Distributed System Labwork 1

Group 6 - ICT

University of Science and Technology of Hanoi

January, 2022

Contents

1 Introduction

1.1 Overview

Based on the given chat system, we try to develop a file transfer via TCP/IP in CLI in this labwork.

1.2 Protocol

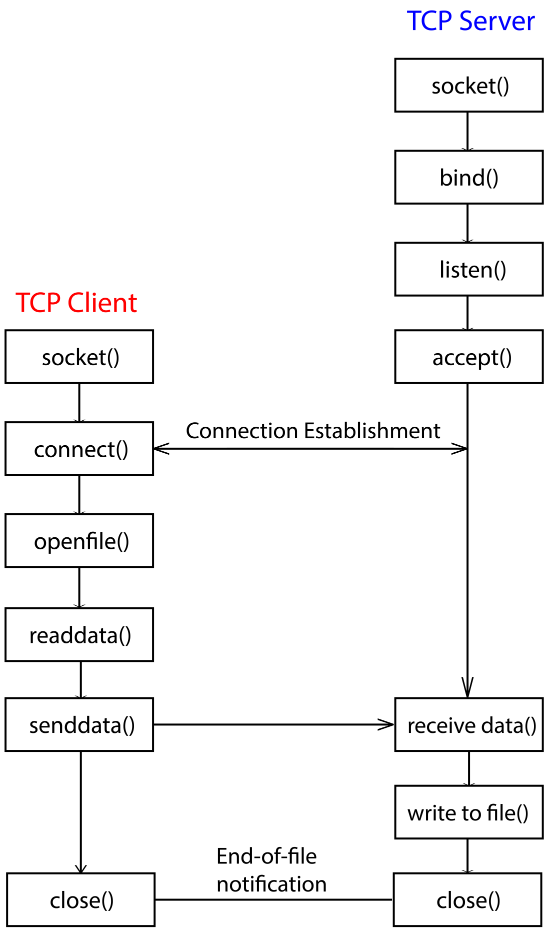


Figure 1: Protocol diagram

1.3 System organization

The server establishes a unique port, which in our case is 12345, which was provided by our teacher. It accepts one argument, the IP address, from the client CLI. A single client connects to a single server. The client sends data through a buffer, which is a character array. The buffer's maximum size is 1024 bytes. The server will write the buffer to the file after receiving it. After reaching the end-of-file, the client will notify the server that there is nothing remaining. Both the server and the client will then shutdown.

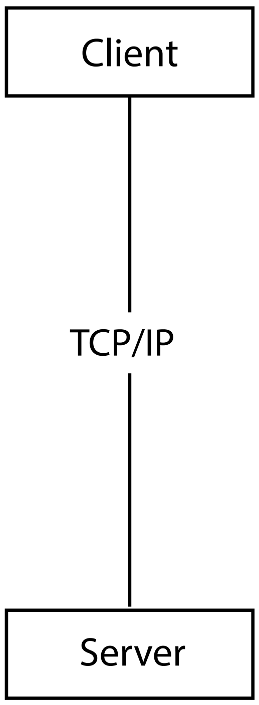


Figure 2: System organization

1.4 Implementation

From the client, we type gcc client.c -o client. Then type ./client localhost We have implemented the client side:

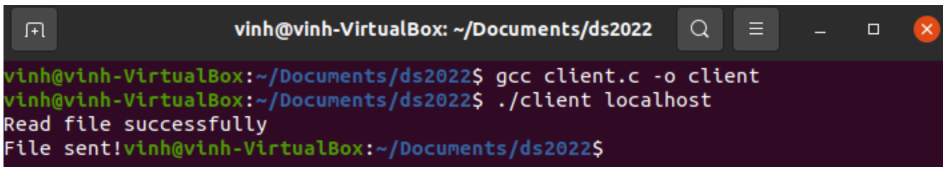
```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>

int main(int argc, char* argv[]) {
    int so;
    char s[100];
    struct sockaddr_in ad;
    socklen_t ad_length = sizeof(ad);
    struct hostent *hep;

    // create socket
    int serv = socket(AF_INET, SOCK_STREAM, 0);

    // init address
    hep = gethostbyname(argv[1]);
    memset(&ad, 0, sizeof(ad));
    ad.sin_family = AF_INET;
    ad.sin_addr = *(struct in_addr*)hep->h_addr_list[0];
    ad.sin_port = htons(12345);

    // connect to server
    connect(serv, (struct sockaddr*)&ad, ad_length);

    memset(&s, 0, 100);
    FILE* file;
    file = fopen("send_file.txt", "r");
    if (file == NULL) {
            printf("The file is null");
    } else {
            printf("Read file successfully\n");
    }

    char buffer[1024] = {0};
    while (fgets(buffer, sizeof(buffer), file) != NULL) {
            int i = send(serv, buffer, sizeof(buffer), 0);
            if (i == -1) {
                    printf("Send data fail");
            }
            memset(&buffer, 0, sizeof(buffer));
    }
    printf("File sent!");

    close(serv);
    return 0;
}
```

