

网络安全实验报告

题 目 基于socket 的扫描器设计

专 业 视听觉信息处理 .

学 号 7203610121 .

学 生 刘天瑞 .

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，先从服务端获得文件名列表，再由用户输入需要的文件名。

(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。

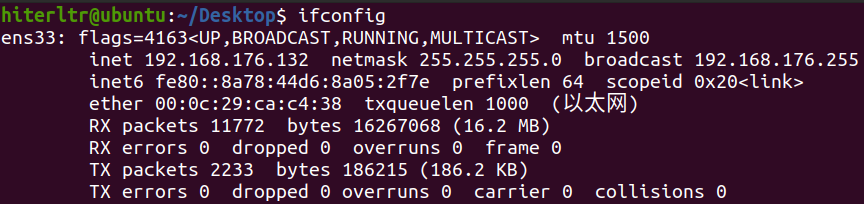
(3)传送文件要求：任何二进制文件。

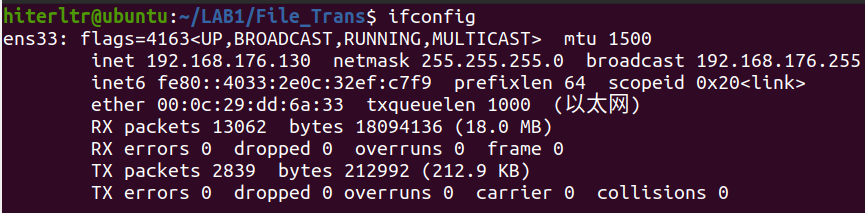
## 2. 环境配置

配置虚拟机的网卡即可：



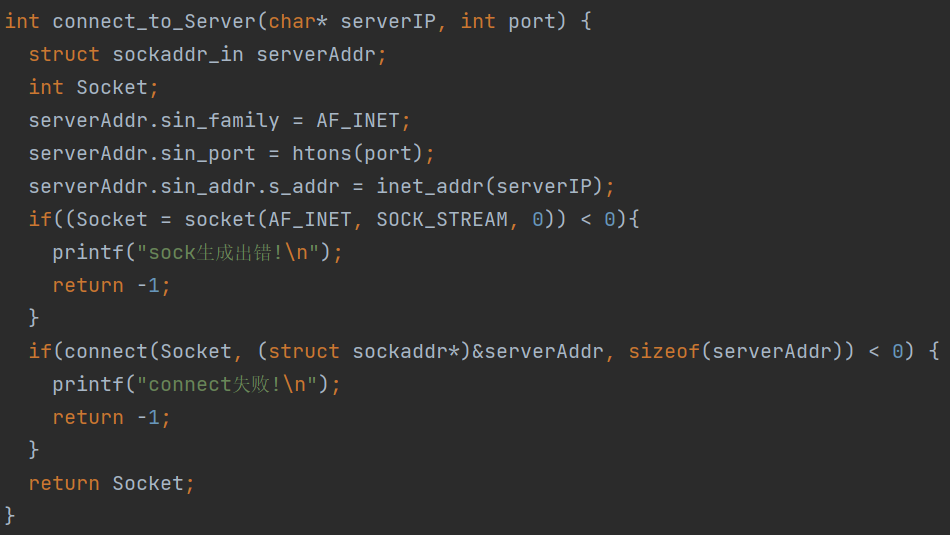
我继续克隆了一份虚拟机，使得两个虚拟机拥有不同的IP地址，从而便于在两个虚拟机之间编写代码进行测试，如下图所示。其中原虚拟机的IP地址为192.168.176.132，另一个克隆虚拟机的IP地址为192.168.176.130。



