

Swetha Saseendran

CSE-C

185001183

Exp 3: FILE TRANSFER USING TCP

FILE.TXT CONTENTS:

This is a sample file.

To be transferred.

Thankyou!

Bye! :)

SERVER.C

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <fcntl.h>
#define SA struct sockaddr
#define SERVER_PORT 4000
#define MAX 256

int error(char *msg){
    perror(msg);
    exit(1);
}

int main(int argc, char const *argv[]) {
    int bc,flag;
    char fileBuf[MAX],buf[MAX];
    int server_fd, client_fd, fd;
    struct sockaddr_in servaddr, clientaddr;
```

```

//CREATE SOCKET
if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) <= 0)
    error("SOCKET FAILED");

//RESET servaddr / ASSIGN IP, PORT, FAMILY
bzero(&servaddr, sizeof(servaddr));
if ((inet_aton("127.0.0.1", &servaddr.sin_addr)) == 0)
    error("IP ERROR");
servaddr.sin_port = htons(SERVER_PORT);
servaddr.sin_family = AF_INET;

//BIND TO PORT
if ((bind(server_fd, (SA*)&servaddr, sizeof(servaddr))) != 0)
    error("BIND ERROR");

//LISTEN ON PORT
if ((listen(server_fd, 5)) != 0)
    error("LISTEN ERROR");

//ACCEPT CONNECTION
int len = sizeof(clientaddr);
if ((client_fd = accept(server_fd, (SA*)&clientaddr, &len)) <= 0)
    perror("ACCEPT ERROR");

//GET FILE NAME
flag = read(client_fd, buf, MAX);
printf("FILE REQUESTED TO BE READ: %s\n", buf);

//OPEN FILE
if ((fd = open(buf, O_RDONLY)) <= 0)
    error("FILE OPEN ERROR");

//READ FILE
if (fd < 0)
    error("FILE NOT FOUND");

len = read(fd, fileBuf, MAX);
close(fd);
flag = write(client_fd, fileBuf, MAX);
printf("\nFILE CONTENTS TRANSFERRED\n");

//CLOSE
bzero(buf, MAX);
bzero(fileBuf, MAX);
close(server_fd);
close(client_fd);
return 0;

```

```
}
```

CLIENT.C

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <fcntl.h>
#define SA struct sockaddr
#define SERVER_PORT 4000
#define MAX 256

int error(char *msg){
    perror(msg);
    exit(1);
}

int main(int argc, char const *argv[]) {
    char buf[MAX],file_name[20];
    int flag;
    int sock_fd = 0, server_fd;
    struct sockaddr_in servaddr;

    //CREATE SOCKET
    if ((sock_fd = socket(PF_INET,SOCK_STREAM,0)) <= 0)
        error("SOCKET FAILED");

    //RESET servaddr / ASSIGN IP, PORT, FAMILY OF **SERVER**
    bzero(&servaddr, sizeof(servaddr));
    if ((inet_aton("127.0.0.1", &servaddr.sin_addr)) == 0)
        error("IP ERROR");
    servaddr.sin_port = htons(SERVER_PORT);
    servaddr.sin_family = AF_INET;

    //CONNECT
    if ((connect(sock_fd, (SA*)&servaddr, sizeof(servaddr))) != 0)
        error("CONNECT ERROR");

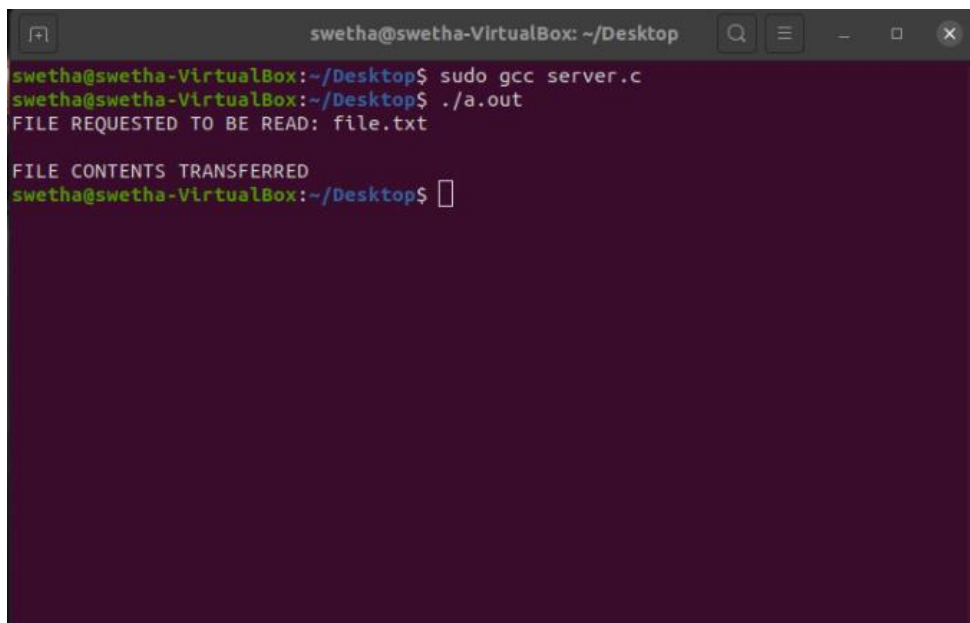
    //SEND FILE NAME
    printf("\nEnter the File Location: ");
    scanf(" %[^\\n]", file_name);
    flag = write(sock_fd, file_name, MAX);

    //GET CONTENT BACK
```

```
flag = read(sock_fd, buf, MAX);  
printf("FILE:\n%s\n", buf);  
int newfd = open("newfile.txt", O_WRONLY | O_CREAT);  
printf("FILE SAVED AS: newfile.txt\n");  
write(newfd, buf, MAX);  
  
//CLOSE  
bzero(buf, MAX);  
close(sock_fd);  
return 0;  
}
```

