program to implement "FTP Server" using UDP connection. (Description : The file.txt may be there on the server side with some text; on the client request, the server has to send the text file file.txt and corresponding text has to print on client terminal.)

AIM: TO IMPLEMENT "FTP SERVER" USING UDP CONNECTION IN CPP.

THEORY: 1. UDP CLIENT SERVER: In UDP, the client does not form a connection with the server like in TCP. Instead, the client just sends a datagram. Similarly, the server does not need to accept a connection and just waits and scans for datagrams to arrive. Datagrams, upon arrival, contain the address of the sender, along with the message, which the server uses to send data to the correct client.

2. FTP : A File Transfer Protocol (FTP) client is a software utility that establishes a connection between a host computer and a remote server, typically an FTP server. FTP refers to a group of rules that govern how computers transfer files from one system to another over the internet. FTP protocol uses TCP protocol for client server communication. However, there is another protocol called Trivial File Transfer Protocol (TFTP) which uses UDP protocol to transfer files. TFTP has minimal features and doesn't have authentication.

3. FTP SERVER: It holds the files and databases that are required to provide the services requested by clients.

4. FTP CLIENT: It is generally a personal computer used by an end user or a mobile device which is running the necessary software that is capable of requesting and receiving files over the internet from a FTP server.

The entire operation can be broken down as follows:

FTP CLIENT:

(i) A socket is created and binded.

(ii) The file name, whose content is to be read / retrieved, is entered as an input and sent from the user using sendto().

(iii) The content of the file whose name was sent is received using recvfrom() function and displayed.

FTP SERVER:

(i) A socket is created and binded to an advertised port number.

(ii) An infinite loop is started to process the client requests for connections.

(iii) The process receives a filename from the client using recvfrom() function and reads content from the file, and sends the content back to the client using sendto() function.

CODE :

```
// FTP UDP SERVER

# include <iostream>
# include <cstdlib>
# include <unistd.h>
# include <cstring>
# include <sys/types.h>
# include <sys/socket.h>
# include <arpa/inet.h>
# include <netinet/in.h>
# include <fstream>

# define PORT 8080
# define MAXLINE 1024
char fileContent [MAXLINE];

using namespace std;

int ftpfileProcess (char *fileName) {
        FILE *ff = fopen (fileName, "r");
        if (ff == NULL)
                return 0;
        [redacted]
        int ii = 0;
        while (!feof (ff))
                fileContent [ii++] = getc (ff);
        fileContent [ii-1] = '\0';
```

```
    if (fileContent[0] == '\0')
        return 0;
    return 1;
}

int main () {
    int sockfd;
    Struct sockaddr_in servaddr, cliaddr;
    if ((sockfd = socket (AF_INET, SOCK_DGRAM, 0)) < 0) {
            perror ("FAIL: SOCKET CREATION\n");
            exit (EXIT_FAILURE);
    }
    cout << " SUCCESS: SOCKET CREATED \n";

    memset (&servaddr, 0, sizeof (servaddr));
    memset (&cliaddr, 0, sizeof (cliaddr));

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons (PORT);

    if (bind (sockfd, (const struct sockaddr *) &servaddr,
                    sizeof (servaddr)) < 0 ) {
        perror ("FAIL: SERVER BIND \n");
        exit (EXIT_FAILURE);
    }
    Cout << " SUCCESS: SERVER BOUND \n";

    cout << " SERVER LISTENING FOR INCOMING MESSAGES...";
```

```
cout << endl <<endl;
char buffer[MAXLINE];
unsigned int len, nn;
len = sizeof(cliaddr);
while (1) {
        memset (buffer, 0, MAXLINE);
        bzero (fileContent, MAXLINE);
        nn = recvfrom (sockfd, (char *) buffer, MAXLINE,
             MSG_WAITALL, (struct sockaddr *) &cliaddr, &len);
        buffer[nn] = '\0';
        cout << "REQUEST RECEIVED : OPENING FILE "<<buffer;
        buffer[nn-1] = '\0';
        if (ftpFileProcess (buffer)) {
            memset (buffer, 0, MAXLINE);
            strcpy (buffer, fileContent);
        }
        else {
            memset (buffer, 0, MAXLINE);
            strcpy (buffer, "ERROR: FILE EITHER
                    EMPTY OR DOES NOT EXIST");
        }
        sendto (sockfd, (const char *) buffer, strlen(buffer),
             MSG_CONFIRM, (const struct sockaddr *)
             &cliaddr, len);
}
close (sockfd);
return 0;
}
```