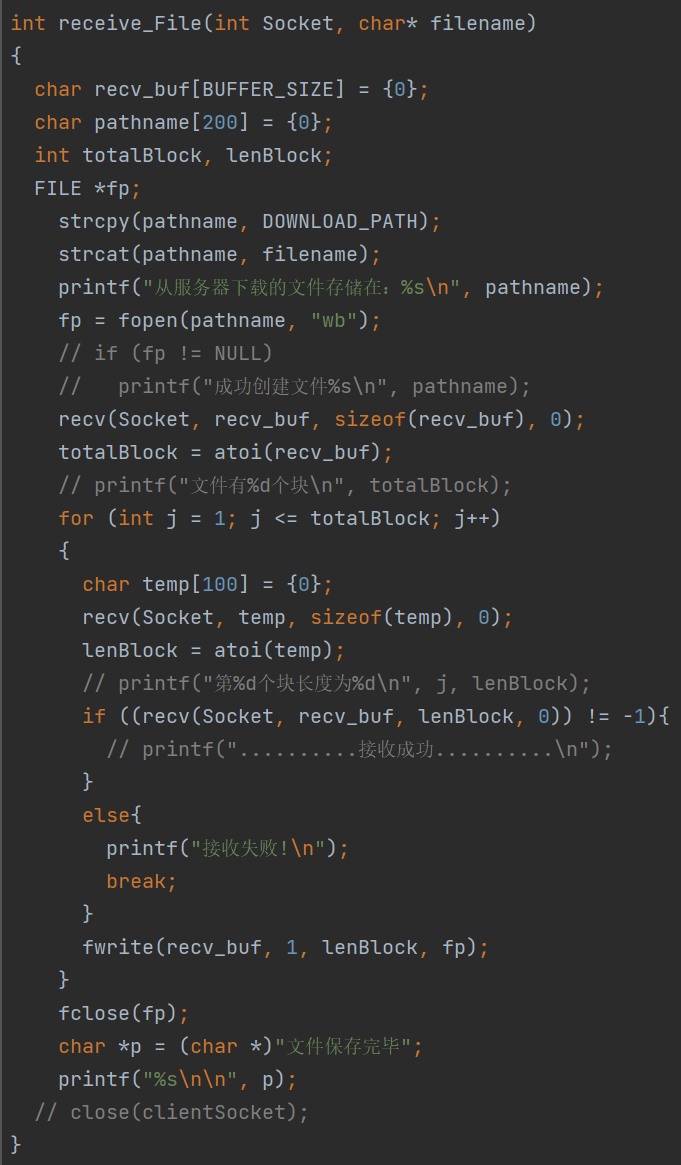


## 3. 客户端编写

在原虚拟机上编写客户端的代码：client.c，用于连接服务器端从而在服务器端的服务器上下载本地文件。具体运行的部分关键代码如下图所示：



首先创建客户端套接字，来访问服务器“192.168.176.132“（相对的另一方于是访问”192.168.176.132“）的任意端口（INADDR\_ANY）。



当TCP连接建立后，客户端便与服务器端进行两次通信交互：

1. 服务器端向客户端传输运行文件夹下的所有文件列表，传送多个文件，相向客户端询问需要下载的文件名；
2. 客户端查看本地文件列表，从中选择一个需要下载的文件，并且将文件名传送到服务器端。



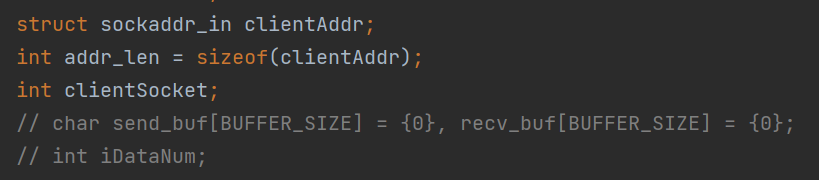
客户端发送文件名结束后就开始准备接收文件：首先创建临时缓存文件，并且持续从服务器端接收数据写入临时文件，数据传输完毕后，再将临时文件改名为客户端刚刚选择的文件名，至此文件传输完毕。

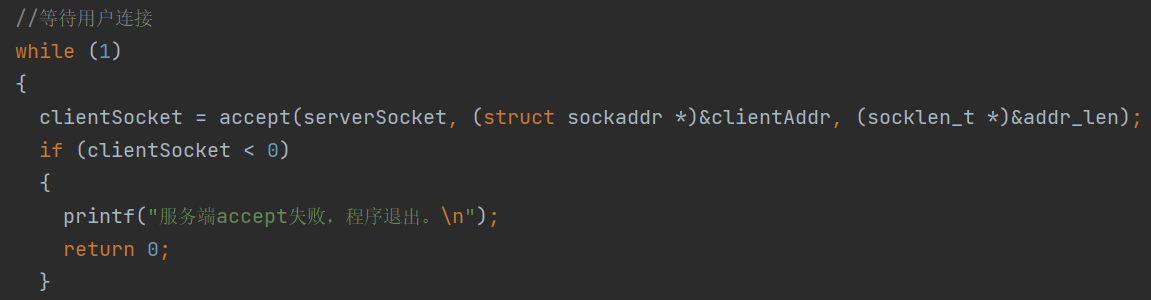
## 4. 服务端编写

在克隆虚拟机上编写服务器端的代码：server.c，其中部分运行的关键代码如下图所示：



首先便是创建服务器端的套接字，并且绑定了端口为1234。

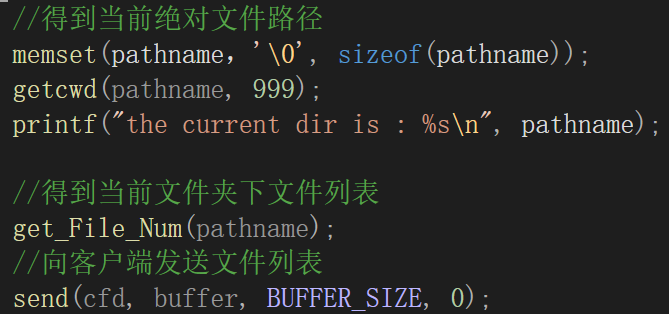




服务器端的监听端口为1234，并且采用了多线程处理来自客户端的请求连接，当有客户端连接时，使用pthread\_create()来创建新的子进程，，然后再转入线程处理函数client\_method()处理，相关参数作为addr来传入到该函数中。使主线程持续一直工作，并且持续等待新的客户端连接请求。除此之外服务器端还可以同时处理来自多个客户端的连接。



服务器端线程处理函数内的处理过程与客户端过程是一一对应的，其中使用get\_File\_Num()函数得到当前文件夹下（参数pathname为当前文件夹绝对路径，使用getcwd得到）的文件列表存储在buffer内。



