**26/10/2021 CN SPOT 8 2019103573**

**FILE TRANSFER PROTOCOL (FTP)**

**SERVER.C**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <arpa/inet.h>

#define CONTROLPORT 2800

#define SERVERPORT 2500

int listeningSocketConnection(long port)

{

    int socketfd;

    struct sockaddr\_in server\_addr;

    socketfd = socket(AF\_INET, SOCK\_STREAM, 0);

    if (socketfd < 0)

    {

        fprintf(stderr, "ERROR IN SOCKET CREATION");

        return -1;

    }

    server\_addr.sin\_family = AF\_INET;

    server\_addr.sin\_port = htons(port);

    inet\_pton(AF\_INET, "127.0.0.1", &server\_addr.sin\_addr);

    if (bind(socketfd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0)

    {

        fprintf(stderr, "ERROR IN BIND CREATION");

        return -1;

    }

    if (listen(socketfd, 5) < 0)

    {

        fprintf(stdout, "ERROR IN LISTEN CREATION");

        return -1;

    }

    fprintf(stdout, "LISTENING AT %s : %d \n", inet\_ntoa(server\_addr.sin\_addr),

            ntohs(server\_addr.sin\_port));

    return socketfd;

}

void loading()

{

    for (int k = 0; k < 10000000; k++)

    {

        int j = k;

    }

}

void file\_transfer(int client)

{

    int socketfd, clientfd, length = 0;

    struct sockaddr\_in server\_addr, client\_addr;

    char buffer[1024], filename[1024];

    const char \*port = "2500";

    socketfd = listeningSocketConnection(SERVERPORT);

    if (socketfd < 0)

        return;

    send(client, port, strlen(port) + 1, 0);

    clientfd = accept(socketfd, (struct sockaddr \*)&client\_addr, &length);

    if (clientfd < 0)

    {

        fprintf(stderr, "Error in accepting connection.\n");

        close(socketfd);

        return;

    }

    fprintf(stdout, "New file trasfer connection established.\n");

    while (1)

    {

        printf("\n---------------------------------------------------------\n");

        recv(clientfd, filename, sizeof(filename), 0);

        if (strncmp(filename, "exit", strlen("exit")) == 0)

        {

            fprintf(stdout, "Client exiting file\_transfer.\n");

            break;

        }

        fprintf(stdout, "\n[-] Requested file : %s\n", filename);

        int fd = open(filename, O\_RDONLY);

        if (fd < 0)

        {

            send(clientfd, "404", strlen("404") + 1, 0);

            fprintf(stdout, "Requested file not found.\n");

            continue;

        }

        send(clientfd, "FOUND", strlen("FOUND") + 1, 0);

        fprintf(stdout, "Requested file found.\n");

        read(fd, buffer, sizeof(buffer));

        send(clientfd, buffer, sizeof(buffer) + 1, 0);

        fprintf(stdout, "FILE SENT SUCCESSFULLY\n");

    }

    close(clientfd);

    close(socketfd);

    fprintf(stdout, "Closing the file-transfer connection.\n\n");

    return;

}

void authorize\_and\_handle(int clientfd)

{

    char buffer[1024], linebuffer[1024];

    char username[1024], password[1024];

    char \*user, \*pass;

    int n, i, j, flag = 0;

AUTHORIZE:

    n = recv(clientfd, buffer, sizeof(buffer), 0);

    if (strncmp(buffer, "auth$", sizeof("auth$")) == 0)

    {

        send(clientfd, "INVALID REQUEST", sizeof("INVALID REQUEST") + 1, 0);

        return;

    }

    i = 5;

    j = 0;

    while (buffer[i] != '$' && buffer[i] != '\n' && buffer[i] != '\0')

    {

        username[j] = buffer[i];

        j++;

        i++;

    }

    username[j] = '\0';

    i++;

    j = 0;

    while (buffer[i] != '$' && buffer[i] != '\n' && buffer[i] != '\0')

    {

        password[j] = buffer[i];

        j++;

        i++;