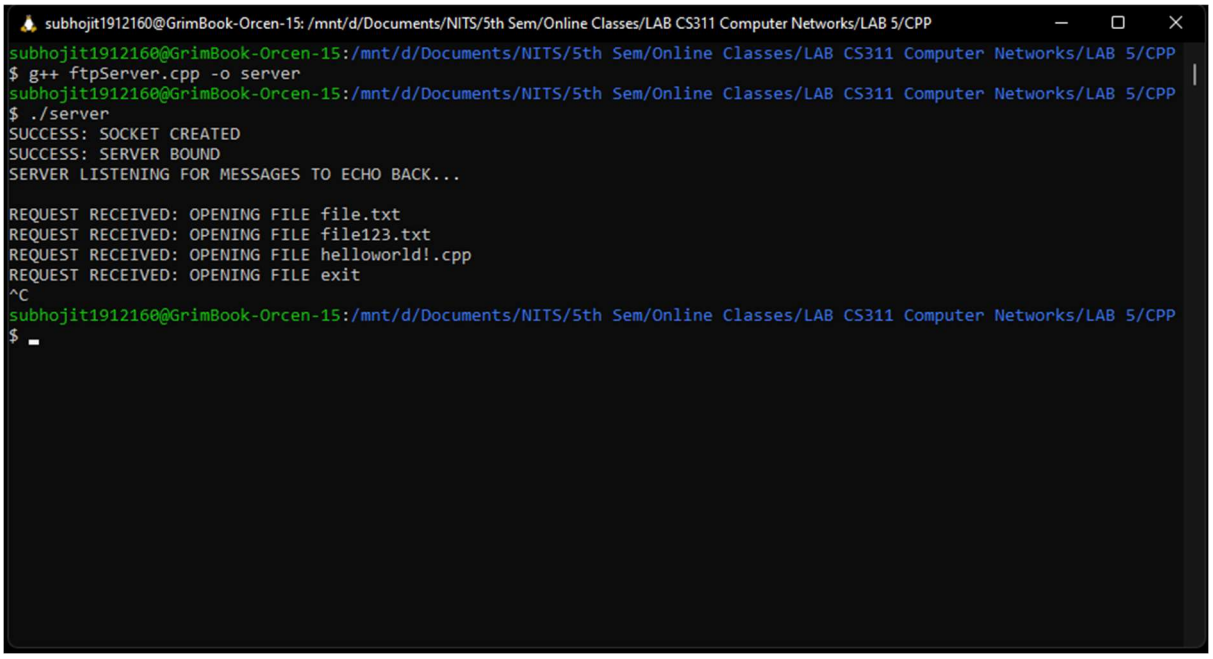
// FTP UDP CLIENT

```
# include <iostream>
# include <cstdlib>
# include <unistd.h>
# include <cstring>
# include <sys/types.h>
# include <sys/socket.h>
# include <arpa/inet.h>
# include <netinet/in.h>

# define PORT 8080
# define MAXLINE 1024

using namespace std;

int main() {
    int sockfd;
    struct sockaddr_in servaddr;
    if ((sockfd = socket (AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror ("FAIL : SOCKET CREATION \n");
        exit (EXIT_FAILURE);
    }
    cout << " SUCCESS : SOCKET CREATED \n\n ";
    memset (&servaddr, 0, sizeof (servaddr));

    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons (PORT);
    servaddr.sin_addr.s_addr = INADDR_ANY;
```