**（二）Windows 环境下的扫描器程序**

## 实验基本信息：

实验环境：Windows11 x64

编程语言：Python 3.9.7 Tk8.0

## 1. 需求分析

实验指导中要求编写界面，可以使用 java，但是 java 编写界面过于麻烦，所以我选择了基于Python的Tkinter GUI来编写程序，Tkinter GUI的界面编写非常方便，且它独有的信号与槽机制能使很多操作变得方便。另外，在程序的设计各方面都追求人性化，用户误操作时会给出准确的提示信息。

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，其中 ip 范围跟平时在电脑上操作一样，输入三个数字后自动跳转到下一个输入框，输入框中只能输入合法的字符；
2. 如果用户在输入未完成的时候就按下了开始扫描按钮，提示输入未完成，如果用户输入的范围错误，提示范围错误；
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：

方案一：由于本人技术有限，采取先把 ip 和端口号一对一保存，根据 step=总端口数/线程数 给每个线程分配 step 个端口（最后一个线程扫描剩下所有端口）。但是这种方法有一个弊端，例如 1000 个端口，300 个线程，前 299 个线程每个线程只用扫描 3 个端口，最后一个线程却需要扫描剩下的 103 个端口，这显然不符合多线程的初衷，于是我改进了分配端口的方法。

方案二：前面与方案一相同，但每给一个线程分配好端口数后，就计算一次剩下的端口/剩下的线程，如果这个值大于 step，就表示之后的每一个线程需要多分配几个端口（准确地说是 1 个），则将其赋给 step，测试程序后发现运行速度明显提高了，不存在一个线程扫描超多端口的现象。

(5) 关于扫描输出：

本着用户友好原则，在扫描过程中打印所有的扫描结果，但因为多线程的原因，扫描出的顺序是乱的，所以在扫描结束后单独打印出开启的端口号，并且打印此次扫描花费的时间、扫描的总端口数以及开启的端口数。

(6) 用户可以在扫描正在进行时按结束扫描的按钮来中断扫描，点击按钮后会跳出对话框确认以防止用户误点，当程序收到结束扫描的信号时会中断所有线程，这一过程是安全的。

