哈爾濱工業大學

网络安全实验报告

题　　目 基于 socket 的扫描器设计

专　　业 信息安全

学　　号 1190202401

学　　生 陈豪

指 导 教 师 王彦

一、实验目的

熟悉 socket 编程，可以利用 socket 编程编写基于 linux 平台的 C/S 程序和基于 windows 平台的扫描器。

二、实验内容

1. 熟悉 Linux 编程环境
2. 在 Windows 机器上安装 Linux 虚拟机
3. 在 Linux 环境下编写 C/S 程序，熟悉 socket 编程。要求客户端和服务器端能够传送指定文件。该程序在后续实验中仍需使用。客户端与服务器端在不同的机器中。
4. 在 Windows 环境下利用 socket 的 connect 函数进行扫描器的设计，要求有界面，界面能够输入扫描的 ip 范围和端口范围，和需使用的线程数，显示结果。
5. 实验课的时候，检验结果和现场截图，为撰写实验报告做准备。

三、实验过程

(一) Linux 环境下的 C/S 程序

实验基本信息：

实验环境：Ubuntu 20.04 x64 编程语言：C

1. 需求分析

需要在两台 linux 虚拟机之间传送文件，所以需要给两台 linux 虚拟机都配置一个可以访问的 ip。

程序功能：

(1)客户端：

a.可以向服务端发送一个本目录下指定的文件，文件名由用户输入；

b.可以从服务端下载一个服务端目录下的文件，文件名由用户输入；

(2)服务端：

a.可以监听来自客户端的连接请求；

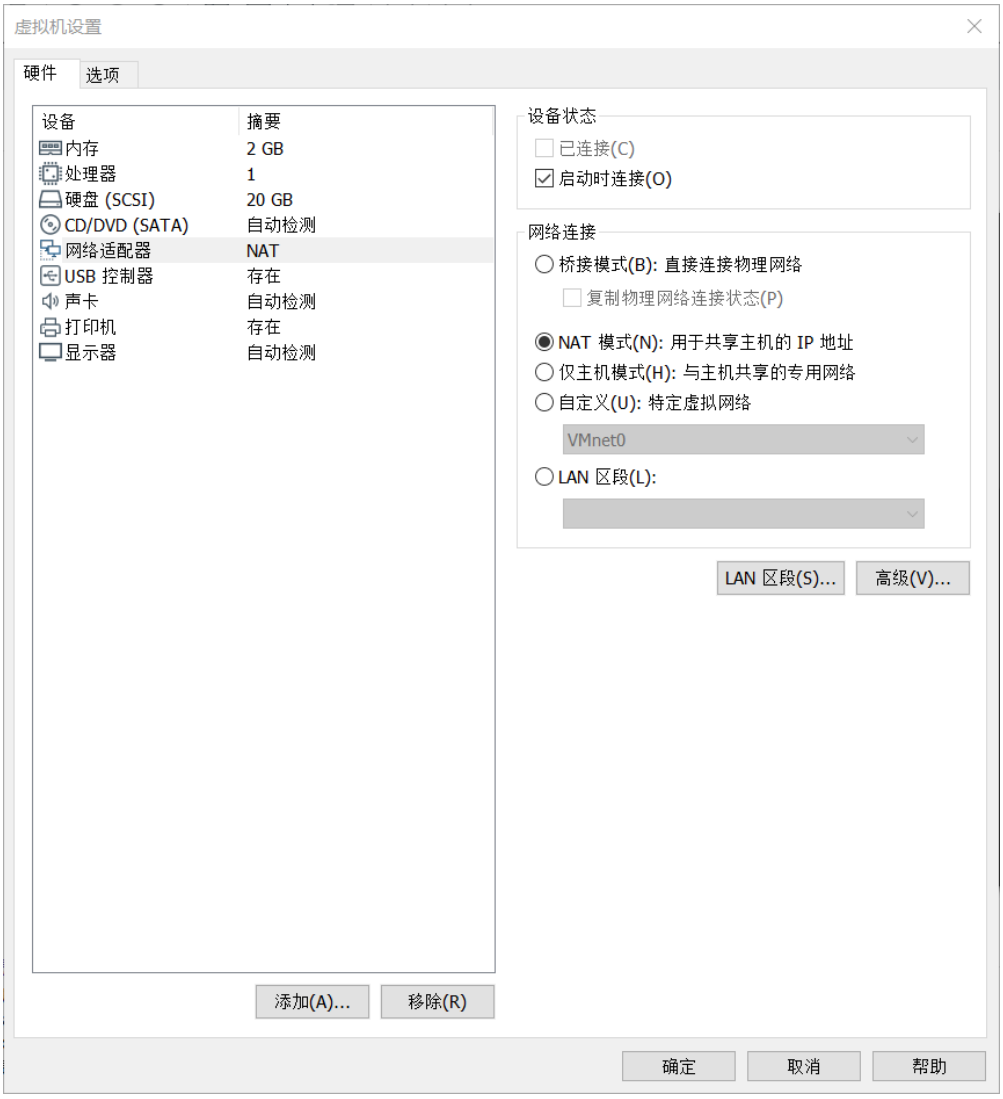
b.可以接收客户端传送的文件；

c.可以向客户端传送一个指定的文件，文件由客户端给出。

(3)传送文件要求：任何二进制文件。

2. 环境配置

配置虚拟机的网卡即可：



3. 客户端编写

新建了一个结构体定义如下：

```
17  typedef struct NetworkFileTool NetworkFileTool;
18  typedef int func(NetworkFileTool *);
19
20  struct NetworkFileTool{
21      int sock;//sockect description
22      struct sockaddr_in* addr;//address
23      int connect_flag;
24      char* buf;
25
26      func *msend_file;
27      func *mrecevie_file;
28      func *mget_directory;//client get file directory from server
29      func *mdelete;//delete the struct
30  };
```

