

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 120L020322

学 生 刘祚甫

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 16.04 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。

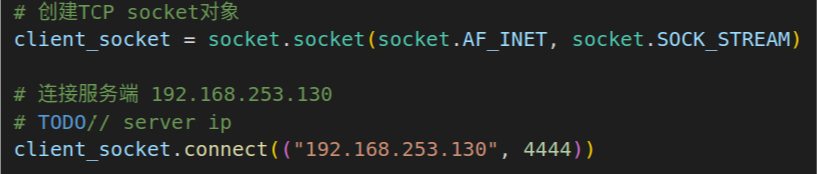
## 2. 环境配置

配置虚拟机的网卡即可：



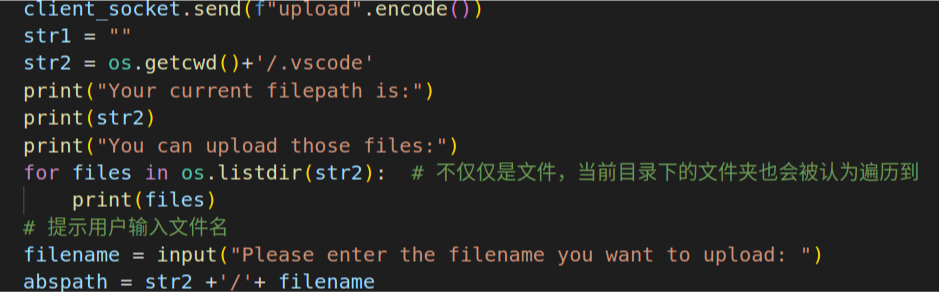
## 客户端编写

（1）创建TCP Socket并连接服务器，服务器所在的虚拟机的ip地址为192.168.252.130，端口号是4444

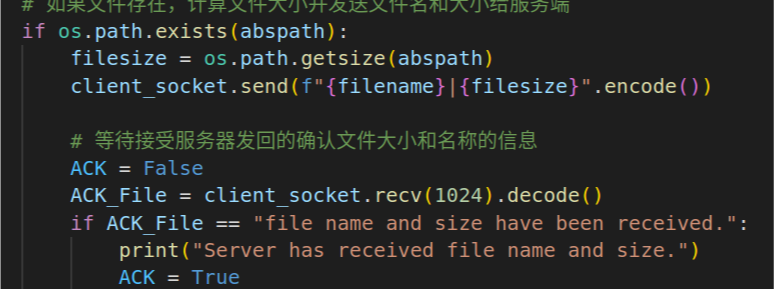


1. 客户端可以完成上传、下载和终止传输三种操作，下面是第一种：上传操作

