Experiment - 8

Roll No. 9/793031
CN Lab

Experiment - 08

Title - Implement a file transfer protocol for two operations, send and receive over Ip and port 21

Objective - Study for Implement a file transfer protocol for two operations send and receive over Ip and port 21

Theory -

The file transfer protocol (Ftp) is a standard communication protocol used for the transfer of computer files from a server to a client on a computer network. FTP is built on a client-server model architecture using separate control and data connections between the client and server.

FTP users may authenticate themselves with a clear-text sign in protocol, normally in the form of a username and password, but can connect anonymously if the server is configured to allow it.

For secure transition that protects the username and password and encrypts the connection. FTP is often secured with SSL/TLS (FTPS) or replaced with SSH File Transfer Protocol (SFTP).

The first FTP client applications were command-line programs developed before operating systems had graphical user interfaces, and are still shipped with most Windows, Unix and Linux operating systems.

Algorithm:

1. Execute the server side and wait for the connection request from client
2. Accept the connection request from client
3. Open the given file in read mode
4. Get the file name from client
5. Find the file contents to the client.

Teacher's Signature _____

⑥ At the client side, create a new file & open it in write mode

⑦ write the received data in it.

Conclusion -

In this experiment we can learn that implement a file transfer protocol for two operation, send and receive over Ip and port 21

Program-

Client:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define SIZE 1024


void send_file_data(FILE* fp, int sockfd, struct
sockaddr_in addr)

{

    int n;

    char buffer[SIZE];


    // Sending the data

    while (fgets(buffer, SIZE, fp) != NULL)

    {
```

```
        printf("[SENDING] Data: %s", buffer);


        n = sendto(sockfd, buffer, SIZE, 0, (struct
sockaddr*)&addr, sizeof(addr));
        if (n == -1)
        {
            perror("[ERROR] sending data to the server.");
            exit(1);
        }
        bzero(buffer, SIZE);
    }


    // Sending the 'END'
    strcpy(buffer, "END");
    sendto(sockfd, buffer, SIZE, 0, (struct
sockaddr*)&addr, sizeof(addr));
```

```
    fclose(fp);
}

int main(void)
{

    // Defining the IP and Port
    char *ip = "127.0.0.1";
    const int port = 21;

    // Defining variables
    int server_sockfd;
    struct sockaddr_in server_addr;
    char *filename = "client.txt";
    FILE *fp = fopen(filename, "r");

    // Creating a UDP socket
```

```
server_sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (server_sockfd < 0)
{
    perror("[ERROR] socket error");
    exit(1);
}
server_addr.sin_family = AF_INET;
server_addr.sin_port = port;
server_addr.sin_addr.s_addr = inet_addr(ip);


// Reading the text file
if (fp == NULL)
{
    perror("[ERROR] reading the file");
    exit(1);
}
```

```
    // Sending the file data to the server
    send_file_data(fp, server_sockfd, server_addr);

    printf("[SUCCESS] Data transfer complete.\n");
    printf("[CLOSING] Disconnecting from the server.\n");

    close(server_sockfd);

    return 0;
}
```

Server:

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

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

#define SIZE 1024

void write_file(int sockfd, struct sockaddr_in addr)
{

    char* filename = "server.txt";
    int n;
    char buffer[SIZE];
    socklen_t addr_size;

    // Creating a file.
    FILE* fp = fp = fopen(filename, "w");

    // Receiving the data and writing it into the file.
    while (1)
```

```
    {
        addr_size = sizeof(addr);
        n = recvfrom(sockfd, buffer, SIZE, 0, (struct
sockaddr*)&addr, &addr_size);

        if (strcmp(buffer, "END") == 0)
        {
            break;
        }

        printf("[RECEVING] Data: %s", buffer);
        fprintf(fp, "%s", buffer);
        bzero(buffer, SIZE);
    }

    fclose(fp);
}
```

```
int main()
{

    // Defining the IP and Port
    char* ip = "127.0.0.1";
    const int port = 21;

    // Defining variables
    int server_sockfd;
    struct sockaddr_in server_addr, client_addr;
    char buffer[SIZE];
    int e;

    // Creating a UDP socket
    server_sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (server_sockfd < 0)
```

```
    {
        perror("[ERROR] socket error");
        exit(1);
    }
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = port;
    server_addr.sin_addr.s_addr = inet_addr(ip);

    e = bind(server_sockfd, (struct
sockaddr*)&server_addr, sizeof(server_addr));
    if (e < 0)
    {
        perror("[ERROR] bind error");
        exit(1);
    }

    printf("[STARTING] UDP File Server started. \n");
```

```
    write_file(server_sockfd, client_addr);


    printf("[SUCCESS] Data transfer complete.\n");
    printf("[CLOSING] Closing the server.\n");


    close(server_sockfd);


    return 0;
}
```