实现中 msend_file 用于发文件，剩下的三个函数指针并未实现。

```c
/**
 * @brief parse for destip and destport
 *
 * @param argc the number of argument
 * @param argv the argument array
 * @param ip the ip address
 * @param port the port between 1 and 65535
 */
void parse_command_line(int argc,char **argv,char *ip,u_int16_t* port){
    printf("argc = %d\n",argc);
    if(argv[1] != NULL && valid_String(argv[1],15)){
        strcpy(ip,argv[1]);
    }else{
        strcpy(ip,DEFAULT_IP);
    }
    if(argv[2] != NULL && atoi(argv[2]) <= 65535 && atoi(argv[2]) >=1){
        *port = atoi(argv[2]);
    }else{
        *port = DEFAULT_PORT;
    }
    printf("ip:%s,port:%d\n",ip,*port);
}
```

```c
void create_tcp_socket(NetworkFileTool* networkFiletool,const char * ip,const u_int16_t port){
    //创建套接字
    networkFiletool->sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    //将套接字和IP、端口绑定
    networkFiletool->addr = (struct sockaddr_in*)malloc(sizeof(struct sockaddr_in));
    networkFiletool->addr->sin_family = AF_INET;  //使用IPv4地址
    networkFiletool->addr->sin_addr.s_addr = inet_addr(ip);  //具体的IP地址
    networkFiletool->addr->sin_port = htons(port);  //端口
    networkFiletool->connect_flag = connect(networkFiletool->sock,(struct sockaddr*)networkFiletool->addr,
        sizeof(struct sockaddr));
}
```

编写了 parse_command_line 和 create_tcp_socket 函数用于从命令行获取 ip 和端口并创建相应的套接字。

```c
NetworkFileTool* create_client(const char *ip,const u_int16_t port){

    DEBUG(printf("sizeof(NetworkFileTool) = %ld\n",sizeof(NetworkFileTool)));

    NetworkFileTool* client = (NetworkFileTool*) malloc(sizeof(NetworkFileTool));
    create_tcp_socket(client,ip,port);
    client->msend_file = &send_file;
    return client;
}
```

create_client 函数用于创建并初始化结构体。

主要的处理函数如下：

```c
void simple_cli(NetworkFileTool* client){
    printf("-----------------------------------\n");
    printf("what do you want to do?\n");
    printf("1.download file from server\n");
    printf("2.upload file to server\n");
    printf("please input (1 or 2 for select) or (0 for exit):");
    int select = -1;
    scanf("%d",&select);
    while(select != 1 && select != 2 && select != 0){
        printf("invalid input,please input again (1 or 2 for select) or (0 for exit):");
        if(scanf("%d",&select)!=1){
            //set buf 0
            char c;
            while ((c = getchar()) != EOF && c != '\n');
        }
    }
    char filename[50];
    if(select == 1){
        printf("please input file name to download:");
        if(scanf("%s",filename) == 1 && valid_FileName(filename)){
            client->buf = malloc(DEFAULT_BUFF_SIZE);
            memset(client->buf,0,DEFAULT_BUFF_SIZE);
            char str[100];
            sprintf(str, "download filename=%s end", filename);
            strcpy(client->buf,str);
            client->msend_file(client);
            while(recv(client->sock,client->buf,DEFAULT_BUFF_SIZE,0)<=0);

            if(!strncmp(client->buf,"upload",6)){
                int file_size = atoi(client->buf+16);
                char filename[DEFAULT_FILENAME_SIZE];
                memset(filename,0,DEFAULT_FILENAME_SIZE);
                strncpy(filename,strstr(client->buf,"filename=")+9,(strstr(client->buf," end"))-(strstr(client->buf,"filename=")+9)
                printf("received file_size = %d\n",file_size);
                printf("received filename = %s\n",filename);
                int remain_data = file_size;
                FILE* file = fopen("dest","w");

                char* real_start = strstr(client->buf," end")+4;
                fwrite(real_start,sizeof(char), strlen(real_start), file);
                int len = strlen(real_start);
                remain_data -= len;
                fprintf(stdout, "Receive %d bytes and we still hope : %d bytes\n", len, remain_data);
                while((remain_data > 0) && ((len = recv(client->sock, client->buf, DEFAULT_BUFF_SIZE, 0)) > 0)){
                    fwrite(client->buf ,sizeof(char), len<remain_data?len:remain_data, file);
                    remain_data -= len;
                    fprintf(stdout, "Receive %d bytes and we still hope : %d bytes\n", len<remain_data?len:(remain_data<0?0:remain_
                }
            }else{
                printf("client->buf = %s",client->buf);
            }


        }
    }else if(select == 2){
        printf("please input file name to upload:");
        if(scanf("%s",filename) == 1 && valid_FileName(filename)){
            FILE* file;
            if((file = fopen(filename,"r"))!=NULL){
                printf("uploading %s to server\n",filename);

                fseek(file,0,SEEK_END);
                int file_size = ftell(file);
                fseek(file,0,SEEK_SET);

                client->buf = malloc(DEFAULT_BUFF_SIZE);
                memset(client->buf,0,DEFAULT_BUFF_SIZE);

                char str[100];
                sprintf(str, "upload filesize=%d,filename=%s end", file_size,filename);
                strcpy(client->buf,str);
                client->msend_file(client);

                while(fread(client->buf,sizeof(char),DEFAULT_BUFF_SIZE,file)>0 ){
                    DEBUG(printf("%s",client->buf));
                    client->msend_file(client);

                }

                printf("the file size is %d\n",file_size);
                close(client->sock);
                fclose(file);
            }else{
                printf("%s does not exist\n",filename);
            }

        }
    }
}

int main(int argc,char *argv[]) {
```

首先从提示选择是下载还是上传，接着要求客户输入文件名。在上传的时候，由于 fread 函数不会清空缓存区会导致最后一次读取时如果没读完将把上一次读取的内容一起上传，因

