SSN College of Engineering, Kalavakkam

Department of Computer Science and Engineering

V Semester - CSE 'B'

UCS1511 NETWORKS LAB

Name: Likhitha Verma A
REG No: 185001084
Date: 25/08/2020

Exercise 4 : FILE TRANSFER USING TCP

Aim :

To develop a socket program to establish a client server communication. The client requests for a file from the server and saves it in a new location. The server opens the file and transfers it through the socket.

Learning objectives :

- To learn C socket programming.
- To learn how to create a TCP client server connection for file transferring.
- To learn how application programs use protocol software to communicate across networks and internets.
- To learn about various system calls used in socket programming.

Algorithm :

SERVER SIDE:

1. Create a socket descriptor with **socket()** system call with AF_INET, SOCK STREAM, default protocol and store as sockfd.
2. If sockfd returns a negative value, print error message.
3. Create sockaddr_in object and initialize with values.

4. Bind newly created socket to address given in sockaddr _in using **bind()** system call.
5. If **bind()** returns negative value, bind failed, print error message.
6. Listen for connections using **listen()** system call.
7. Accept connections from socket using **accept()** system call and store client socket descriptor in a separate variable.
8. Read filename into fname variable using **read()** system call open the same using **open()** system call in read-only mode.
9. If the **open()** system call returns a negative value, print an error message and goto step 11.
10. Read the contents of the file using **read()** into the buffer.
11. Write the buffer onto the client using **write()** system call.
12. Close connections on socket using **close()** and terminate program.

CLIENT SIDE:

1. Create a socket descriptor with **socket()** system call with AF_INET, SOCK STREAM,
   default protocol and store as sockfd.
2. If sockfd returns a negative value, print error message.
3. Create sockaddr_in object and initialize with values.
4. Connect the client to the server at the address given in the socket descriptor using **connect()** system call.
5. If **connect()** returns negative value, connection failed, print error message.
6. Read a filename from the user and write it onto the server socket using **write()** system call.

7. Read the message from the server into a buffer variable using **read()** system call and write it into the destination file using **open()** and **write()** system calls.
8. Close connections on socket using **close()** and terminate program.

Program:

SERVER SIDE:

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<fcntl.h>
#include <unistd.h> //for read, write and close

int main(){

    int sockfd,newfd,n,fd;
    struct sockaddr_in seraddr, client1;
    char buff[1024],fname[100];

    //Socket Creation
    sockfd=socket(AF_INET,SOCK_STREAM,0);
    if(sockfd<0)
        perror("Socket Creation failed...!");
    else
        printf("Socket Created Successfully...!\n");

    bzero(&seraddr,sizeof(seraddr));

    seraddr.sin_family=AF_INET;
    seraddr.sin_port=htons(3335);
    seraddr.sin_addr.s_addr=htonl(INADDR_ANY);

    //binding to the socket
    if(bind(sockfd,(struct sockaddr *) &seraddr, sizeof(seraddr))<0)
```

```
        perror("Bind failed...!");
    else
        printf("Bound Successfully...!\n");

    //listen for connections
    listen(sockfd,3);

    socklen_t len=sizeof(client1);
    //Accept connection from client
    newfd=accept(sockfd,(struct sockaddr*)&client1,&len);

    //Read client message
    n=read(newfd,fname,sizeof(fname));

    printf("File to be transferred is : %s\n",fname);
    fd=open(fname,O_RDONLY);

    if(fd<0){
        strcpy(buff,"File does not exist..!!");
        printf("File does not exist..!!\n");
    }
    else{
        bzero(buff,sizeof(buff));
        read(fd,buff,1024);
        close(fd);
        printf("File transferred..!!!\n");
    }

    n=write(newfd,buff,sizeof(buff));
    close(newfd);
    close(sockfd);

    return 0;
}
```

CLIENT SIDE:

```c
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<fcntl.h>
#include <unistd.h> //for read write and close
#include <arpa/inet.h> //for inet_addr()

int main(){

	int sockfd,n,fd;
	struct sockaddr_in seraddr;
	char buff[1024],fname[100];

	//Socket Creation
	sockfd=socket(AF_INET,SOCK_STREAM,0);
	if(sockfd<0)
		perror("Socket Creation failed...!");
	else
		printf("Socket Created Successfully...!\n");

	bzero(&seraddr,sizeof(seraddr));

	seraddr.sin_family=AF_INET;
	seraddr.sin_port=htons(3335);
	seraddr.sin_addr.s_addr=inet_addr("127.0.0.1");

	//Connect to a socket
	connect(sockfd,(struct sockaddr *)&seraddr,sizeof(seraddr));

	//Send message
	printf("Enter the path of file : ");
	scanf("%s",fname);

	n=write(sockfd,fname,sizeof(fname));
	n=read(sockfd,buff,sizeof(buff));
```

```
	if(strcmp(buff,"File does not exist..!!")==0){
		printf("%s\n",buff);
	}
	else{

		printf("Enter path to save file: ");
		scanf("%s",fname);

		//writing the buffer msg into the file
		fd=open(fname,O_CREAT | O_WRONLY | S_IRWXU);
		n=write(fd,buff,sizeof(buff));

		printf("File transferred..!!!\n");
	}

	close(sockfd);

	return 0;
}
```

OUTPUT :

SERVER SIDE :

```
msml@MSMLs-MacBook-Pro 4-ex % gcc server.c -o s
msml@MSMLs-MacBook-Pro 4-ex % ./s
Socket Created Successfully...!
Bound Successfully...!
File to be transferred is : /Users/msml/Desktop/samp.txt
File transferred..!!!
msml@MSMLs-MacBook-Pro 4-ex %
```

CLIENT SIDE :

```
msml@MSMLs-MacBook-Pro 4-ex % gcc client.c -o c
msml@MSMLs-MacBook-Pro 4-ex % ./c
Socket Created Successfully...!
Enter the path of file : /Users/msml/Desktop/samp.txt
Enter path to save file: /Users/msml/Desktop/newfile.txt
File transferred..!!!
msml@MSMLs-MacBook-Pro 4-ex %
```

SOURCE/DESTINATION FOLDER BEFORE EXECUTION:

```
msml@MSMLs-MacBook-Pro Desktop % ls *.txt
samp.txt
msml@MSMLs-MacBook-Pro Desktop %
```

SOURCE/DESTINATION FOLDER AFTER EXECUTION:

```
msml@MSMLs-MacBook-Pro Desktop % ls *.txt
newfile.txt     samp.txt
msml@MSMLs-MacBook-Pro Desktop % cat samp.txt
this is a file used for file transfer
exercise 4 of CN LAB
Can't write anymore
Bye.
msml@MSMLs-MacBook-Pro Desktop % cat newfile.txt
this is a file used for file transfer
exercise 4 of CN LAB
Can't write anymore
Bye.
msml@MSMLs-MacBook-Pro Desktop %
```

Learning Outcomes:

- I have learnt to create a socket program in C.
- I have learnt about various system calls used in socket programming.
- I have learnt file transfer system calls.
- I have learnt to create a TCP server/client program to transfer contents of a file from server to client.