## 2. 界面编写

使用Python的Tkinter GUI来编写界面，界面主要分为三个部分，主界面为Sniffer嗅探器，结果显示界面为Result，错误提示界面为Error。

1. 主界面Sniffer：

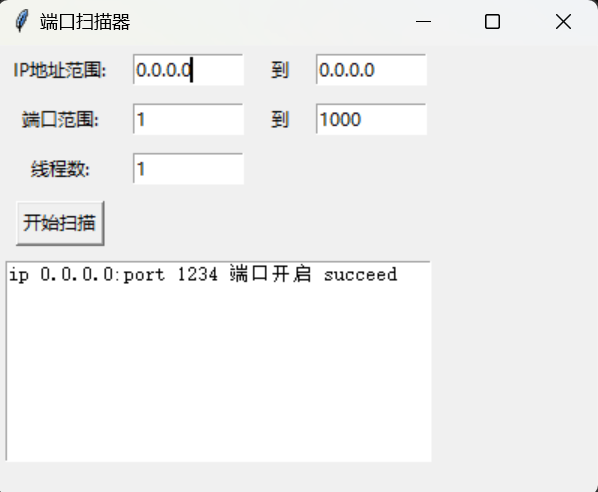
IP范围的两个端点IP共两个输入框，端口范围两个输入框，线程数一个输入框。

点击开始扫描，检查输入的正确规范性后程序便可以开始运行。



1. 结果显示界面Result：

显示扫描结果，其中包括了探测成功的IP地址与端口号（所耗时间省略）。



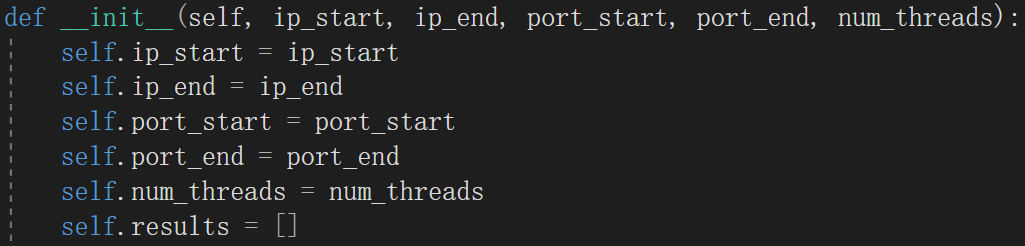
1. 错误提示界面Error：

当输入正确性出错时（例如输入框内输入了非数字，或者输入范围不是从大到小。又或者输入为空）。可以根据错误信息提示（根据错误类型改变提示信息）。

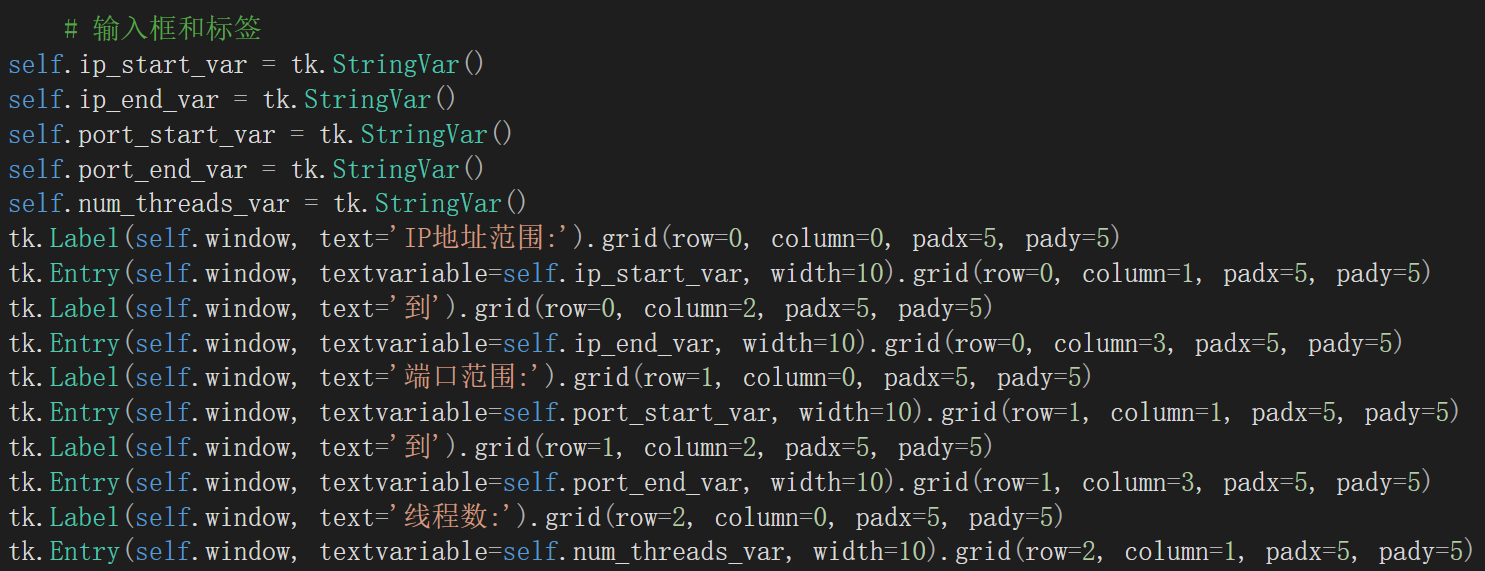


## 3. 控件逻辑编写

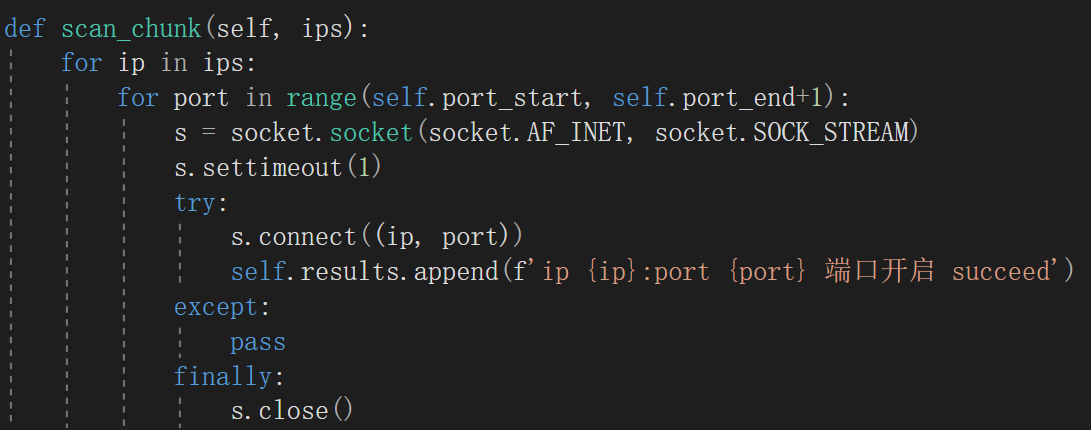
结果和错误提示控件逻辑较为简单，均为点击按钮“确定”来关闭当前界面的。



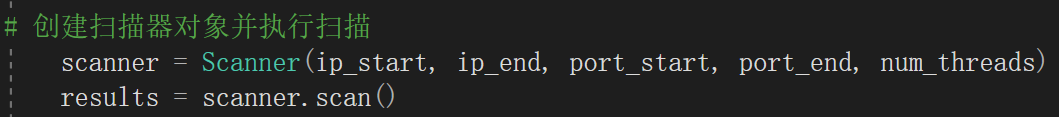