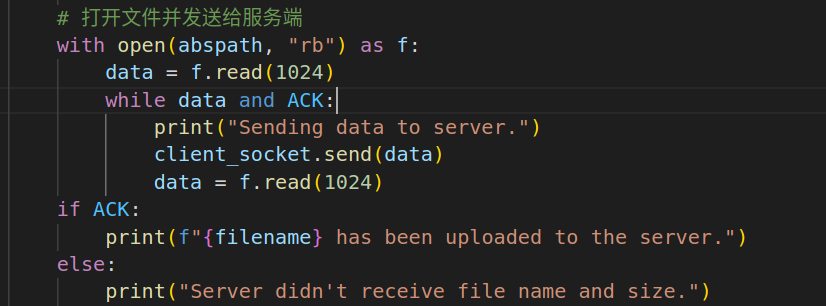
先给出所有可以上传的文件，用户输入想要上传的文件



文件如果存在，就向服务器发送一个传输请求，等待ack

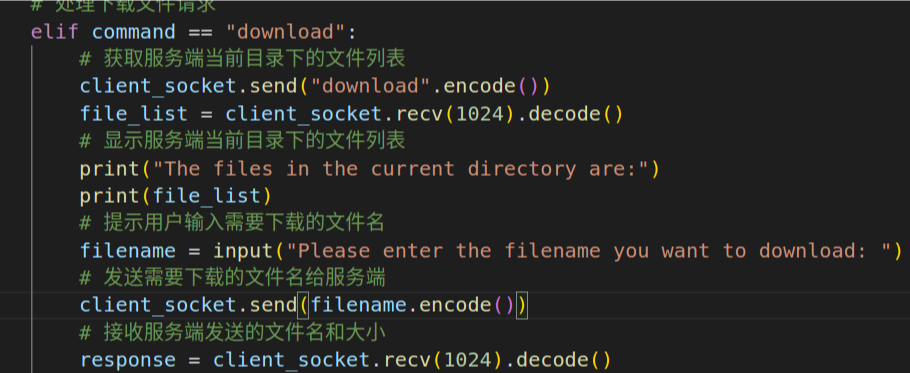


收到ack确认“ok”后，发送文件

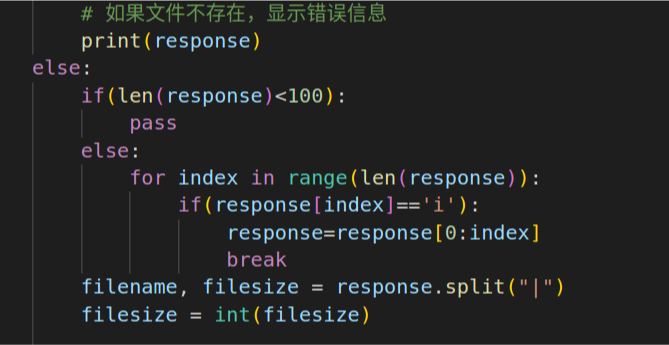


1. 如果命令为下载（download）

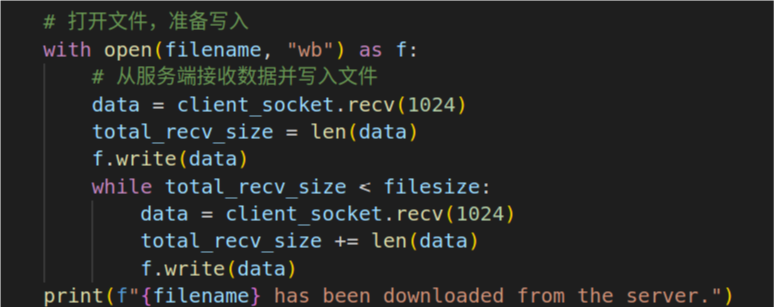
先向服务器发送指令，获取可以下载的文件列表



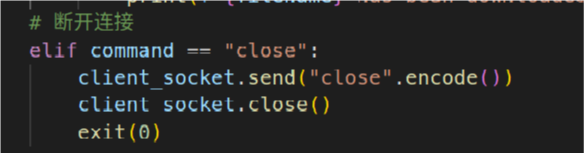
向服务器端传送要下载的文件名后，若存在文件，服务器先传送文件名字和文件的大小。这里我发现了一个问题：传送时总是有一定概率会带上文件的内容，这会导致filesize变量报错。所以我先检查一下长度，并做出截取操作