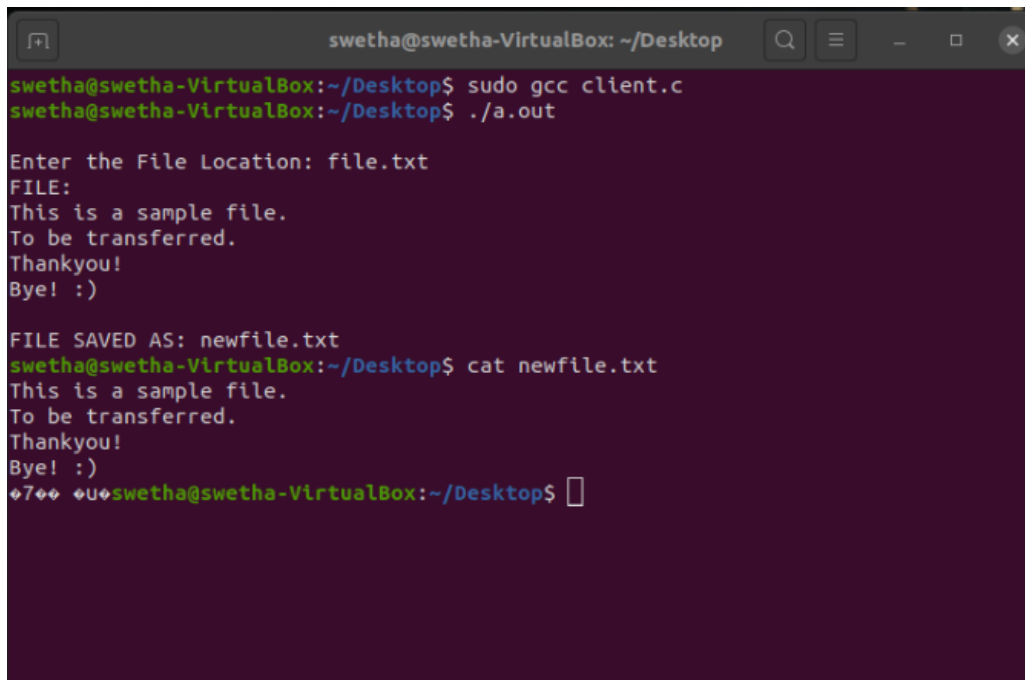
Snapshot of output:

Server Side:

A terminal window titled 'swetha@swetha-VirtualBox: ~/Desktop' with standard window controls. The terminal shows the execution of a server program. The user runs 'sudo gcc server.c' and then './a.out'. The program outputs 'FILE REQUESTED TO BE READ: file.txt' and 'FILE CONTENTS TRANSFERRED' before returning to the prompt.

```
swetha@swetha-VirtualBox: ~/Desktop  
swetha@swetha-VirtualBox:~/Desktop$ sudo gcc server.c  
swetha@swetha-VirtualBox:~/Desktop$ ./a.out  
FILE REQUESTED TO BE READ: file.txt  
  
FILE CONTENTS TRANSFERRED  
swetha@swetha-VirtualBox:~/Desktop$
```

Client Side:



```
swetha@swetha-VirtualBox: ~/Desktop
swetha@swetha-VirtualBox:~/Desktop$ sudo gcc client.c
swetha@swetha-VirtualBox:~/Desktop$ ./a.out

Enter the File Location: file.txt
FILE:
This is a sample file.
To be transferred.
Thankyou!
Bye! :)

FILE SAVED AS: newfile.txt
swetha@swetha-VirtualBox:~/Desktop$ cat newfile.txt
This is a sample file.
To be transferred.
Thankyou!
Bye! :)
♦7♦♦ ♦u♦swetha@swetha-VirtualBox:~/Desktop$
```

The image shows a terminal window titled "swetha@swetha-VirtualBox: ~/Desktop". The user runs the command `sudo gcc client.c` to compile a C program, followed by `./a.out` to execute it. The program prompts for a file location, reads "file.txt", and displays its contents: "This is a sample file. To be transferred. Thankyou! Bye! :)". It then reports "FILE SAVED AS: newfile.txt". The user runs `cat newfile.txt` to verify the saved content, which matches the original file's content. The terminal ends with a prompt and some decorative symbols.