下面介绍Sniffer界面控件逻辑：



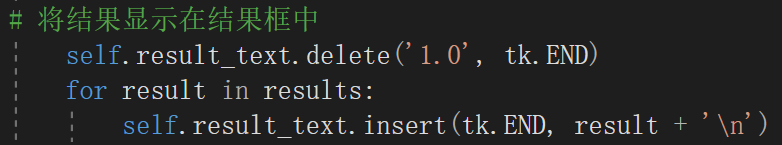
首先得到各个输入框内的数字，然后再检查各个字段的正确性（图片只包含部分检查）。



若发现错误则跳转到提示界面并且停止继续运行。



否则创建线程池并且创建子线程来扫描端口。



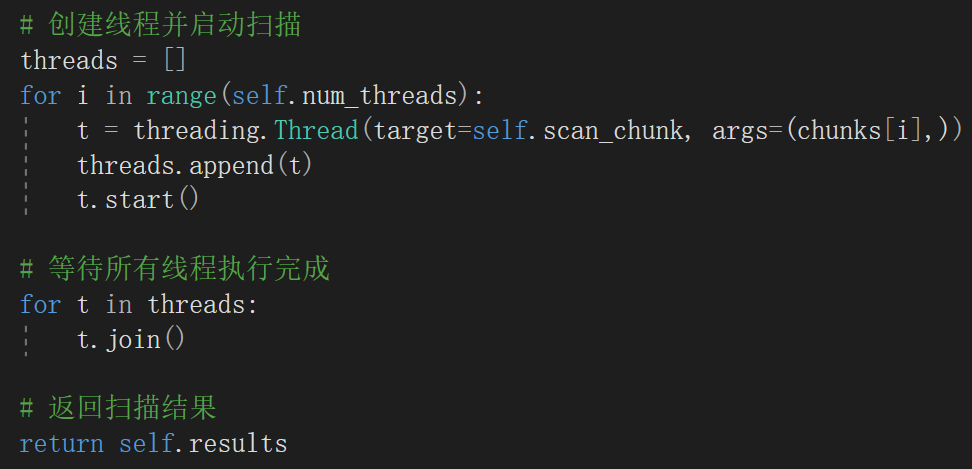
最后将扫描结果传输到结果界面输出。

## 4. 具体功能编写

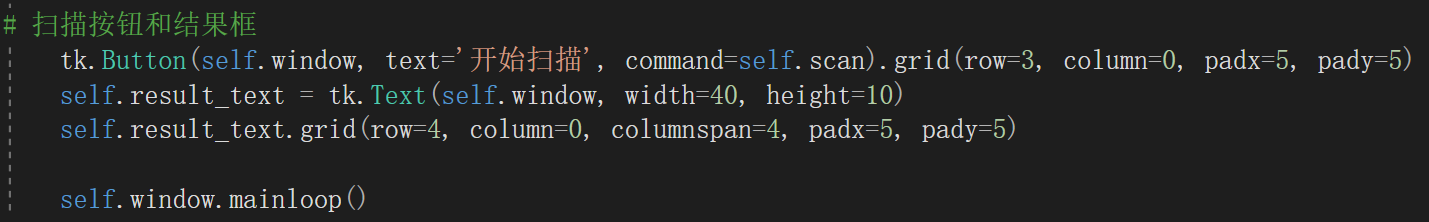
1. 扫描主线程

扫描主线程：其中编写了界面代码以及控件逻辑，除此之外还有创建子线程进行扫描的主线程。

主要逻辑分为四阶段：得到输入框输入、判断输入正确性、创建子线程扫描端口+等待子线程结束、输出扫描结果。



1. 扫描按钮与结果框



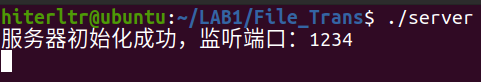
注意使用tk库中的Button()函数进行IP地址、端口号工具类的扫描成功记录，内部成员变量host记录IP地址，port记录端口号。除此之外，可以进一步扩展：工具类内含有判断输入字符串是否为非负数字的辅助函数使用正则表达式进行匹配。

