SSN College of Engineering, Kalavakkam
Department of Computer Science and Engineering
V Semester - CSE 'B'
UCS1511 NETWORKS LAB

Date :13/08/2020 **Name :** Rahul Ram M
Exercise : 04 **Reg No :** 185001121

FILE TRANSFER USING TCP

Learning Objective:

To transfer a file from server to client using TCP socket programming.

Algorithm for Server:

1. Creating a socket using the function socket(domain, type, protocol) which the returns an integer as the status of the socket creation. Here the domain is AF_INET(iPv4 protocol), type is SOCK_STEAM and protocol as 0.
2. Using bzero(&server_addr, sizeof(server_addr)) function setting values of all the socket structures to null.
3. Using bind() to binf the socket to the address and port number specified in addr(custom data structure). Here, we bind the server to the localhost, hence we use INADDR_ANY to specify the IP address.
4. listen() function is used to set the server socket in the passive mode, where it waits for the client to approach the server to make a connection, with maximum number of connection in this case is 2.
5. accept() creates a new connected socket and returns a new file descriptor reffering to the socket. After this the connection between server and client is established.
6. read(new_socket, filename, sizeof(filename)) - reads the filename sent by the client in the variable filename specified in the parameter along with its size preceded by the new socket descriptor.
7. Printing the filename received from the client using printf().
8. Open the file with Read mode using fopen(filename, “r”) and assign the file descriptor to fp.
9. Get the number of bytes of the file using fseek(fp, 0L, SEEK_END) and ftell(fp) and assgin the value in numbytes.
10. Now reset the file position indicator to the beginning of the file using fseek(fp, 0L, SEEK_SET).
11. Copy all the text from the file into the buffer using fread(buffer, sizeof(char), numbytes, fp).
12. Close the file using fclose().
13. write(new_socket, buffer, sizeof(buffer)) - is used to write the contents of the file stored in buffer to be read by the client.
14. close() function shuts down the socket associated with socket descriptor’s, and frees resources allocated to the socket.

Algorithm for Client:

1. Creating a socket using the function socket(domain, type, protocol) which the returns an integer as the status of the socket creation. Here the domain is AF_INET(iPv4 protocol), type is SOCK_STEAM and protocol as 0.
2. Using bzero(&server_addr, sizeof(server_addr)) function setting values of all the socket structures to null.
3. The above two steps are same as the server.
4. The connect() system call connects the socket referred to by the file descriptor socket_fd to the address specified by server_addr. Server's address and port is specified in server_addr.
5. Read the filename from the user using scanf().
6. write(socket_fd, filename, sizeof(filename)) – sends the filename given by the cient to the server.
7. read(new_socket, buffer, sizeof(buffer)) - reads the contents of the file sent by the server in the buffer specified in the parameter along with its size preceded by the socket_fd.
8. Read the path of the file from user where the new file to be stored.
9. Open the file using fopen() with write persmission.
10. Using fprintf(fp, "%s", buffer) to write the contents of the buffer to the file specified by the filename along with the path given by the user.
11. close() function shuts down the socket associated with socket descriptor's, and frees resources allocated to the socket.

Program for Server:

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

#define PORT 8080

int main()
{
	int  server_fd, new_socket, value;
	struct sockaddr_in server_addr, client_addr;
	char buffer[1024], filename[100];
	socklen_t len;
	long numbytes;
	FILE *fp;

	if((server_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
	{
		perror("Socker error");
	}

	bzero(&server_addr,sizeof(server_addr));

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

```
	if(bind(server_fd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0)
	{
		perror("Bind error: ");
	}

	if(listen(server_fd,2) < 0)
	{
		perror("Listen error");
	}

	len = sizeof(client_addr);

	printf("Waiting for client...\n");

	if((new_socket = accept(server_fd, (struct sockaddr*)&client_addr, &len)) < 0)
	{
		perror("Accept error");
	}

	value = read(new_socket, filename, sizeof(filename));
	printf("File to be transferred is %s\n", filename);

	fp = fopen(filename, "r");
	fseek(fp, 0L, SEEK_END);
	numbytes = ftell(fp);
	fseek(fp, 0L, SEEK_SET);
	fread(buffer, sizeof(char), numbytes, fp);
	fclose(fp);

	value = write(new_socket, buffer, sizeof(buffer));
	printf("File Transferred!\n");

	close(server_fd);
	close(new_socket);
	return 0;
}
```

Program for Client:

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

#define PORT 8080

int main()
{
	int socket_fd, value;
	struct sockaddr_in server_addr;
	FILE *fp;
	char buffer[1024], filename[100];
```

```
	if((socket_fd=socket(AF_INET, SOCK_STREAM, 0)) < 0)
	{
		perror("Socket error");
	}

	bzero(&server_addr,sizeof(server_addr));

	server_addr.sin_family = AF_INET;
	server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
	server_addr.sin_port = htons(PORT);

	if(connect(socket_fd,(struct sockaddr*)&server_addr, sizeof(server_addr)) < 0)
	{
		perror("Connect error");
	}

	printf("Enter the path of file : ");
	scanf("%s", filename);
	value = write(socket_fd, filename, sizeof(filename));

	value = read(socket_fd, buffer, sizeof(buffer));
	printf("FIle Transferred!\n");

	printf("Save the file in path : ");
	scanf("%s", filename);

	fp = fopen(filename, "w");
	fprintf(fp, "%s", buffer);
	fclose(fp);

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

Server Screenshot:

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

#define PORT 8080

int main()
{
        int  server_fd, new_socket, value;
        struct sockaddr_in server_addr, client_addr;
        char buffer[1024], filename[100];
        socklen_t len;
        long numbytes;
        FILE *fp;

        if((server_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
        {
                perror("Socker error");
        }

        bzero(&server_addr,sizeof(server_addr));

