

# 实例 1: 扫描开放端口

用户可根据"IP 地址:端口号"访问网络中计算机的进程,为避免不法分子利用闲散端口侵入计算机,运维人员通常会采取关闭冗余端口的措施进行预防。但计算机中拥有的端口数量较多,仅靠人力排查的方式显然是不可取的。因此,考虑通过编程解决这一问题。

本实例要求编写程序、扫描计算机端口、输出开放的端口号。

### 实例目标

● 掌握 socket 内置方法的使用

## 实例描述

扫描计算机端口的基本思想是:在循环结构中通过子进程/线程逐一与各个端口建立连接,若连接建立成功,说明当前端口为开放端口,计数加一;否则为未开放端口,继续遍历。

## 代码实现

本实例的具体实现代码如下所示:

```
from socket import *
import threading
lock = threading.Lock()
openNum = 0
threads = []
def portScanner(host,port):
   global openNum
   try:
      s = socket(AF_INET, SOCK_STREAM)
      s.connect((host,port))
      lock.acquire()
      openNum+=1
      print('[+] %d open' % port)
      lock.release()
      s.close()
   except:
      pass
def main():
```

网址: yx.boxuegu.com 教学交流QQ/微信号: 2011168841



```
setdefaulttimeout(1)

for p in range(1,65534):

    t = threading.Thread(target=portScanner,args=('127.0.0.1',p))

    threads.append(t)
    t.start()

for t in threads:
    t.join()

print('[*] 扫描完成! ')

print('[*] 一共有%d个开放端口 '% (openNum))

if __name__ == '__main__':

main()
```

以上代码中的全局变量 openNum 用于记录计算机中开放端口的数量;函数 portScanner() 为子线程功能函数,该函数接收主机 ip 与端口号,建立 TCP 连接,以测试指定端口状态,若子线程成功建立连接,说明相应端口号开放,计数变量 openNum 值加 1,并打印当前端口号;主函数 main()用于遍历计算机所有端口,以及创建子线程,遍历完成后打印全局变量 openNum 的值,即可得知开放端口数量。

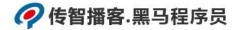
另外考虑到全局变量 openNum 为各个线程的共享资源,程序中创建了互斥锁 lock,以避免多个线程同时修改 openNum 的值,导致记数出错。

## 代码测试

运行程序,结果如下所示。

```
[+] 135 open
[+] 443 open
[+] 445 open
[+] 902 open
[+] 912 open
[+] 1025 open
[+] 1026 open
[+] 1027 open
[+] 1033 open
[+] 1034 open
[+] 1688 open
[+] 3306 open
[+] 4300 open
[+] 4301 open
[+] 4302 open
[+] 4304 open
```

网址: yx.boxuegu.com 教学交流QQ/微信号: 2011168841



- [+] 4303 open
- [+] 4305 open
- [+] 5357 open
- [+] 8082 open
- [+] 8307 open
- [+] 33060 open
- [\*] 扫描完成!
- [\*] 一共有 22 个开放端口