Exercise 2b: File Transfer using TCP

Aim:

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

Algorithm:

Client:

1. Start
2. Create a socket using socket() system call.
3. Connect it to the server.
4. Prompt the user to enter the file name.
5. Transfer the file name to the server.
6. Receive the contents of the file and save in a new location.
7. Close the socket.
8. Stop

Server:

1. Start
2. Create a socket using socket() system call.
3. Bind the created socket with the port.
4. Listen for the connections.
5. When the server receives the file name from the client, read the contents and send the contents to the client.
6. Stop

Code:

/* C program to implement FTP using TCP */

```
//Client
#include <netdb.h>
#include <unistd.h>
#include <arpa/inet.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 verification
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd == -1)
        {
                fprintf(stderr,"Socket creation failed!\n");
                exit(0);
        }
        else
                printf("Socket creation successful!\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);
}

//Server
#include <stdio.h>
#include <unistd.h>
#include <arpa/inet.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
```

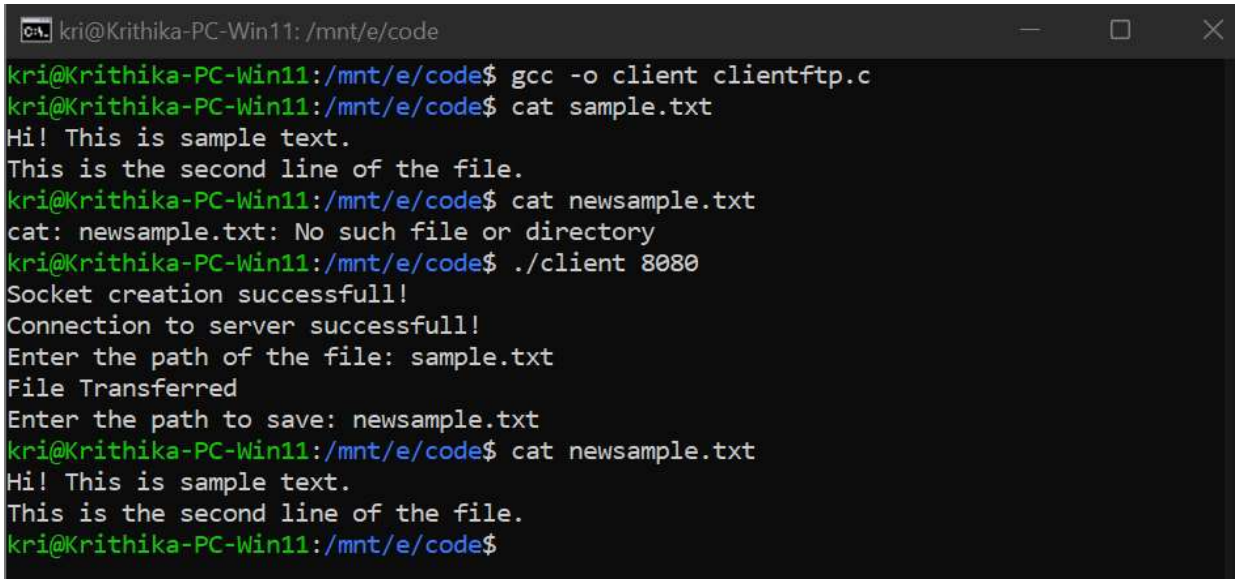
```
#define SA struct sockaddr
int main(int argc, char ** argv)
{
        if(argc < 2){
      fprintf(stderr, "Please pass port number for server as second argument!\n");
      exit(EXIT_FAILURE);
   }

        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 successful!\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 accept failed!\n");
        exit(EXIT_FAILURE);
    }
    else
        printf("Accept Successful!\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);
}
```

Output:

Server side:

Client side:



Learning outcomes:

- The process of establishing a connection between a client and a server was understood.
- A client-server connection was established and implemented.
- Sockets were used to establish a client-server connection.
- File transfer was implemented using TCP.