UCS 1511 - Network Lab
Exercise 4 - File Transfer using TCP

Name:	Mahesh Bharadwaj K
Reg No:	185001089
Semester:	V
Date:	September 15, 2020

---

1 File Transfer using TCP

Aim:

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

Algorithm

1. Server

1. Create a socket descriptor with **socket()** system call with AF_INET (IPV4 domain), SOCK_STREAM, default protocol and store as sockfd.
2. If sockfd is a negative number, socket creation failed, end program.
3. Create sockaddr_in object to assign IP address and Port number for socket. Set family to AF_INET, IP address to INADDR_ANY to accept connections from any client and port number required.
4. Bind newly created socket to addresss given in sockaddr_in.
5. If bind is non zero, bind failed, print error message and terminate.
6. Listen on the socked defined for as many clients as required. If **listen()** returns non zero value, print error message and terminate.
7. Accept connections from socket using **accept()** system call and store client socket descriptor in a separate variable.
8. Read file requested into buffer using **read()** system call.
9. Create file descriptor 'fd' using **open()** in read mode (O_RDONLY).
10. IF fd < 0
    - Write "File Not found" to buffer using **write()**.
    - Display Error Message and terminate program.
11. ELSE
    - Read the contents of file into buffer using **read()**
    - Write the contents into client socket descriptor using **write()**
12. Close connections on socket using **close()** and terminate program.

2. Client

1. Create a socket descriptor with **socket()** system call with AF_INET (IPV4 domain), SOCK_STREAM, default protocol and store as sockfd.
2. If sockfd is a negative number, socket creation failed, end program.
3. Create sockaddr_in object to assign IP address and Port number for socket. Set family to AF_INET, IP address to localhost(127.0.0.1) to connect to server and port number required.
4. Connect the client to server at address given in socket descriptor using **connect()** system call.
5. If connect() returns -1, connection failed; Print error message and terminate the program.
6. Read file path from user into buffer variable and write into server socket using **write()** system call.
7. Read the response from server into buffer variable using **read()** system call.
8. IF response is "File Not found"
    - Print Error Message that the file requested doesn't exit.
    - Terminate program
9. ELSE
    - Accept the path to save file from user.
    - Create new file using **creat()** with Read, Write permissions.
    - Write data into file descriptor using **write()** system call and close file descriptor
10. Close the connections on socket using **close()** and terminate program.

Program

1. Server Side

```
#include <stdio.h>
#include <netdb.h>
#include<fcntl.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#define MAX 1024

#define SA struct sockaddr
int main(int argc, char ** argv)
{
         if(argc < 2){
        fprintf(stderr, "Please pass port number for server as second argument!\n");
        exit(EXIT_FAILURE);
    }

        int PORT = atoi(argv[1]);
        int sockfd, new_fd, len;
        struct sockaddr_in servaddr, cli;
        char buff[MAX];
        int n;
        // socket create and verification
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd == -1)
        {
                fprintf(stderr, "Socket creation failed!\n");
                exit(EXIT_FAILURE);
        }
        else
                printf("Socket creation successfull!\n");

        bzero(&servaddr, sizeof(servaddr));
        // assign IP, PORT
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
        servaddr.sin_port = htons(PORT);

        // Binding newly created socket to given IP and verification
        if ((bind(sockfd, (SA *)&servaddr, sizeof(servaddr))) != 0)
        {
                fprintf(stderr,"Socket bind failed!\n");
                exit(EXIT_FAILURE);
        }
        else
                printf("Socket successfully binded..\n");

        // Now server is ready to listen and verification
        if ((listen(sockfd, 5)) != 0)
        {
                fprintf(stderr,"Listen failed!\n");
                exit(EXIT_FAILURE);
        }
        else
                printf("Server listening..\n");

        len = sizeof(cli);
        // Accept the data packet from client and verification
        new_fd = accept(sockfd, (SA *)&cli, &len);
        if (new_fd < 0)
        {
                fprintf(stderr,"Server acccept failed!\n");
                exit(EXIT_FAILURE);
        }
        else
                printf("Accept Successfull!\n");

        bzero(buff, MAX);
        // read the message from client and copy it in buffer
        read(new_fd, buff, MAX);

        printf("File to be transferred: %s\n", buff);

        int fd = open(buff, O_RDONLY);

        bzero(buff, MAX);

        if (fd < 0){
                strcpy(buff, "File Not Found!");
                fprintf(stderr, "%s\n", buff);
                write(new_fd, buff, MAX);
        }
        else{
                read(fd, buff, MAX);
                printf("File Transferred\n");
                write(new_fd, buff, MAX);
        }
        close(new_fd);
```

```
        close(sockfd);
}
```