From the server, we type gcc server.c -o server. Then type ./server. We have implemented the server side:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <unistd.h>
```

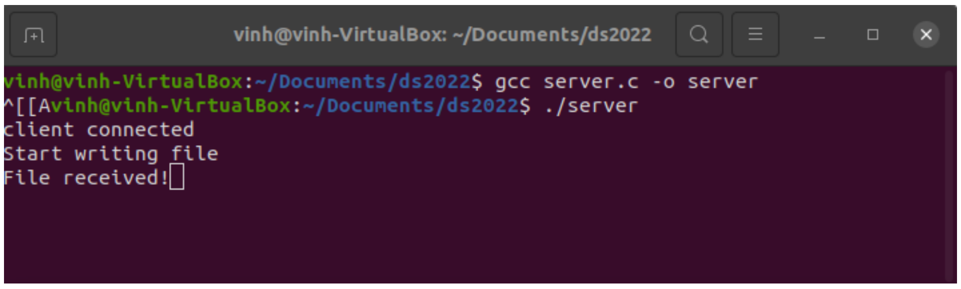
```
int main() {
    int ss, cli, pid;
    struct sockaddr_in ad;
    char s[100];
    socklen_t ad_length = sizeof(ad);

    // create the socket
    ss = socket(AF_INET, SOCK_STREAM, 0);

    // bind the socket to port 12345
    memset(&ad, 0, sizeof(ad));
    ad.sin_family = AF_INET;
    ad.sin_addr.s_addr = INADDR_ANY;
    ad.sin_port = htons(12345);
    bind(ss, (struct sockaddr*)&ad, ad_length);

    // then listen
    listen(ss, 0);

    while (1) {
        // an incoming connection
        cli = accept(ss, (struct sockaddr*)&ad, &ad_length);

        pid = fork();
        if (pid == 0) {
            // I'm the son, I'll serve this client
            printf("client connected\n");

            FILE* file;
            file = fopen("recieve_file.txt", "w");
            if (file == NULL) {
                    printf("Cannot open file");
            } else {
                    printf("Start writing file\n");
            }

            char buffer[1024];
            while (1) {
                int i = recv(cli, buffer, sizeof(buffer), 0);
                if (i <= 0) {
                        break;
                }
                fprintf(file, "%s", buffer);
                memset(&buffer, 0, sizeof(buffer));
            }
            printf("File received!");
            return 0;
        } else {
            // I'm the father, continue the loop to accept more clients
            continue;
        }
    }
    // disconnect
    close(cli);
    close(ss);

}
```

Here is the result:



```
vinh@vinh-VirtualBox:~/Documents/ds2022$ gcc client.c -o client
vinh@vinh-VirtualBox:~/Documents/ds2022$ ./client localhost
Read file successfully
File sent!vinh@vinh-VirtualBox:~/Documents/ds2022$
```

Figure 3: Client side



```
vinh@vinh-VirtualBox:~/Documents/ds2022$ gcc server.c -o server
^[[Avinh@vinh-VirtualBox:~/Documents/ds2022$ ./server
client connected
Start writing file
File received!
```

Figure 4: Server side

1.5 Contribution

Member	Contribution
Nguyen Quang Vinh	Client code
Nguyen Tran Nguyen	Server code
Mai Xuan Hieu	Design Protocol
Nguyen Anh Quan	Design Architecture
Nguyen Tuong Quynh	Report

Table 1: Contribution Table