此这里先使用了 fseek 和 ftell 获取文件大小。实际还可以选择每次上传一次后就使用 memset 清空缓存区。下载同服务器下载相同原理。

4. 服务端编写

```
void simple_cli(NetworkFileTool* server){
    printf("-------------------------------------\n");
    DEBUG(printf("server->sock = %d\n",server->sock));
    struct sockaddr_in *ipv4 = (struct sockaddr_in *)server->addr;
    char ipAddress[INET_ADDRSTRLEN];
    inet_ntop(AF_INET, &(ipv4->sin_addr), ipAddress, INET_ADDRSTRLEN);
    printf("The IP address is: %s\n", ipAddress);
    if(bind(server->sock,(struct sockaddr*)server->addr,sizeof(struct sockaddr))==0){
        listen(server->sock,20);//进入监听状态，等待用户发起请求
        struct sockaddr_in client_addr;
        socklen_t client_addr_size = sizeof(client_addr);
        int client_sock = accept(server->sock,(struct sockaddr*)&client_addr,&client_addr_size);
        server->buf = malloc(DEFAULT_BUFF_SIZE);
        memset(server->buf,0,DEFAULT_BUFF_SIZE);

        recv(client_sock,server->buf,DEFAULT_BUFF_SIZE,0);
        if(!strncmp(server->buf,"upload",6)){
            int file_size = atoi(server->buf+16);
            char filename[DEFAULT_FILENAME_SIZE];
            memset(filename,0,DEFAULT_FILENAME_SIZE);
            strncpy(filename,strstr(server->buf,"filename=")+9,(strstr(server->buf," end"))-(strstr(server->buf,"filename=")+9));
            printf("uploaded file_size = %d\n",file_size);
            printf("uploaded filename = %s\n",filename);
            int remain_data = file_size;
            FILE* file = fopen("dest","w");

            char* real_start = strstr(server->buf," end")+4;
            fwrite(real_start,sizeof(char), strlen(real_start), file);
            int len = strlen(real_start);
            remain_data -= len;
            fprintf(stdout, "Receive %d bytes and we still hope : %d bytes\n", len, remain_data);
            while((remain_data > 0) && ((len = recv(client_sock, server->buf, DEFAULT_BUFF_SIZE, 0)) > 0)){
                fwrite(server->buf ,sizeof(char), len<remain_data?len:remain_data, file);
                remain_data -= len;
                fprintf(stdout, "Receive %d bytes and we still hope : %d bytes\n", len<remain_data?len:(remain_data<0?0:remain_dat
            }
        }else if(!strncmp(server->buf,"download",8)){
            char filename[DEFAULT_FILENAME_SIZE];
            memset(filename,0,DEFAULT_FILENAME_SIZE);
            strncpy(filename,strstr(server->buf,"filename=")+9,(strstr(server->buf," end"))-(strstr(server->buf,"filename=")+9));
            FILE* file;
            if((file = fopen(filename,"r"))==NULL){
                perror("file open error:");
            }

            printf("uploading %s to client\n",filename);

            fseek(file,0,SEEK_END);
            int file_size = ftell(file);
            fseek(file,0,SEEK_SET);
            char str[100];

            sprintf(str, "upload filesize=%d,filename=%s end", file_size,filename);

            strcpy(server->buf,str);
            DEBUG(printf("server->buf = %s\n",server->buf));

            send(client_sock,server->buf,strlen(server->buf),0);
            memset(server->buf,0,DEFAULT_BUFF_SIZE);
            while(fread(server->buf,sizeof(char),DEFAULT_BUFF_SIZE,file)>0 ){
                DEBUG(printf("%s\n",server->buf));
                send(client_sock,server->buf,strlen(server->buf),0);
                memset(server->buf,0,DEFAULT_BUFF_SIZE);
            }

            printf("the file size is %d\n",file_size);
            close(server->sock);
            close(client_sock);
            fclose(file);
        }


    }else{
        printf("server bind false");
    }

}
```

服务器程序上传方面会让客户端发来文件名字和大小，以方便进行文件的大小校对。

（二）Windows 环境下的扫描器程序

实验基本信息：

实验环境：Windows10 x64

vscode

编程语言：python

1. 需求分析

实验指导中要求编写界面，由于不会使用 C 语言编写界面，所以使用 python 加 qtpython 编写界面。

程序功能：

(1)用户可以输入需要扫描的 ip 范围、端口范围和想使用的线程数，输入框中只能输入合法的字符；

(2)如果用户输入的错误，将不予执行。

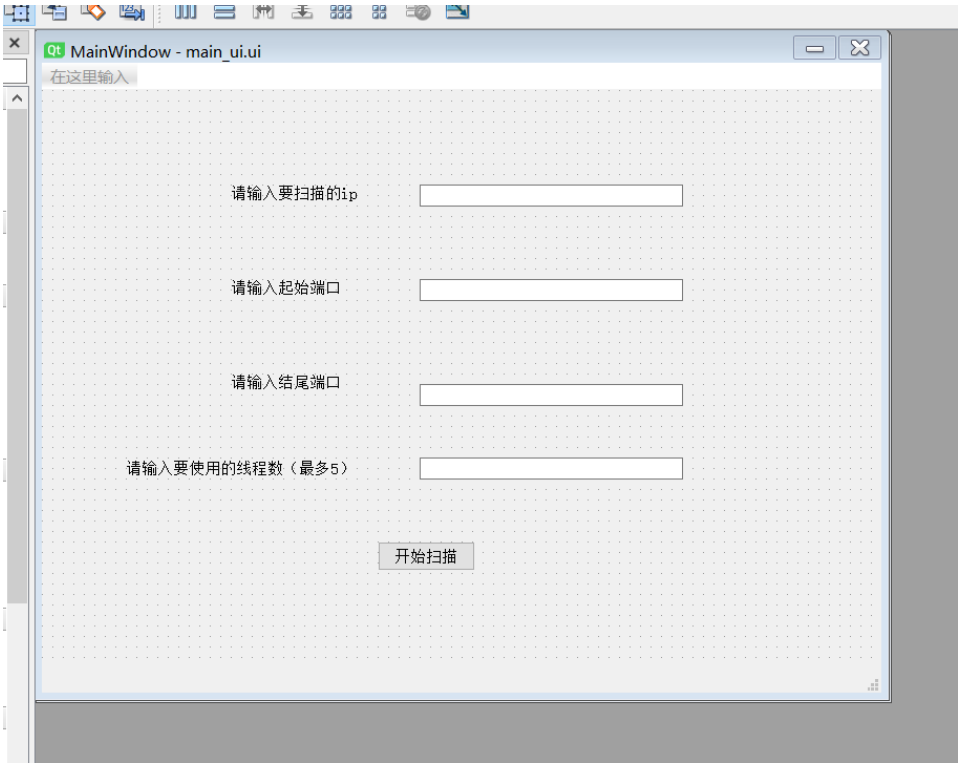
(3)当所有输入都正确无误后，按下开始扫描，程序开始扫描用户指定的 ip 和端口；