最后打开文件进行写入



1. 如果是中断连接操作，则直接发送close命令

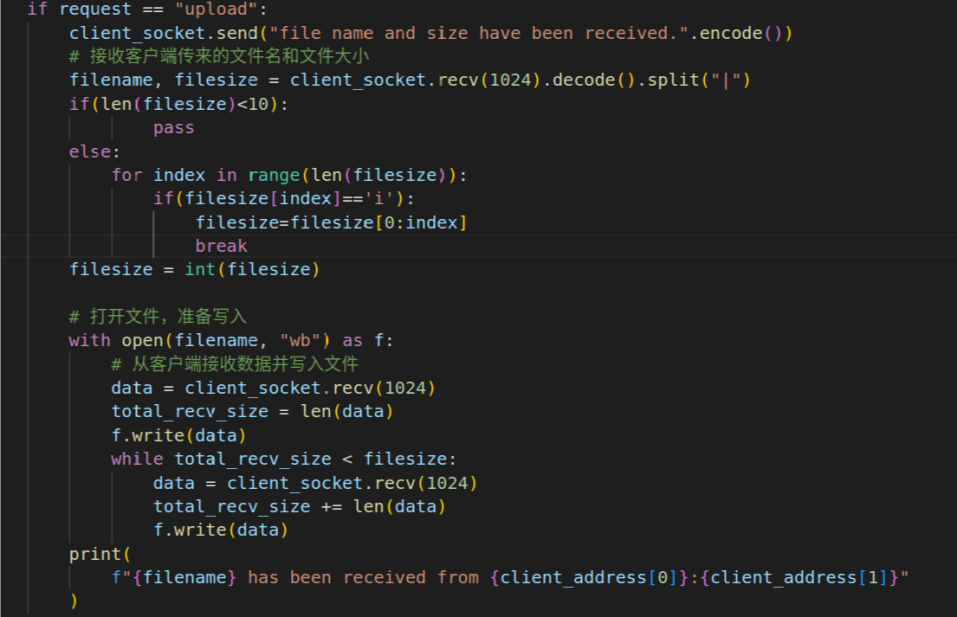


## 服务端编写

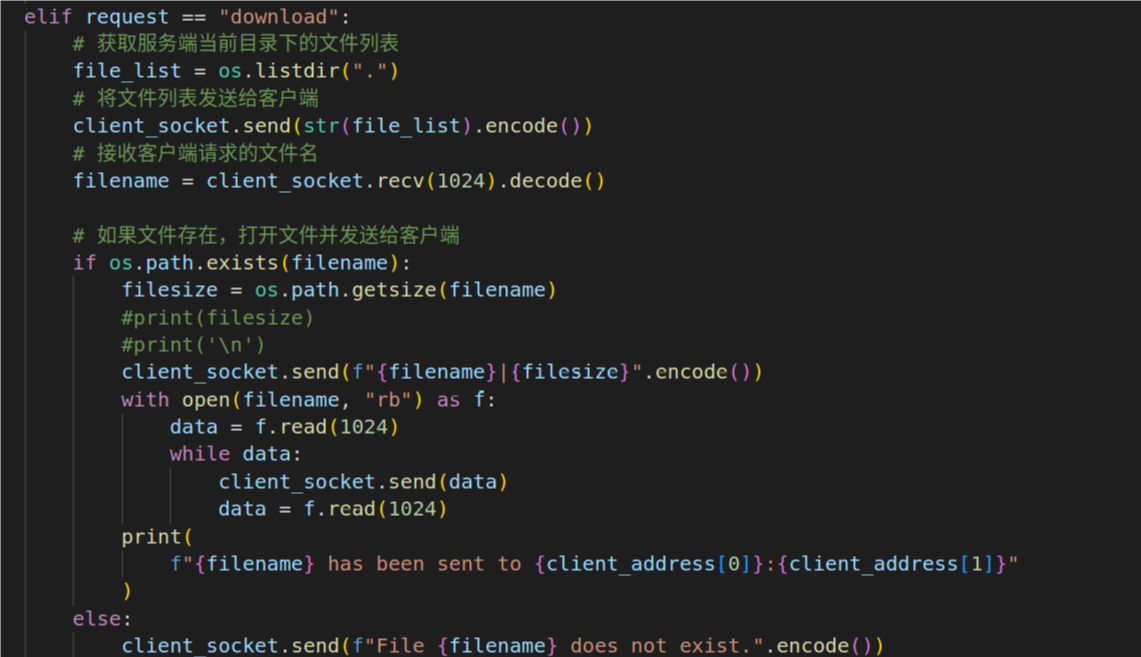
先创建TCP套接字，绑定IP地址和端口号，用8888端口进行监听客户端的连接请求