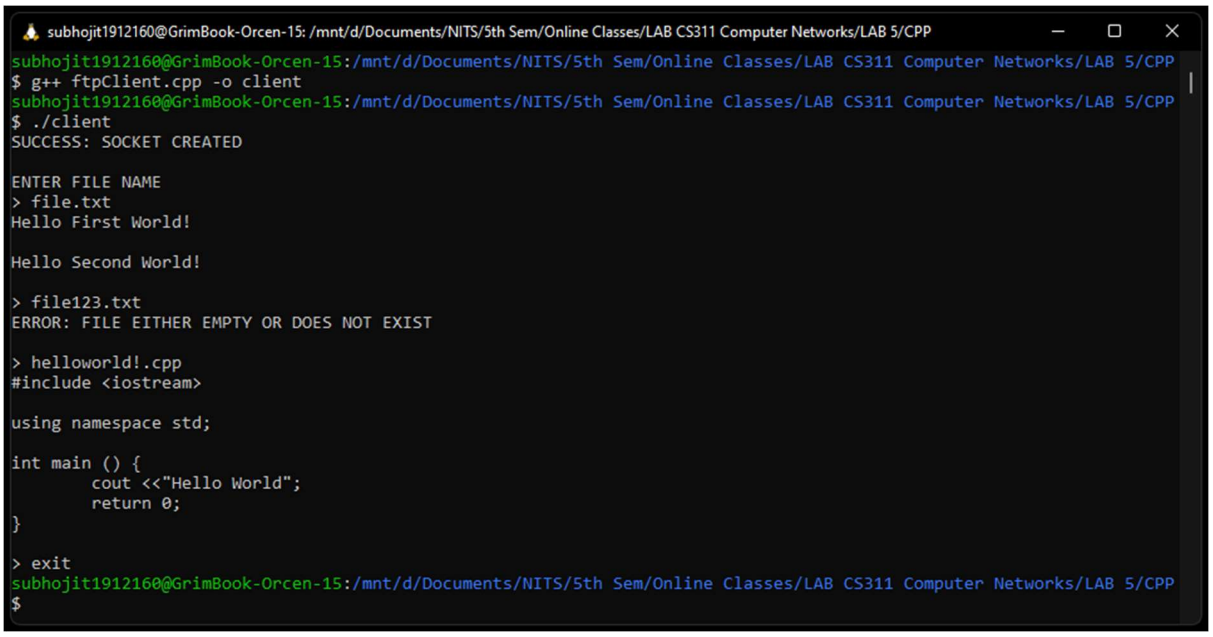
```
char buffer [MAXLINE];
char fileName [MAXLINE];
cout << "ENTER FILE NAME : " ;
unsigned int nn, len;

while (1) {
    memset (fileName, 0, MAXLINE);
    memset (buffer, 0, MAXLINE);
    cout << "\n>  " ;
    fgets (fileName, MAXLINE, stdin);

    sendto (sockfd, (const char *) fileName, strlen (fileName),
            MSG-CONFIRM, (const struct sockaddr *) &servaddr,
            sizeof (servaddr));
    nn = recvfrom (sockfd, (char *) buffer, MAXLINE,
            MSG-WAITALL, (struct sockaddr *) &servaddr,
            &len);
    buffer[nn] = '\0';
    if (strncmp (fileName, "exit", 4) == 0)
            break;
    cout << buffer << endl;
}

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

OUTPUT:

// FTP SERVER



// FTP CLIENT



Output Explanation:

Firstly, two terminals are opened and FTP client and server programmes are compiled and run. The client terminal asks the user to enter file name for the file the user is looking to read. Once the file name is entered, the file name is sent as a message from client to server. The server accepts the file name as the buffer value and searches for the file in its database with the same name as the buffer message. If the file is found, then the server reads and stores the content of that file in a string and sends it back to the client, which is then displayed on the client terminal. If the file is not found, or the file is found but it is empty (i.e., even if the file is found but the content is not found), in such cases, server sends back an error message which is received by the client and is displayed on the client terminal. After the client is done with the interaction, the client will enter an “exit” command or message to terminate the connection.