(4)关于扫描的线程分配：

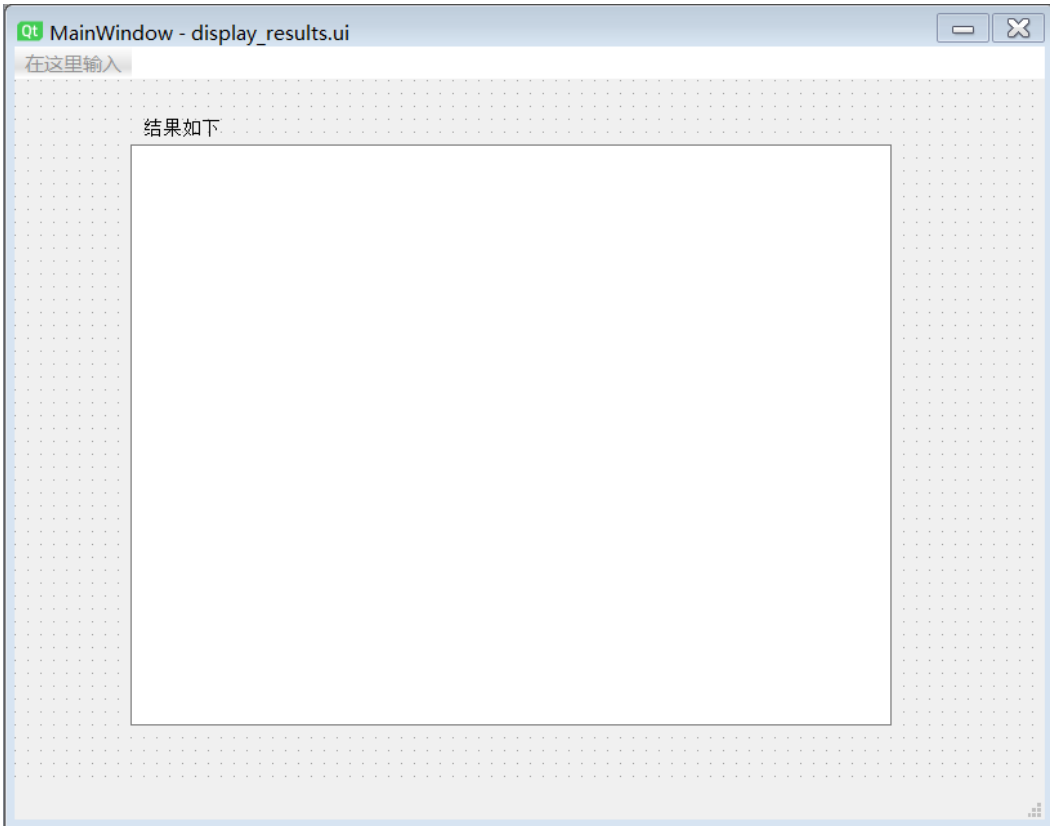
采用的方案是总端口除以总线程数，然后给每个线程分配端口范围。

2. 界面编写

界面使用 qtpython 自带的 designer 软件只需拖一拖就可以实现实现界面了。



主界面



输出界面

3. 控件逻辑编写

为主界面的开始扫描按钮添加了点击事件

```
class MyWindow(StartUI, QMainWindow):
    def __init__(self) -> None:
        super(MyWindow, self).__init__()
        self.setupUi(self)
        self.pushButton.pressed.connect(self.get_args)

    def get_args(self):

        self.display = Display()
        try:
            self.scanner = Scanner(self.lineEdit.text(), int(self.lineEdit_2.text()),
                                   int(self.lineEdit_3.text()), int(self.lineEdit_4.text()), self.display.textEdit
            self.display.show()
            self.scanner.multiScan()

        except Exception as err:
            print(err)
```

当点击的时候就会创建输出界面并且开始多线程扫描。

4. 具体功能编写

(1) 扫描主方法

```
def __scan(self, startport, endport):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    for port in range(startport, endport+1):
        try:
            s.connect((self.ip, port))
            print('{0} port {1} is open'.format(self.ip, port))
            if self.output is not None:
                self.output.insertPlainText(
                    '{0} port {1} is open\n'.format(self.ip, port))
            s.close()
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        except Exception as err:
            print('{0} port {1} is not open'.format(self.ip, port))
            if self.output is not None:
                self.output.insertPlainText(
                    '{0} port {1} is not open\n'.format(self.ip, port))
    s.close()
```

对一定范围内的端口扫描，如果 connect 上了就关闭 socket 并重建一个，并打印 xx 端口 is open 否则打印 xx 端口 is not open。