1. 若为上传请求，先接收文件名和文件大小。此处也会出现与客户端相同的有一定概率同时传输过来文件内容的问题，所以也要对字符串进行拆解和检查。随后对文件进行写入



（2）若为下载操作，先传输文件列表，然后检查用户要下载的文件名称，如果在文件列表中，就传输给用户



（3）若为close操作，则直接中断连接

**（二）Windows 环境下的扫描器程序**

## 实验基本信息：

实验环境：Windows10 x64

QtCreator 4.8.1 编程语言：C++

## 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 C++的 QtCreator 来编写程序，QtCreator 的界面编写非常方便（拖拖拖），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。

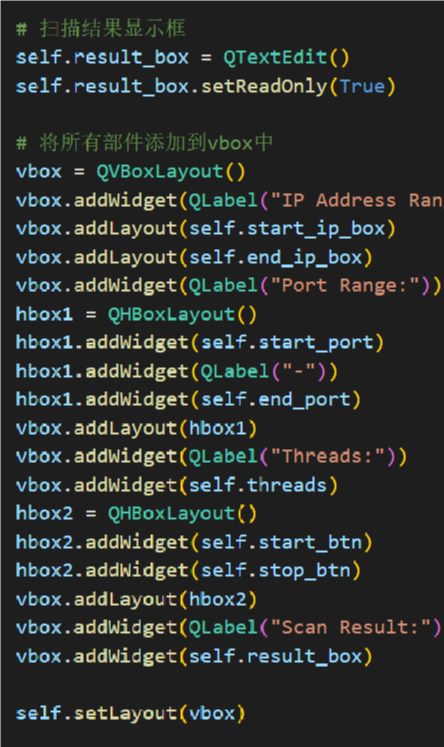
方案二：前面与方案一相同，但每给一个线程分配好端口数后，就计算一次剩下的端口/剩下的线程，如果这个值大于 step，就表示之后的每一个线程需要多分配几个端口（准确地说是 1 个），则将其赋给 step，测试程序后发现运行速度明显提高了，不存在一个线程扫描超多端口的现象。 (5) 关于扫描输出：

