

## 计算机网络课程设计报告

设计题目：《端口扫描工具的设计与实现》

姓名：

学号：

专业班级：

完成日期： 2024年12月20日

1. 课程设计任务
2. 实现一个功能比较简单的、具有图形界面的端口扫描工具共,主线程响应用户界面操

作,工作线程完成端口扫描等工作;

2. 能够扫描指定IP地址的主机/服务器开放了哪些端口;

3. 能够扫描指定IP地址范围内的哪些主机/服务器开放了物了特寺定端口,如常见的TCP端口FTP(21)、SMTP(25), UDP端口 端口 DNS (53)、SNMP(161);

4.扫描动作要具有一定的隐蔽性和效率。

1. 课程设计目的和要求。

2.1 设计目的

* 掌握Python网络编程技术
* 理解多线程编程原理
* 熟悉GUI程序设计方法
* 提高系统架构设计能力

2.2 具体要求

1. 功能要求：

* 支持IP范围扫描
* 支持多端口扫描
* 支持TCP/UDP协议
* 具备进度显示功能
* 可实时显示结果

1. 性能要求：

* 具有较高的扫描效率
* 保证程序稳定性
* 具备良好的用户体验

1. 开发环境。

3.1 硬件环境

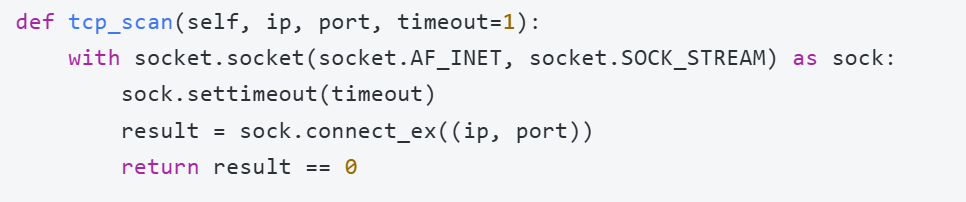
* CPU: AMD 6800
* 内存: 16G

3.2 软件环境

* 操作系统: Windows 11
* Python版本: 3.13.1
* 主要库依赖:
  + tkinter (GUI界面)
  + socket (网络通信)
  + threading (多线程)
  + queue (线程通信)
  + ipaddress (IP地址处理)

1. 相关原理及算法
   1. **端口扫描原理**

1. TCP扫描:



TCP扫描的工作原理：

1. 通过 socket 建立 TCP 连接尝试
2. 利用 connect\_ex() 方法判断连接结果
3. 返回值为 0 表示端口开放
4. 设置超时机制避免长时间等待

1. UDP扫描:

图形用户界面, 文本, 应用程序

描述已自动生成

UDP扫描的特点：

1. 无连接特性导致扫描结果不如 TCP 准确
2. 需要处理超时情况
3. 可能受到防火墙干扰
   1. **多线程设计原理**

使用线程池管理多个扫描线程，通过Queue实现线程间通信：

文本

描述已自动生成

* 多线程设计考虑因素：

1. 线程数量控制：默认设置20个线程
2. 任务分配策略：IP和端口的组合任务
3. 资源竞争处理：使用队列实现线程通信
4. 任务调度优化：随机化扫描顺序

* 通信实现要点：

1. 使用 Queue 确保线程安全
2. GUI 定时轮询结果队列
3. 使用特殊消息标记进度更新
4. 实现扫描状态同步
5. 系统结构和主要的算法设计思路描述