(2) 扫描线程方法

```
def multiScan(self):
    tempstart = self.startport
    tempInternal = int((self.endport-self.startport)/self.threadNumbers)
    for i in range(self.threadNumbers):
        self.threads.append(threading.Thread(target=self.__scan, args=(
            tempstart, self.endport if (tempstart+tempInternal) > self.endport else (tempstart+tempInternal),)))
        self.threads[i].start()
        tempstart = tempstart+tempInternal+1
```

根据端口总数除以线程数进行扫描分配。

(3) 输入检查

```
def __check_ip(self):
    compile_ip = re.compile(
        '^(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-
9])\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|\d)\.(1\d{2}|2[0-4]\d|25[0-
5]|[1-9]\d|\d)\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|\d)$')
    if compile_ip.match(self.ip):
        return True
    else:
        return False
```

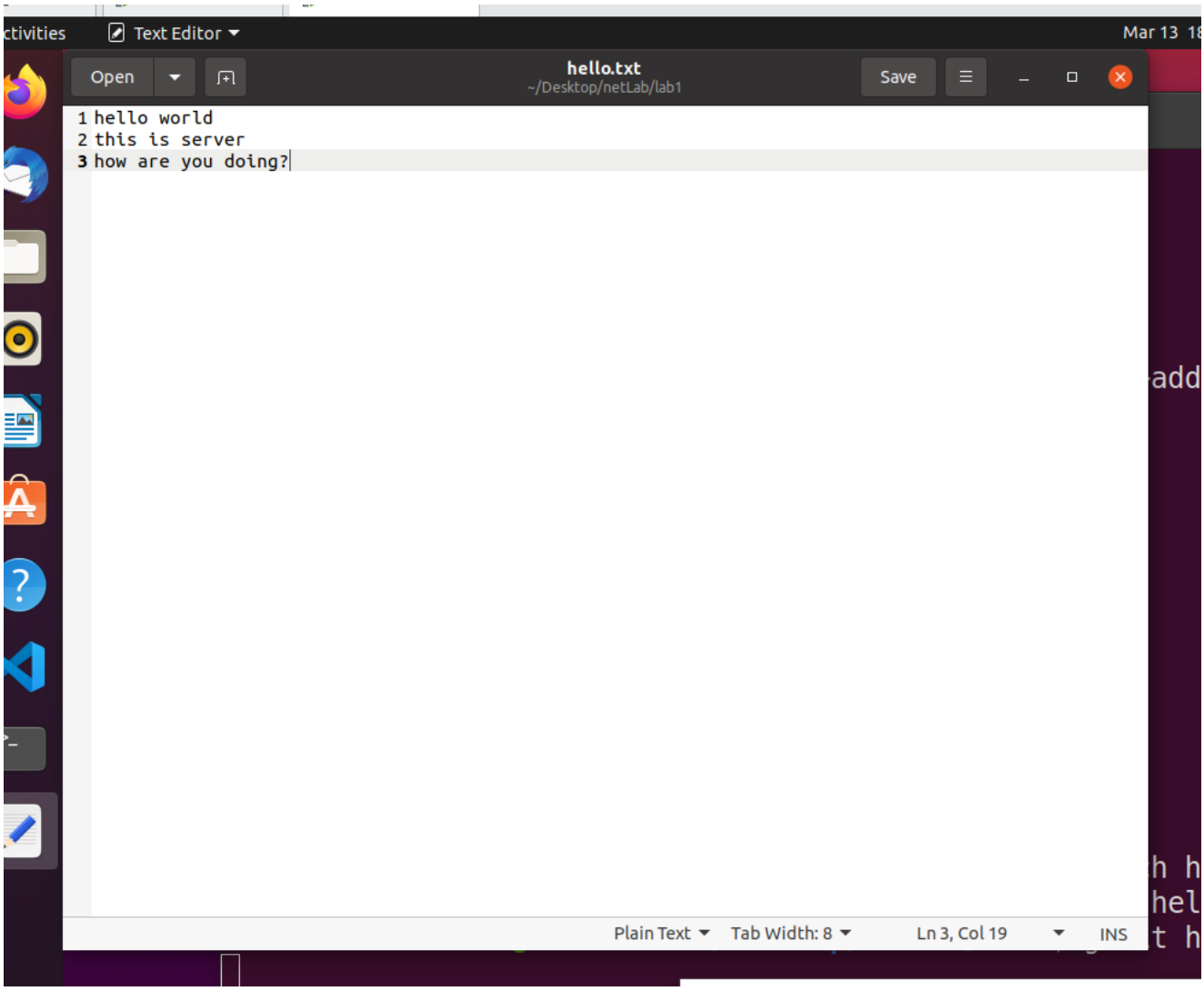
根据正则表达式检查 ip，对于端口的话就简单检查是否在 1 和 65535 之内。