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



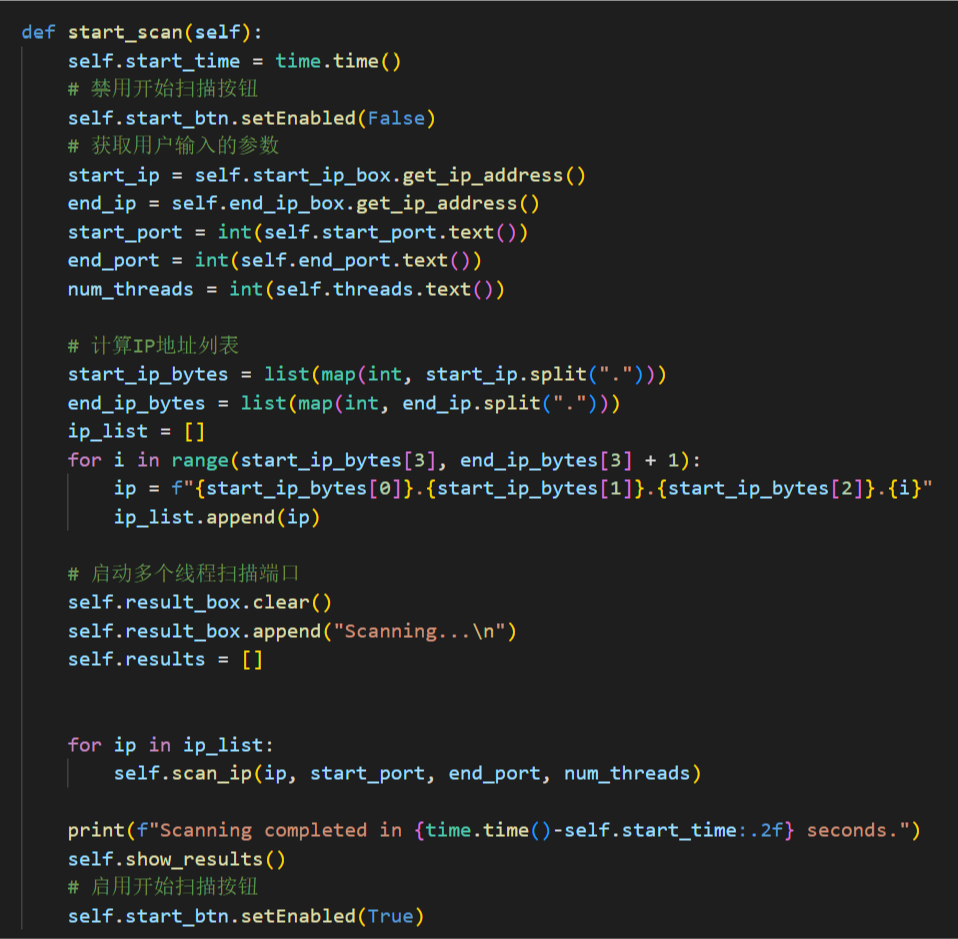


## 控件逻辑编写 、



## 4. 具体功能编写

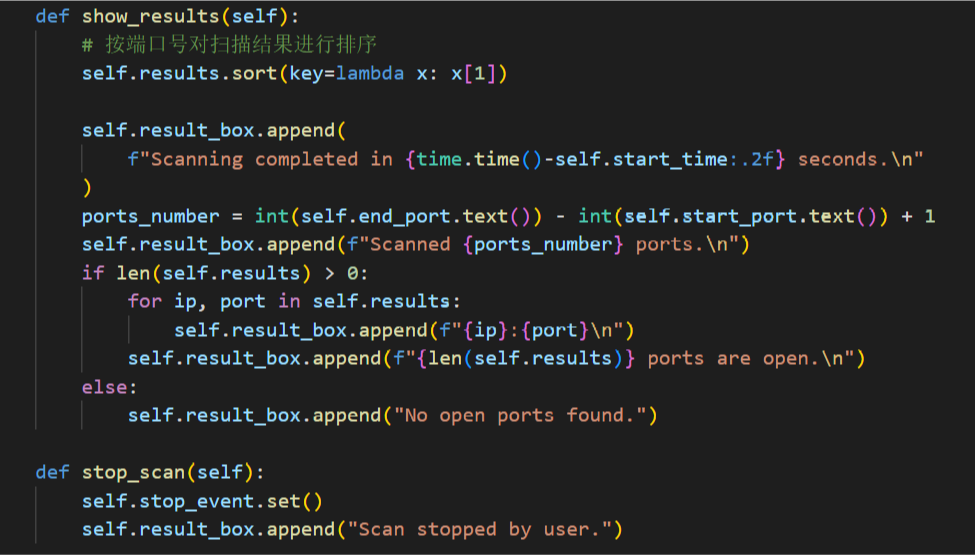
1. 扫描主方法 void MainWindow::**scan**(int numOfThread)



1. 扫描线程方法 void\* MainWindow::**scan\_some\_port**(void \*arg)

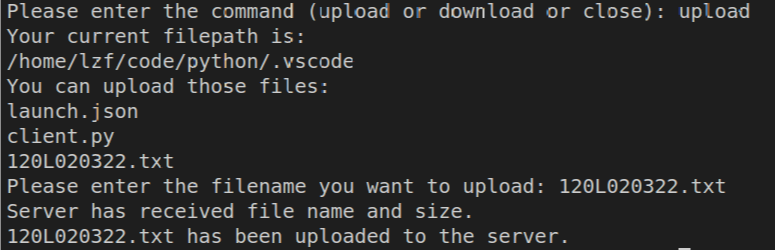


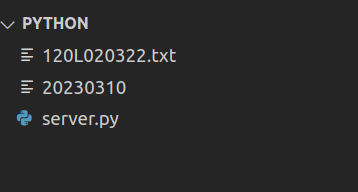
1. 向 TextBrowser 打印扫描结果 void MainWindow::**upload**()