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

        if(bind(server_fd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0)
        {
                perror("Bind error: ");
        }

        if(listen(server_fd,2) < 0)
        {
                perror("Listen error");
        }

        len = sizeof(client_addr);

        printf("Waiting for client...\n");

        if((new_socket = accept(server_fd, (struct sockaddr*)&client_addr, &len)) < 0)
        {
                perror("Accept error");
        }

        value = read(new_socket, filename, sizeof(filename));
        printf("File to be transferred is %s\n", filename);

        fp = fopen(filename, "r");
```

```c

        fp = fopen(filename, "r");
        fseek(fp, 0L, SEEK_END);
        numbytes = ftell(fp);
        fseek(fp, 0L, SEEK_SET);
        fread(buffer, sizeof(char), numbytes, fp);
        fclose(fp);

        value = write(new_socket, buffer, sizeof(buffer));
        printf("File Transferred!\n");

        close(server_fd);
        close(new_socket);
        return 0;
}
```

Client Screenshot:

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

#define PORT 8080

int main()
{
        int socket_fd, value;
        struct sockaddr_in server_addr;
        FILE *fp;
        char buffer[1024], filename[100];

        if((socket_fd=socket(AF_INET, SOCK_STREAM, 0)) < 0)
        {
                perror("Socket error");
        }

        bzero(&server_addr,sizeof(server_addr));

        server_addr.sin_family = AF_INET;
        server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
        server_addr.sin_port = htons(PORT);

        if(connect(socket_fd,(struct sockaddr*)&server_addr, sizeof(server_addr)) < 0)
        {
                perror("Connect error");
        }

        printf("Enter the path of file : ");
        scanf("%s", filename);
        value = write(socket_fd, filename, sizeof(filename));

        value = read(socket_fd, buffer, sizeof(buffer));
        printf("FIle Transferred!\n");

        printf("Save the file in path : ");
        scanf("%s", filename);

        fp = fopen(filename, "w");
        fprintf(fp, "%s", buffer);
        fclose(fp);

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

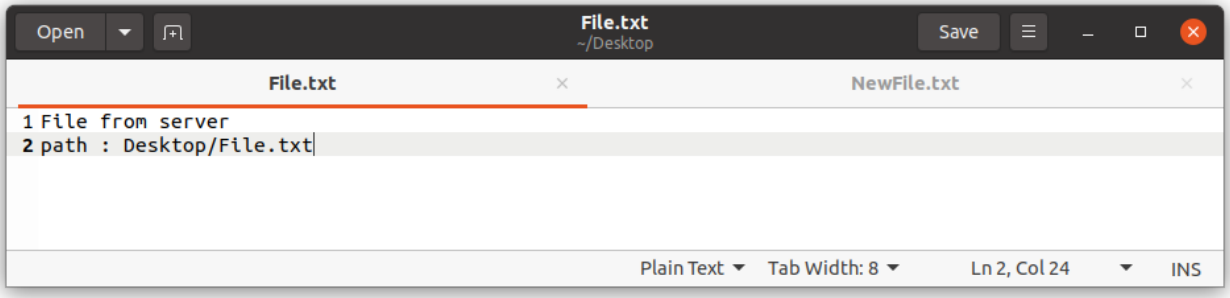
Server Output:

```
rahul@rahul-Ubuntu:~/Sem_05/NWLAB/Ex_04$ ./s
Waiting for client...
File to be transferred is /home/rahul/Desktop/File.txt
File Transferred!
rahul@rahul-Ubuntu:~/Sem_05/NWLAB/Ex_04$
```

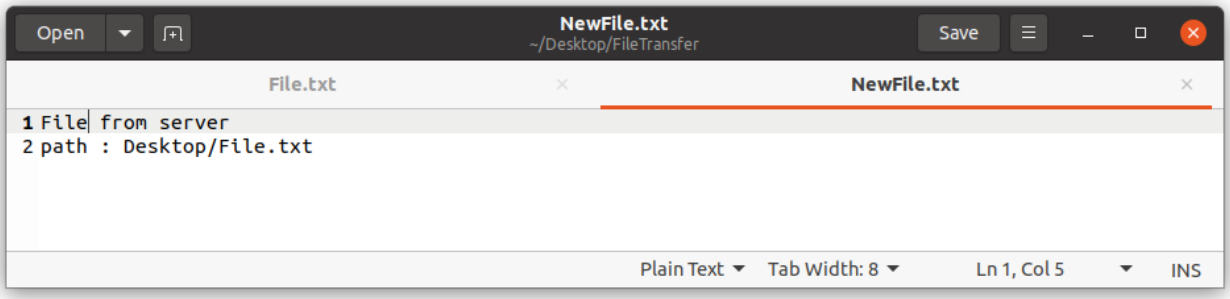
Client Output:

```
rahul@rahul-Ubuntu:~/Sem_05/NWLAB/Ex_04$ ./c
Enter the path of file : /home/rahul/Desktop/File.txt
FIle Transferred!
Save the file in path : /home/rahul/Desktop/FileTransfer/NewFile.txt
rahul@rahul-Ubuntu:~/Sem_05/NWLAB/Ex_04$
```

File at Server:
/home/rahul/Desktop/File.txt:



File at Client:
/home/rahul/Desktop/FileTransfer/NewFile.txt:



Learning Outcomes:

This assignment helped me to

1. Write program for server and client with socket programming.
2. Understand various functions invloved in creating, estabilishing, maintaining, Sending, recieving and termininating the connection between the server and client.
3. Write code to make server and client communicate with each other using read() and write() functions.
4. Modify the code to transfer a file between client and server.