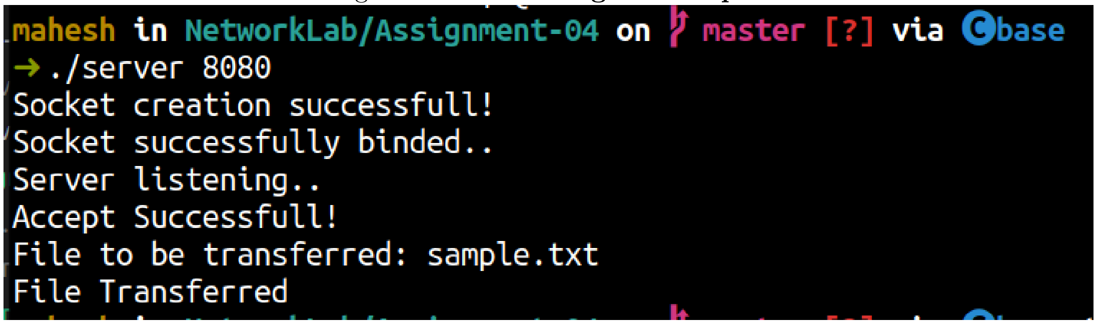
---

2. Client Side

```
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>
#include <sys/socket.h>
#define MAX 1024
#define SA struct sockaddr
int main(int argc, char ** argv)
{


         if(argc < 2){
        fprintf(stderr, "Please pass port number of server as second argument!\n");
        exit(EXIT_FAILURE);
    }
        int PORT = atoi(argv[1]);

        int sockfd, connfd;
        struct sockaddr_in servaddr, cli;
        char buff[MAX] = {0},
             filename[30] = {0};

        int n; // socket create and varification
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd == -1)
        {
                fprintf(stderr,"Socket creation failed!\n");
                exit(0);
        }
        else
                printf("Socket creation successfull!\n");

        bzero(&servaddr, sizeof(servaddr));
        // assign IP, PORT
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
        servaddr.sin_port = htons(PORT);

        // connect the client socket to server socket
        if (connect(sockfd, (SA *)&servaddr, sizeof(servaddr)) != 0)
        {
                fprintf(stderr,"Connection failed!\n");
                exit(0);
        }
        else
                printf("Connection to server successfull!\n");

        printf("Enter the path of the file: ");
        scanf("%[^\n]", buff);
        getchar();
        write(sockfd, buff, MAX);
        read(sockfd, buff, MAX);
        if(strcmp(buff, "File Not Found!")==0){
                fprintf(stderr, "%s\n", buff);
                exit(EXIT_FAILURE);
        }

        else{
                printf("File Transferred\nEnter the path to save: ");
                scanf("%[^\n]", filename);
                int fd = creat(filename,S_IRWXU);

                if (fd < 0){
                        fprintf(stderr, "Unable to create file!\n");
                        exit(EXIT_FAILURE);
                }

                write(fd, buff, strlen(buff));
                close(fd);
        }

        close(sockfd);
}
```