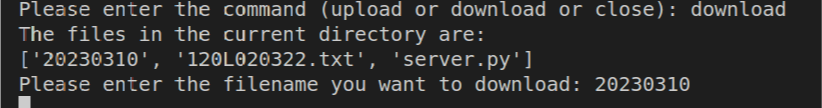
# 四、实验结果

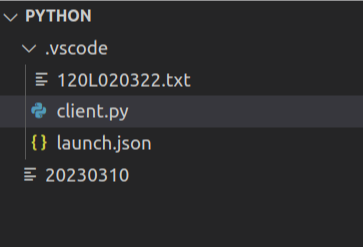
1. **Linux 环境下的 C/S 程序**
2. **upload功能**



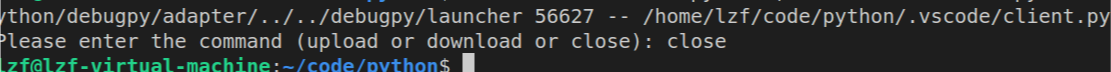


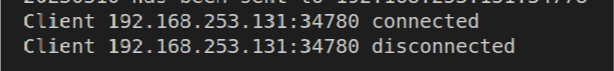
1. **download功能**



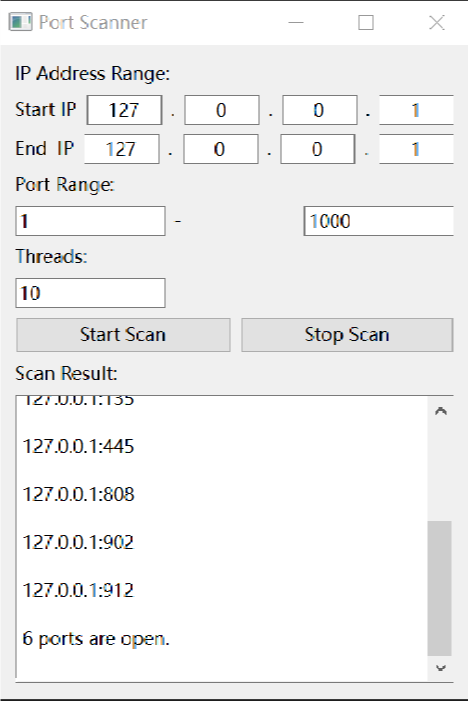
****

1. **close功能**

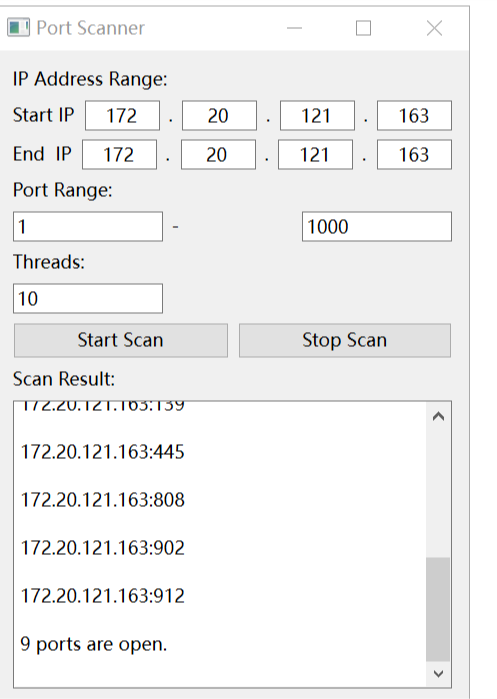




1. **Windows 环境下的扫描器程序**
2. 扫描本机



1. 扫描hit-WLAN



1. 扫描baidu.com