    }

    password[j] = '\0';

    printf("\n---------------------------------------------------\n");

    fprintf(stdout, " [-] REQUEST FOR CLIENT's USERNAME : %s\n\tCLIENT's PASSWORD %s\n",

username, passsword);

    printf("\nCHECKING");

    for (int k = 0; k < 8; k++)

    {

        loading();

        printf(".");

    }

    printf("\n");

    FILE \*fd = fopen("authorize.txt", "r");

if (fd < 0)

    {

        fprintf(stderr, "Error in opening auth file.\n");

        send(clientfd, "ERROR", strlen("ERROR") + 1, 0);

        return;

    }

    while (fgets(linebuffer, sizeof(linebuffer), fd))

    {

        user = strtok(linebuffer, "$");

        pass = strtok(NULL, "\n");

        if (strcmp(username, user) == 0 && strcmp(password, pass) == 0)

        {

            file\_transfer(clientfd);

            flag = 1;

            break;

        }

    }

    if (flag == 0)

    {

        fprintf(stdout, "Invalid Access Denied.\n");

        send(clientfd, "INVALID\_CRED", strlen("INVALID\_CRED") + 1, 0);

        goto AUTHORIZE;

    }

    return;

}

int main()

{

    int socketfd, clientfd, length = 0;

    struct sockaddr\_in client\_addr;

    socketfd = listeningSocketConnection(CONTROLPORT);

    if (socketfd < 0)

    {

        return -1;

    }

    while (1)

    {

        clientfd = accept(socketfd, (struct sockaddr \*)&client\_addr, &length);

        if (clientfd < 0)

        {

            fprintf(stderr, "ERROR IN ACCEPTING CONNECTION\n");

            continue;

        }

        fprintf(stdout, "NEW CONTROL CONNECTION ESTABLISHED\n");

        authorize\_and\_handle(clientfd);

        close(clientfd);

    }

    close(socketfd);

    return 0;

}

**CLIENT.C**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/socket.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#define SERVERPORT 2800

int getConnectedSocket(long port)

{

    int socketfd = 0, n = 0;

    struct sockaddr\_in server\_addr;

    socketfd = socket(AF\_INET, SOCK\_STREAM, 0);

    if (socketfd < 0)

    {

        fprintf(stderr, "Error in socket creation.\n");

        return -1;

    }

    server\_addr.sin\_family = AF\_INET;

    server\_addr.sin\_port = htons(port);

    inet\_pton(AF\_INET, "127.0.0.1", &server\_addr.sin\_addr);

    if (connect(socketfd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0)

    {

        fprintf(stderr, "Error in connection.\n");

        return -1;

    }

    fprintf(stdout, "Connection established with %s : %d\n", inet\_ntoa(server\_addr.sin\_addr), ntohs(server\_addr.sin\_port));

    return socketfd;

}

void file\_transfer(int file\_port)

{

    int socketfd = 0;

    char filename[1024], buffer[1024];

    filename[0] = '\0';

    char cnfm;

    socketfd = getConnectedSocket(file\_port);

    if (socketfd < 0)

        return;

    fprintf(stdout, "Enter exit to close the connection.\n");

    while (1)

    {

        fprintf(stdout, "\n---------------------------------------------\n");

        fprintf(stdout, "[-] Enter filename : ");

        scanf("%s", filename);

        if (strncmp(filename, "exit", strlen("exit")) == 0)

        {

            send(socketfd, "exit", strlen("exit") + 1, 0);

            break;

        }

        send(socketfd, filename, strlen(filename) + 1, 0);

        recv(socketfd, buffer, sizeof(buffer), 0);

        if (strncmp(buffer, "404", strlen("404")) == 0)

        {

            fprintf(stdout, "File not found. Try again.\n");

            continue;

        }

        else if (strncmp(buffer, "FOUND", strlen("FOUND")) == 0)

        {

            fprintf(stdout, "File found.\n");