# 四、实验结果

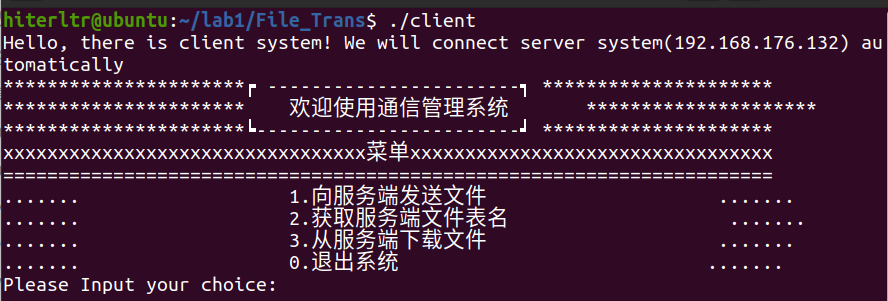
1. **Linux 环境下的 C/S 程序**

**传输前：**

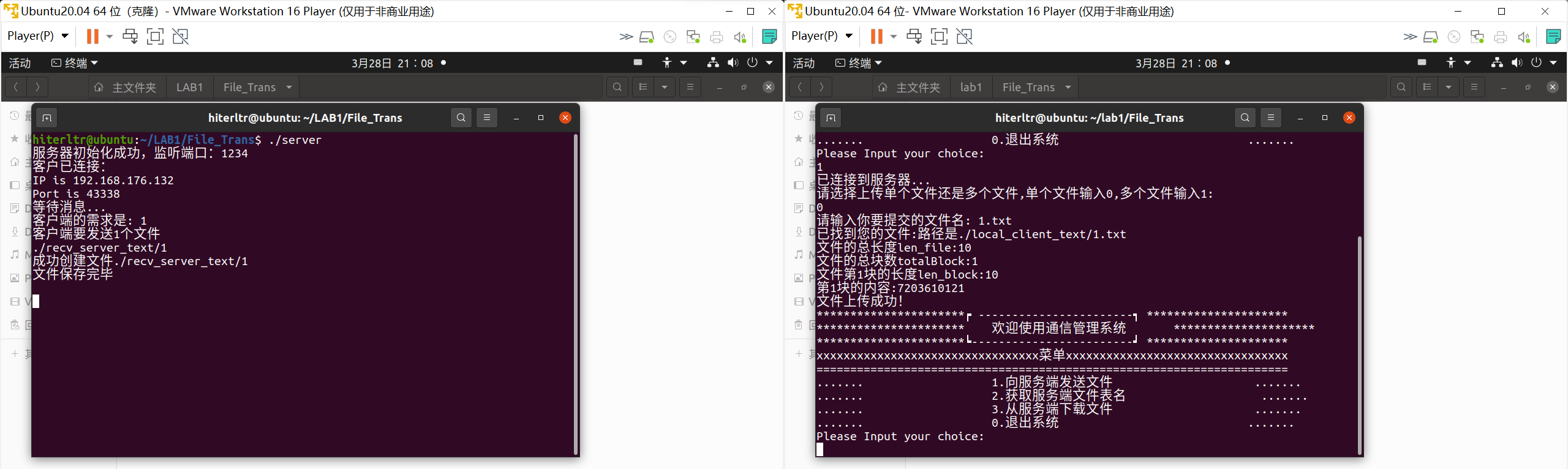
服务器端：



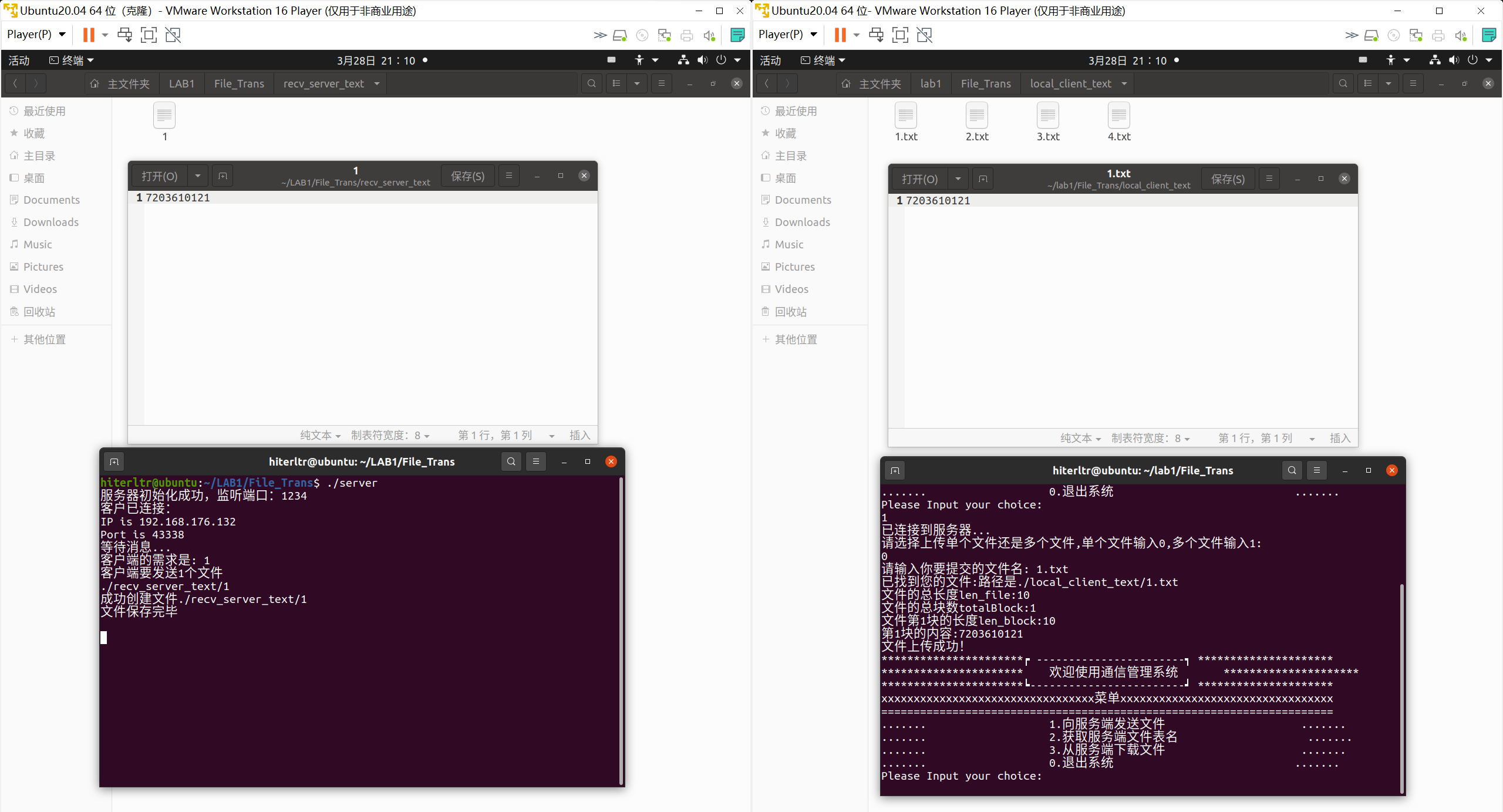