四、实验结果

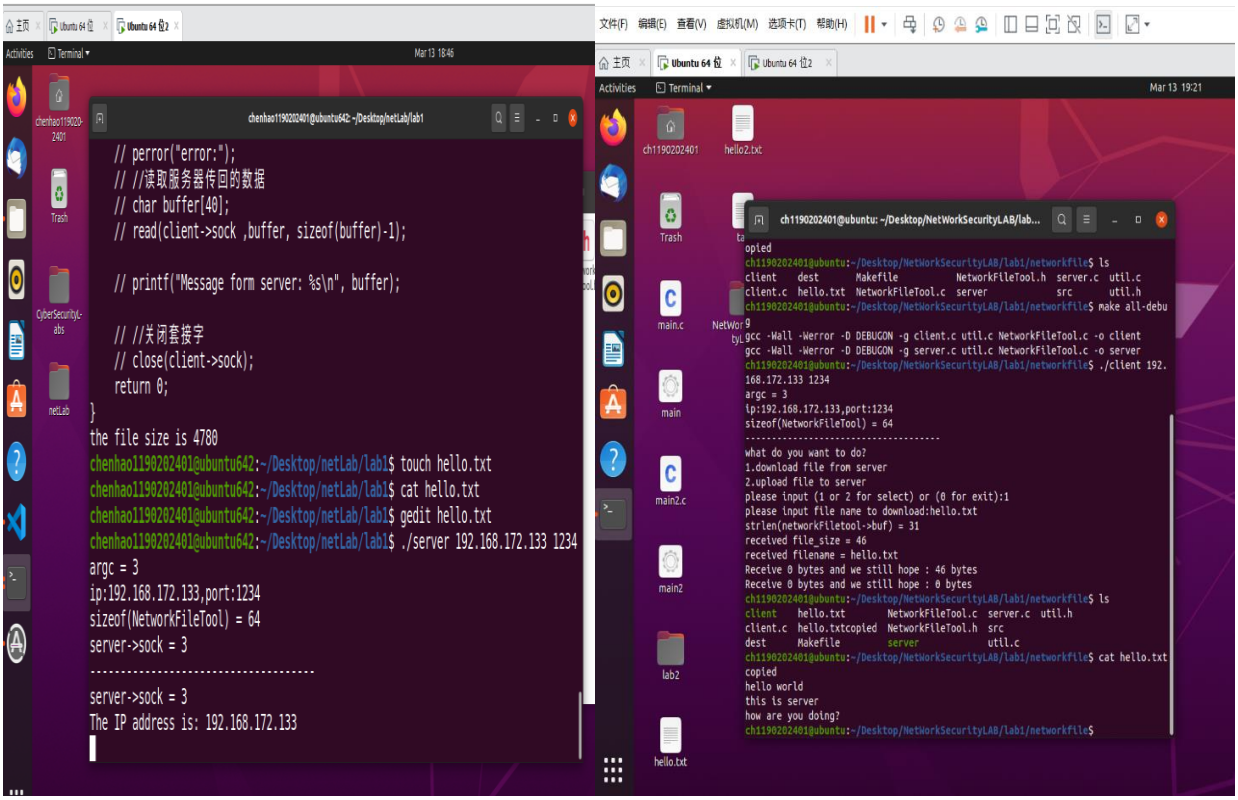
（一）Linux 环境下的 C/S 程序

首先在测试下载，在服务器端准备文件 hello.txt

其中的内容如下：


ctivities
Text Editor
Mar 13 18
Open
hello.txt
~/Desktop/netLab/lab1
Save
1 hello world
2 this is server
3 how are you doing?
Plain Text
Tab Width: 8
Ln 3, Col 19
INS