        }

        else

        {

            fprintf(stdout, "Unexpected Error.\n");

            break;

        }

        recv(socketfd, buffer, sizeof(buffer), 0);

        int fd = open(filename, O\_CREAT | O\_WRONLY);

        write(fd, buffer, strlen(buffer) + 1);

        close(fd);

        fprintf(stdout, "FILE RECEIVED AND STORED LOCALLY.\n");

        printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

        printf("\n%s\n", filename);

        printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

        printf("\n%s\n", buffer);

    }

    close(socketfd);

    return;

}

void loading()

{

    for (int k = 0; k < 10000000; k++)

    {

        int j = k;

    }

}

int main()

{

    int socketfd;

    struct sockaddr\_in client\_addr;

    char buffer[1024], username[1024], password[1024];

    socketfd = getConnectedSocket(SERVERPORT);

    if (socketfd < 0)

    {

        return -1;

    }

REQ:

    buffer[0] = '\0';

    fprintf(stdout, "\n----------------------------------------\n");

    fprintf(stdout, "\n [-] Enter username : ");

    fscanf(stdin, "%s", username);

    fprintf(stdout, "\n [-] Enter password : ");

    fscanf(stdin, "%s", password);

    strcat(buffer, "auth$");

    strcat(buffer, username);

    strcat(buffer, "$");

    strcat(buffer, password);

    strcat(buffer, "$");

    printf("\nLOADING");

    for (int k = 0; k < 8; k++)

    {

        loading();

        printf(".");

    }

    printf("\n");

    send(socketfd, buffer, strlen(buffer) + 1, 0);

    recv(socketfd, buffer, sizeof(buffer), 0);

    if (strncmp(buffer, "INVALID\_CRED", strlen("INVALID\_CRED")) == 0)

    {

        fprintf(stdout, "INVALID..TRY AGAIN\n");

        goto REQ;

    }

    if (strncmp(buffer, "ERROR", strlen("ERROR")) == 0)

    {

        fprintf(stdout, "Couldn't authorize credentials now. Try Again.\n");

        return -1;

    }

    int file\_port = atoi(buffer);

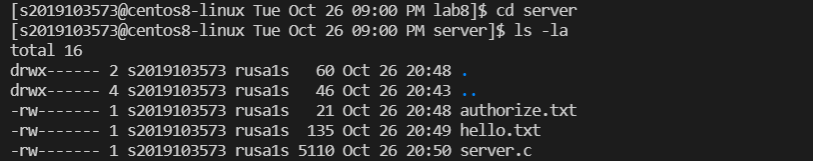
    file\_transfer(file\_port);

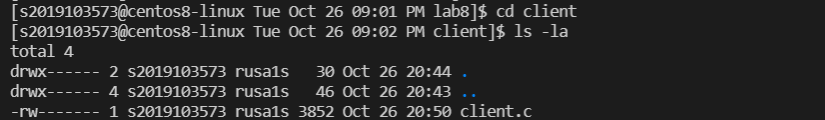
    close(socketfd);

    return 0;