**5.1 系统架构**

采用MVC架构模式：

文本

描述已自动生成

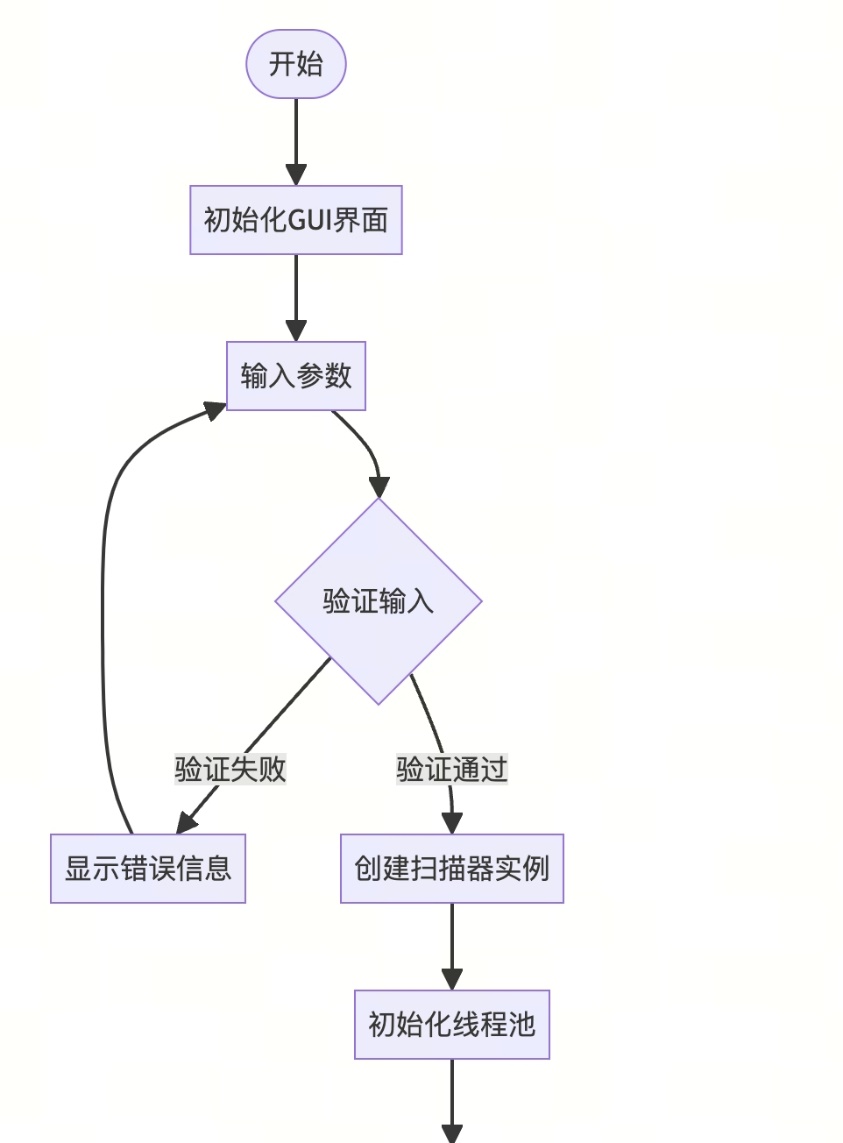
* 1. **主要模块**

1. 扫描模块(scanner.py)
2. 界面模块(gui.py)
3. 主程序模块(main.py)

**5.3 核心算法流程**

文本

描述已自动生成



图示

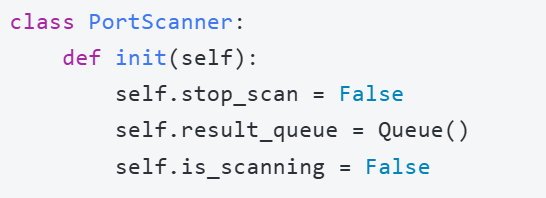
描述已自动生成

图示

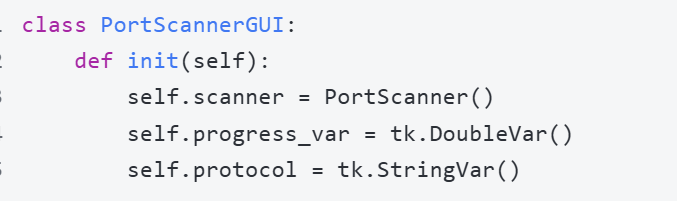
描述已自动生成

1. 程序实现-主要数据结构

**6.1 端口扫描器类：**



**6.2 GUI类：**



1. 程序实现-细节描述

7.1 各个模块说明

**1. main.py**

程序的入口文件，它主要做了以下事情：

* 从gui模块中导入PortScannerGUI类。
* 在if \_\_name\_\_ == "\_\_main\_\_"条件下，创建PortScannerGUI类的实例，并调用run方法来启动图形用户界面（GUI）。

