SSN College of Engineering
UCS1511 - Networks Lab
Exercise 4 – File transfer using TCP

Prasanna Kumaran D
185001110
Semester V
Batch 2018 - 2022

October 3, 2020

---

1 File Transfer using TCP

1.1 Aim

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

1.2 Algorithm

1.2.1 Server

Server opens the file and transfers it through the socket.

1. Create a socket descriptor with **socket()** system call and use AF_INET as domain and SOCK_STREAM for domain and communication type, store the socket descriptor in sockfd. Initialize client file descriptors to 0
2. If sockfd is a negative number,
   (a) print socket creation failed, terminate program.
3. Assign family, address and port to the server socketadd_in object. Set the family as AF_INET to access IPv4 protocols, and INADDR_ANY for address to accept connections from any client.
4. bind the socket to the server sockadd_in object
5. If bind is non zero,

   (a) Print bind creation failed and terminate.

6. Read the path of the file to be transferred from the buffer Open the file in read mode (**"r"**) and save it in a **FILE** pointer

7. Open the destination file in write mode (**"w"**)

8. Looping while there are contents to be read from the source file

   (a) Read a character from the source file

   (b) Write the character into the destination file

9. Write the destination file location into the buffer

10. Close socket connection using **close()** and terminate program

1.2.2 Client

Client requests for a File from the server. Client receives the file through the TCP socket and save it in a new location.

1. Create a socket descriptor using **socket()**
   with AF_INET(IPv4 domain), SOCK_STREAM(connection type)

2. if socket < 0

   (a) Print socket creation failed and terminate program

3. Create a sockaddr_in object for the client and set up family, address and port number.

4. Call connect() to establish connection between client and server

5. Request for a file from the server. Write the path of the file into the buffer

6. Read the message sent from the server

7. If message not NULL

   - Display destination file location

8. Close socket connection using **close()** and terminate program

1.3 Program

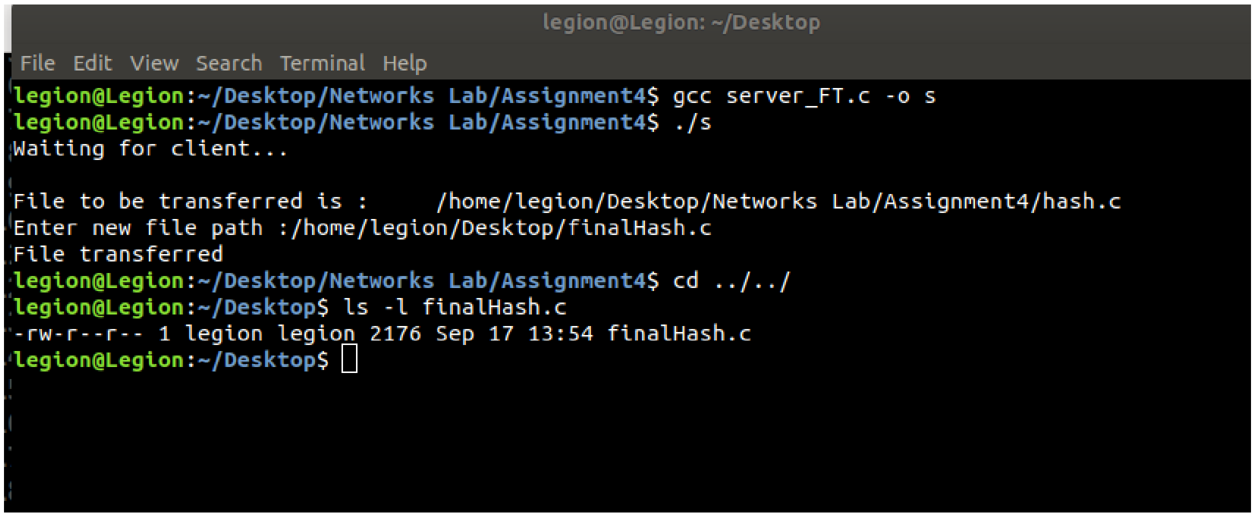
1.3.1 Server

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<unistd.h>
int main(int argc,char **argv)
{

        int len;
        int sockfd, newfd,n;
        struct sockaddr_in servaddr,cliaddr;

        char buff[1024];
        char destination[200];
        char ch;
        sockfd=socket(AF_INET,SOCK_STREAM,0);
        if(sockfd < 0)
                perror("cannot create socket");

        bzero(&servaddr,sizeof(servaddr));

        servaddr.sin_family=AF_INET;
        servaddr.sin_addr.s_addr=INADDR_ANY;
        servaddr.sin_port=htons(8083);

        if(bind(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr))<0)
                perror("Bind error");
        printf("Waiting for client...\n");
        listen(sockfd,2);
        len = sizeof(cliaddr);
        newfd = accept(sockfd,(struct sockaddr*)&cliaddr,&len);
        //Receiving the message
        FILE *fdest, *fsource;
        n = read(newfd,buff,sizeof(buff));