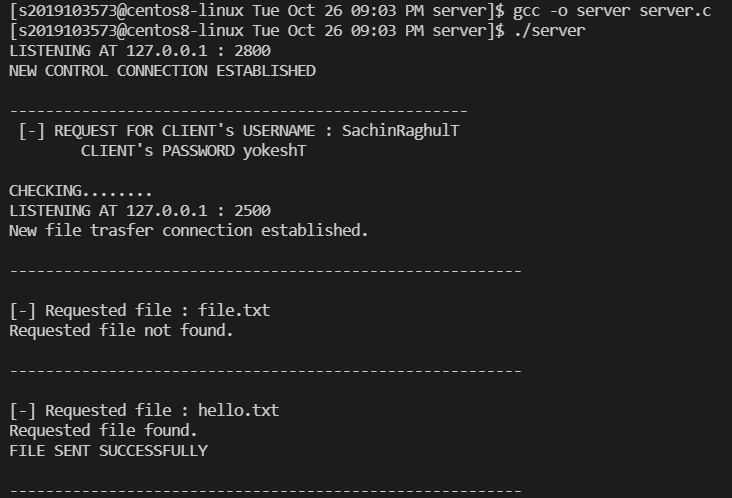
}

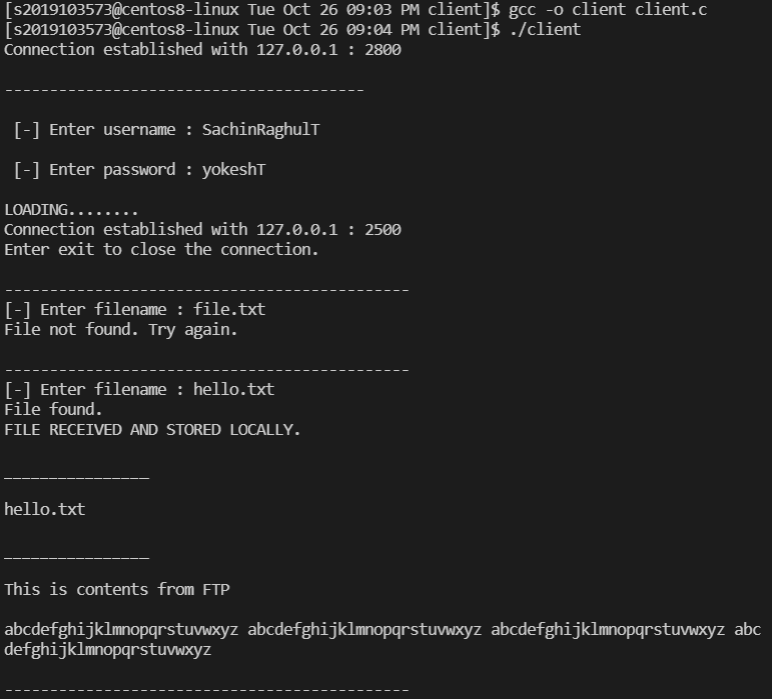
**BEFORE EXECUTION :-**

**SERVER DIRECTORY**

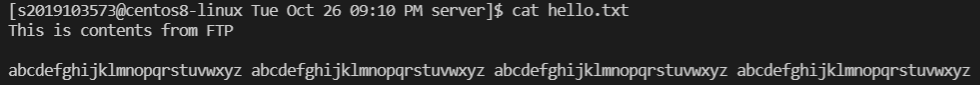
**CLIENT DIRECTORY**

**OUPUT :-**

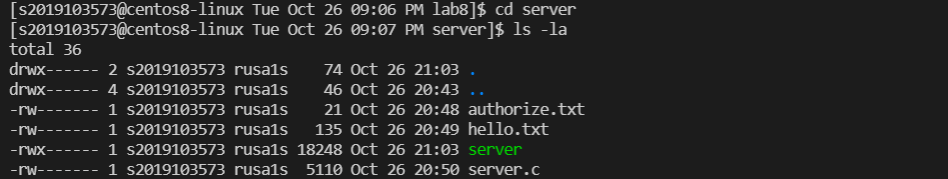
**SERVER OUPTUT**

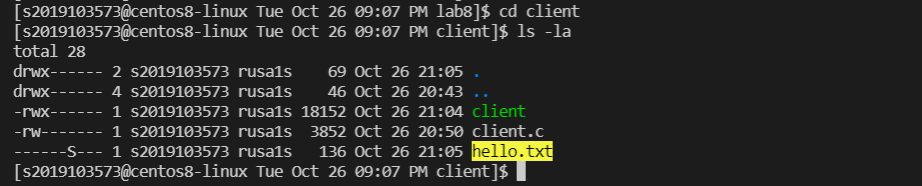
**CLIENT OUTPUT**

**FTP FILE :-**

**hello.txt**

**AFTER EXECUTION :-**

**SER** **VER DIRECTORY**

**CLIENT DIRECTORY**