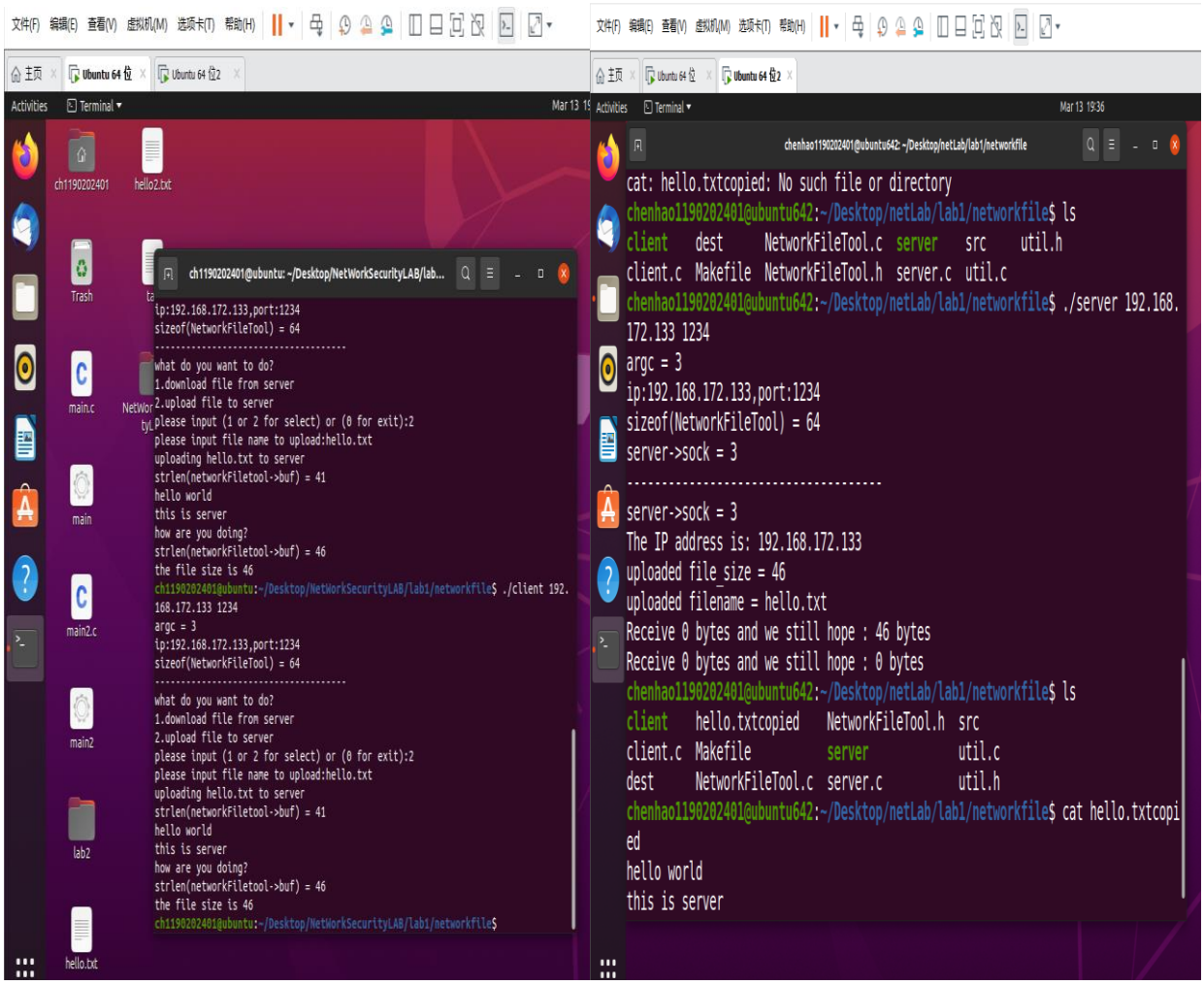
然后开始测试下载



图中左边为服务端，右边为客户端。

接下来测试上传

这一次在客户端上传 hello.txt 文件，服务器接收并写入到 hello.txtcopied 文件中

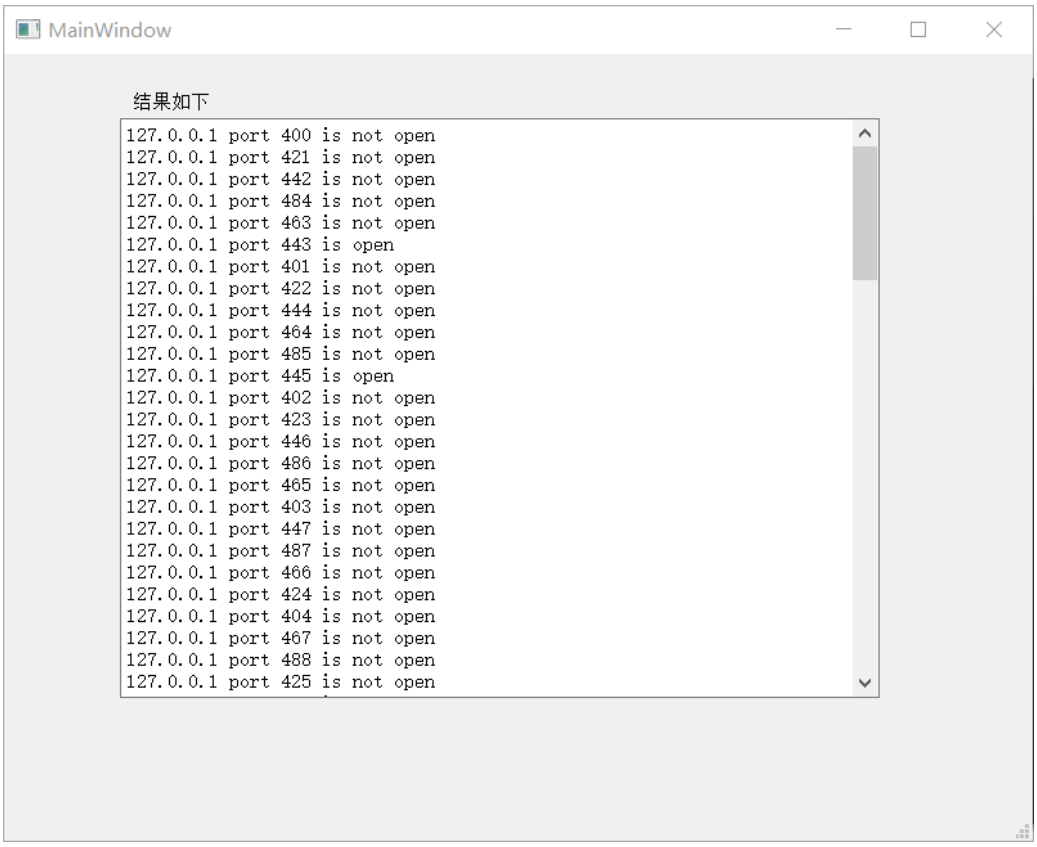


图左为客户端，图右为服务器

（二）Windows 环境下的扫描器程序

使用该程序扫描本地部分端口

部分结果如下：

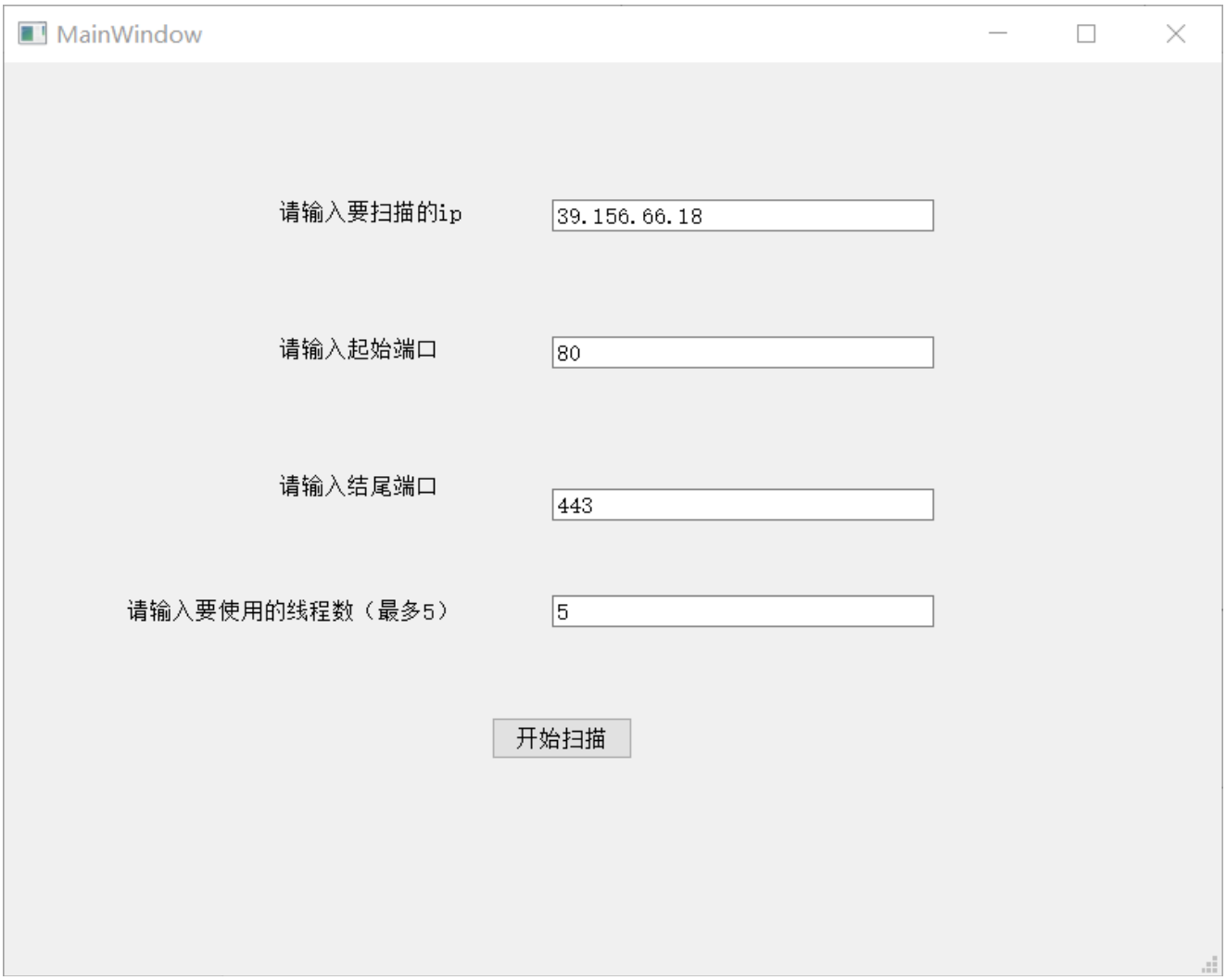


与使用 nmap 扫描的结果对比如下：

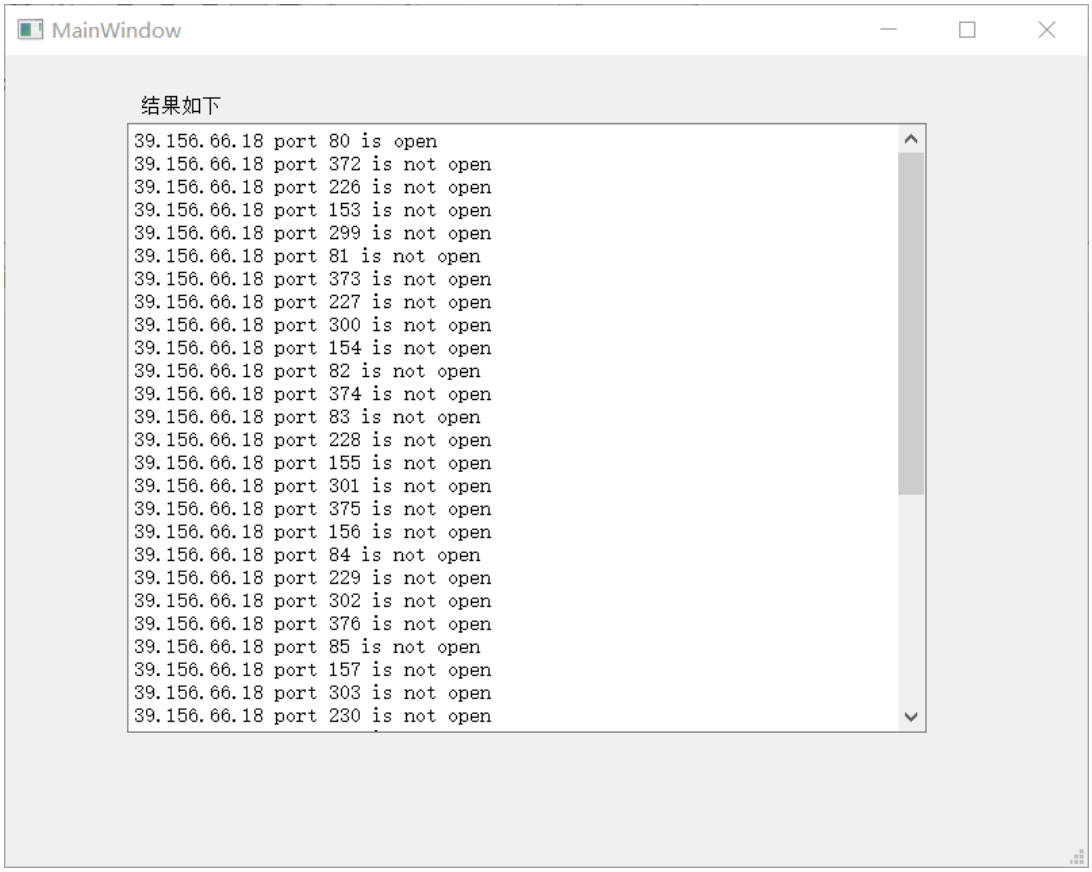
```
15815  ~  ♥ 13:05  nmap 127.0.0.1
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-14 13:05 ♦й♦♦♦▯^♦♦
Nmap scan report for 127.0.0.1
Host is up (0.0011s latency).
Not shown: 993 closed tcp ports (reset)
PORT      STATE SERVICE
135/tcp   open  msrpc
443/tcp   open  https
445/tcp   open  microsoft-ds
903/tcp   open  iss-console-mgr
3306/tcp  open  mysql
5357/tcp  open  wsdapi
55555/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 0.42 seconds
```

扫描百度端口如下：



部分结果如下：



使用 nmap 做对比：

```
15815  ~  ♥ 13:05  nmap 39.156.66.18
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-14 13:06 ◆Й◆◆◆◘ˆ◆◆
Nmap scan report for nxdomain (39.156.66.18)
Host is up (0.026s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 11.75 seconds
```