        printf("\nFile to be transferred is :\t%s",buff);
        fsource = fopen(buff,"r");
        printf("\nEnter new file path :");
        scanf("%s",destination);
        fdest = fopen(destination, "w");
        while(( ch = fgetc(fsource) )!=EOF )
            fputc(ch,fdest);
        printf("File transferred\n");
        strcpy(buff,destination);
        n = write(newfd, buff, sizeof(buff));
        close(sockfd);
        close(newfd);
        return 0;
}
```

1.3.2 Client

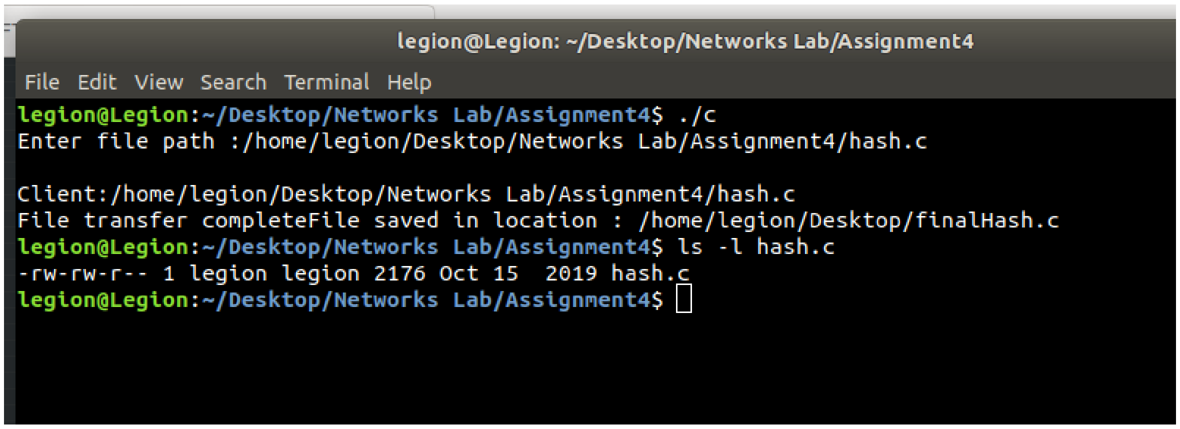
```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
int main(int argc,char **argv)
{

        int len;
        int sockfd,n;
        struct sockaddr_in servaddr,cliaddr;

        char buff[1024];
        char response[200];

        sockfd=socket(AF_INET,SOCK_STREAM,0);
        if(sockfd<0)
                perror("cannot create socket");

        bzero(&servaddr,sizeof(servaddr));

        cliaddr.sin_family = AF_INET;
        cliaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
        cliaddr.sin_port = htons(8083);

        connect(sockfd,(struct sockaddr*)&cliaddr,sizeof(cliaddr));

        //Sending file information
        printf("Enter file path :");
        scanf("%[^\n]s",buff);
        printf("\nClient:%s",buff);
        n = write(sockfd,buff,sizeof(buff));
        // Receiving file from the server
        n = read(sockfd, response, sizeof(response));
        if (response[0] != '\0'){
                printf("\nFile transfer complete");
                printf("File saved in location : %s\n", response);
                }
        else
                printf("No files transferred! Exitting...");
        close(sockfd);
        return 0;
}
```

1.4 Output



```
legion@Legion:~/Desktop/Networks Lab/Assignment4$ gcc server_FT.c -o s
legion@Legion:~/Desktop/Networks Lab/Assignment4$ ./s
Waiting for client...

File to be transferred is :     /home/legion/Desktop/Networks Lab/Assignment4/hash.c
Enter new file path :/home/legion/Desktop/finalHash.c
File transferred
legion@Legion:~/Desktop/Networks Lab/Assignment4$ cd ../../
legion@Legion:~/Desktop$ ls -l finalHash.c
-rw-r--r-- 1 legion legion 2176 Sep 17 13:54 finalHash.c
legion@Legion:~/Desktop$ 
```

Figure 1: Server side



```
legion@Legion:~/Desktop/Networks Lab/Assignment4$ ./c
Enter file path :/home/legion/Desktop/Networks Lab/Assignment4/hash.c

Client:/home/legion/Desktop/Networks Lab/Assignment4/hash.c
File transfer completeFile saved in location : /home/legion/Desktop/finalHash.c
legion@Legion:~/Desktop/Networks Lab/Assignment4$ ls -l hash.c
-rw-rw-r-- 1 legion legion 2176 Oct 15  2019 hash.c
legion@Legion:~/Desktop/Networks Lab/Assignment4$ 
```

Figure 2: Client side

2 Learning Outcomes

- Learnt how to establish a simple client server connection using TCP
- Learnt about the basic syntax and system calls used in socket programming
- Learnt how to handle errors in socket programming
- Learnt how to handle files in C programming and how to transfer the file contents from the server to the requesting client