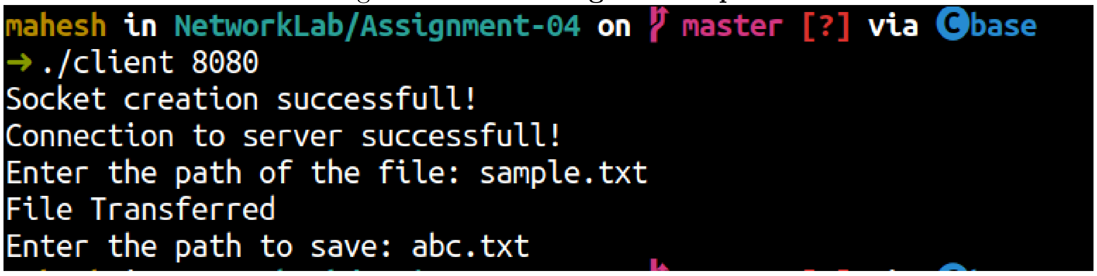
Output

Figure 1: **Server Program Output**



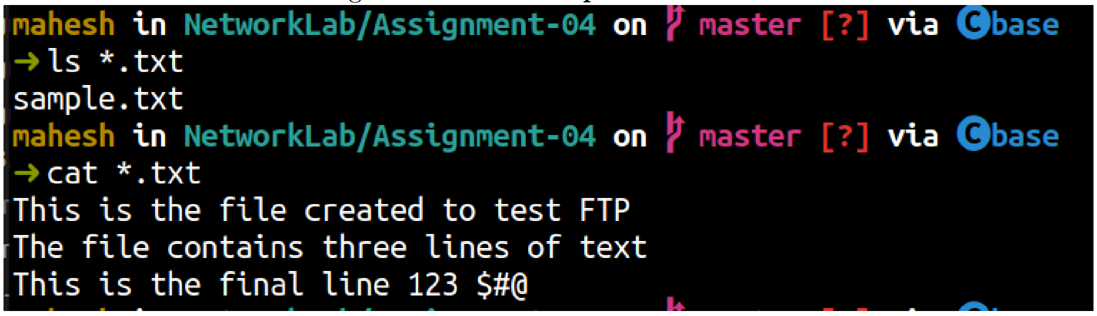
```
mahesh in NetworkLab/Assignment-04 on master [?] via base
→ ./server 8080
Socket creation successfull!
Socket successfully binded..
Server listening..
Accept Successfull!
File to be transferred: sample.txt
File Transferred
```

Figure 2: **Client Program Output**



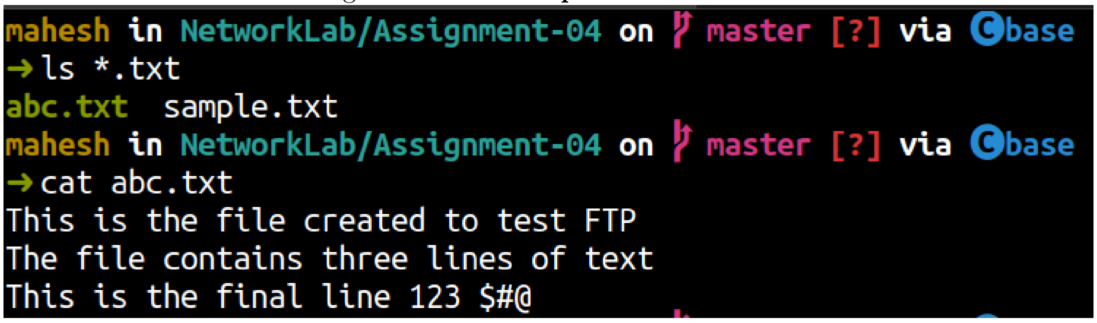
```
mahesh in NetworkLab/Assignment-04 on master [?] via base
→ ./client 8080
Socket creation successfull!
Connection to server successfull!
Enter the path of the file: sample.txt
File Transferred
Enter the path to save: abc.txt
```

Figure 3: **Before Request at server**



```
mahesh in NetworkLab/Assignment-04 on  master [?] via base
→ls *.txt
sample.txt
mahesh in NetworkLab/Assignment-04 on  master [?] via base
→cat *.txt
This is the file created to test FTP
The file contains three lines of text
This is the final line 123 $#@
```

Figure 4: **After Request from client**



```
mahesh in NetworkLab/Assignment-04 on  master [?] via base
→ls *.txt
abc.txt  sample.txt
mahesh in NetworkLab/Assignment-04 on  master [?] via base
→cat abc.txt
This is the file created to test FTP
The file contains three lines of text
This is the final line 123 $#@
```

Learning Outcomes:

- We learn how to create a simple TCP client server connection.
- We learn how to appropriate system calls to set up Server and Client Programs.
- We learn to open file in server, send through socket and save received file at the client.