客户端：



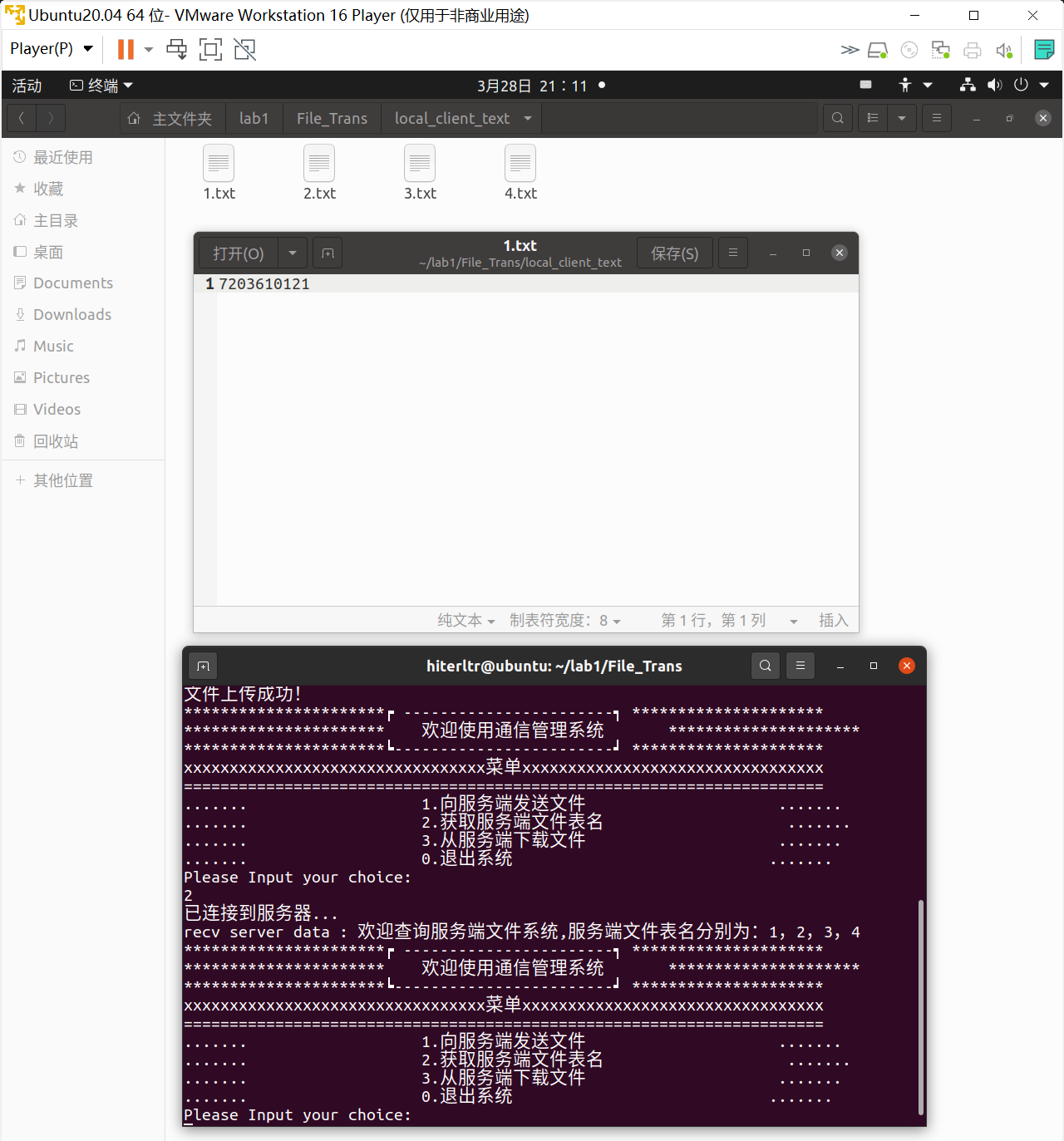
选择向服务器端发送文件：



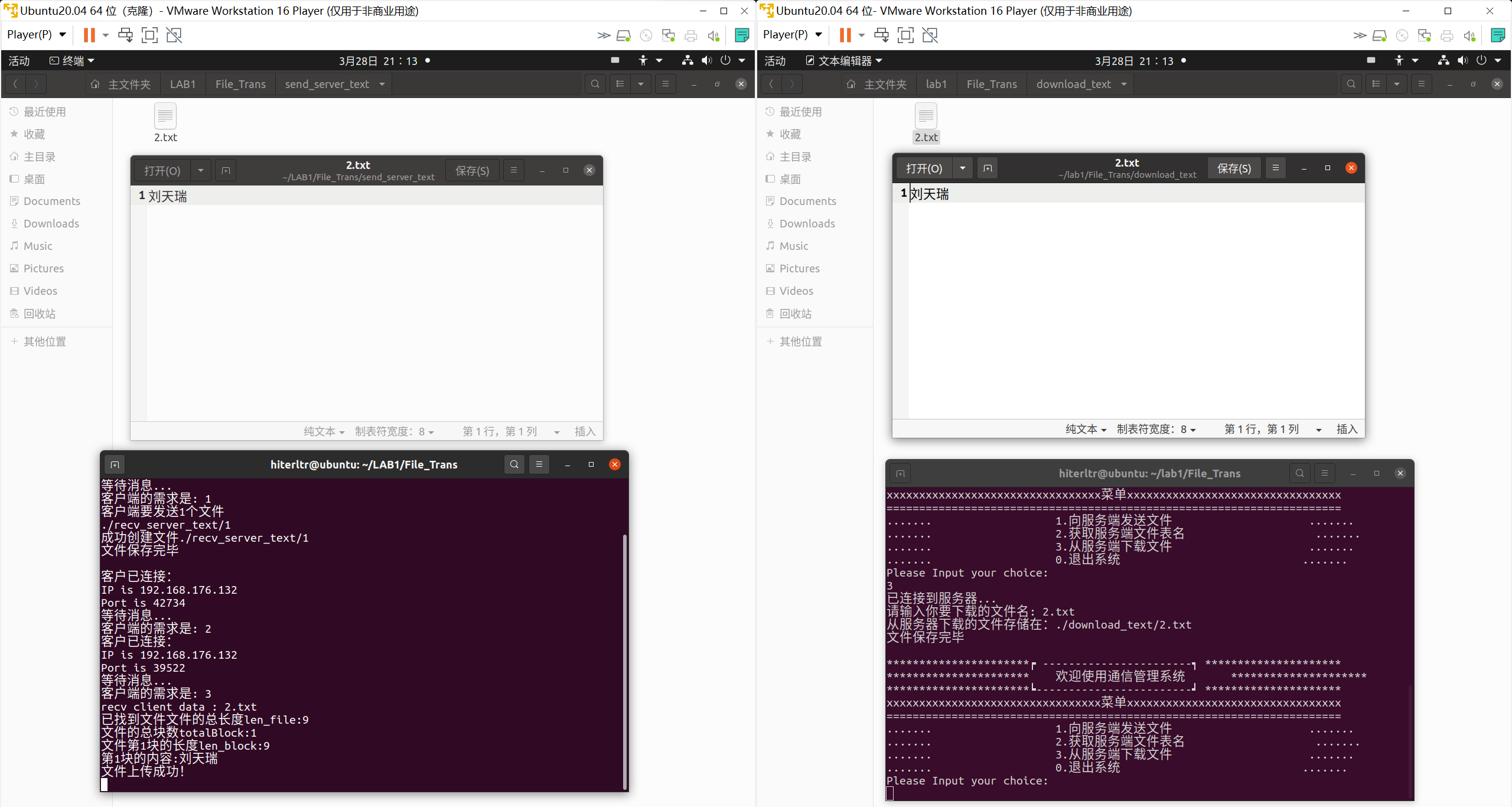
**传输后：**



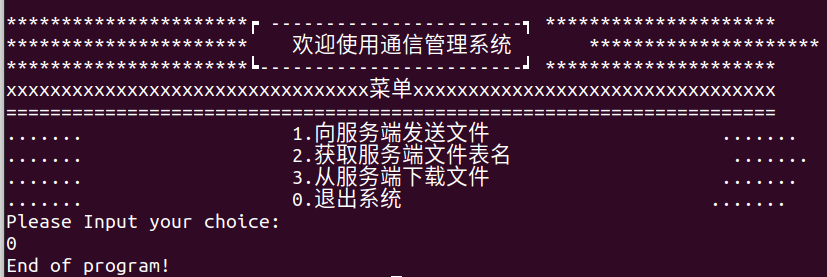
选择获取服务器端文件列表名称：



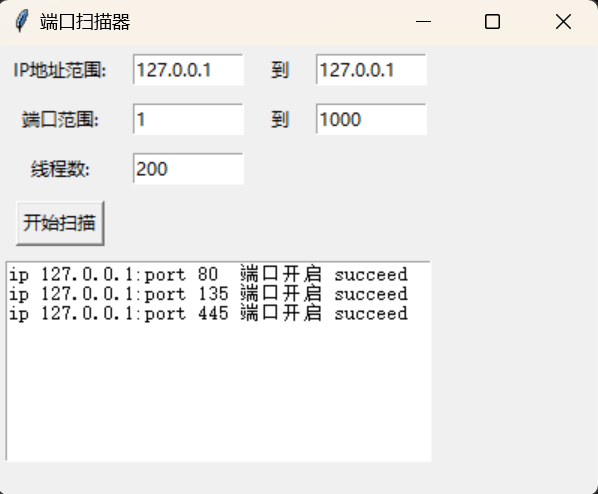
选择从服务器端下载文件：



退出系统：



1. **Windows 环境下的扫描器程序**



如上图所示为这台计算机的部分端口号的扫描结果。