**2. scanner.py**

定义了PortScanner类，负责实现端口扫描的核心逻辑：

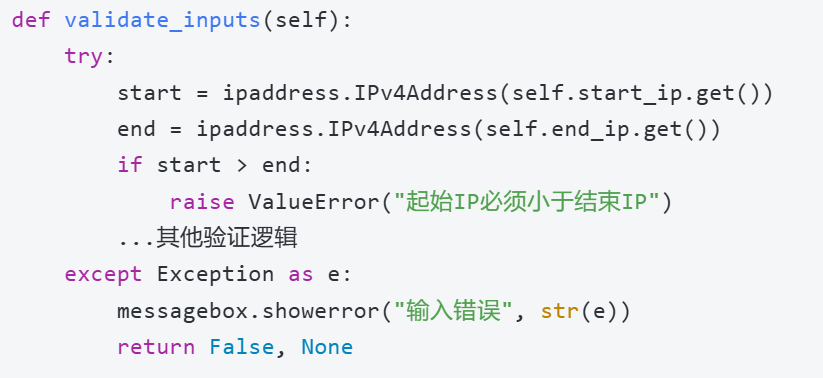
* **初始化方法（\_\_init\_\_）**
  + 初始化了一些实例变量，如stop\_scan（用于控制扫描是否停止）、result\_queue（用于存储扫描结果的队列）等。
  + 调用setup\_logging方法来配置日志记录。
* **setup\_logging方法**
  + 使用logging.basicConfig配置日志记录的基本设置，包括日志级别为INFO和日志格式。
* **tcp\_scan方法**
  + 用于扫描 TCP 端口。它创建一个 TCP 套接字，尝试连接指定的 IP 和端口。
  + 如果连接成功（connect\_ex返回 0），则获取该端口对应的服务名称（使用getservbyport），并将结果放入result\_queue中，同时记录日志。
* **udp\_scan方法**
  + 用于扫描 UDP 端口。它创建一个 UDP 套接字，向指定 IP 和端口发送空数据，并尝试接收响应。
  + 如果在超时时间内接收到响应，则获取该端口对应的服务名称，并将结果放入result\_queue中，同时记录日志。
* **get\_service\_name方法**
  + 尝试获取给定端口对应的服务名称，如果获取失败则返回unknown。
* **scan\_host方法**
  + 根据指定的协议（TCP 或 UDP）调用相应的扫描方法（tcp\_scan或udp\_scan）。
  + 在扫描前会随机等待一小段时间（0.01 到 0.1 秒），以增加扫描的隐蔽性。
  + 将当前扫描信息放入result\_queue中，并返回扫描结果。
* **scan\_range方法**
  + 这是主要的扫描方法，用于扫描指定 IP 范围内的所有主机的指定端口。
  + 首先对 IP 地址范围进行验证和处理，将 IP 地址转换为IPv4Address对象，并生成 IP 地址列表。
  + 使用ThreadPoolExecutor创建线程池，根据max\_threads参数控制并发线程数量。
  + 将扫描任务分批提交给线程池，并处理扫描结果和进度更新。
  + 如果在扫描过程中stop\_scan被设置为True，则停止扫描。
* **stop\_scanning方法**
  + 简单地将stop\_scan变量设置为True，用于停止正在进行的扫描。

**3. gui.py**

文件定义了PortScannerGUI类，用于创建图形用户界面：

* **初始化方法（\_\_init\_\_）**
  + 创建主窗口root，设置窗口标题和大小。
  + 创建PortScanner类的实例，用于执行扫描操作。
  + 调用setup\_gui方法来设置图形界面的各个组件。
  + 调用update\_results方法来开始更新扫描结果。
* **setup\_gui方法**
  + 创建各种 GUI 组件，包括：
    - IP 输入区域：包括起始 IP 和结束 IP 的输入框。
    - 端口输入区域：输入要扫描的端口范围。
    - 协议选择：通过单选按钮选择扫描协议（TCP 或 UDP）。
    - 进度条区域：包括进度条和进度百分比标签，以及当前扫描状态标签。
    - 按钮区域：包括开始扫描和停止扫描按钮。
    - 结果显示区域：使用scrolledtext组件来显示扫描结果。
* **validate\_inputs方法**
  + 验证用户输入的起始 IP、结束 IP 和端口范围的合法性。
  + 如果输入合法，则返回True和处理后的端口列表；如果不合法，则弹出错误消息框并返回False和None。
* **update\_progress方法**
  + 根据传入的百分比值更新进度条和进度百分比标签。
* **reset\_progress方法**
  + 将进度条重置为 0，并更新进度百分比标签为0%。
* **start\_scan方法**
  + 验证用户输入，如果输入合法，则禁用开始扫描按钮，启用停止扫描按钮，清空结果显示区域，重置进度条。
  + 创建一个新线程来执行scanner.scan\_range方法，开始扫描操作。
* **stop\_scan方法**
  + 调用scanner.stop\_scanning方法来停止扫描，更新开始扫描和停止扫描按钮的状态。
* **update\_results方法**
  + 不断从scanner.result\_queue中获取扫描结果。
  + 如果结果是进度信息，则更新进度条；如果是状态信息，则更新状态标签；如果是扫描完成信息，则更新进度条为 100%，更新按钮状态和状态标签，并在结果显示区域显示扫描完成信息；否则，将扫描结果插入到结果显示区域。
  + 最后，使用root.after(100, self.update\_results)在 100 毫秒后再次调用自身，以实现实时更新扫描结果。
* **run方法**
  + 调用root.mainloop()来启动图形用户界面的主事件循环。

**7.2 输入验证实现**



7.3 进度更新实现

文本

描述已自动生成

1. 程序运行的主要界面和结果截图

**8.1 主界面**

图形用户界面, 应用程序

描述已自动生成

主界面包含：

* IP地址输入区域
* 端口范围输入
* 协议选择按钮
* 进度条显示
* 扫描结果显示区域

**8.2 扫描结果示例**

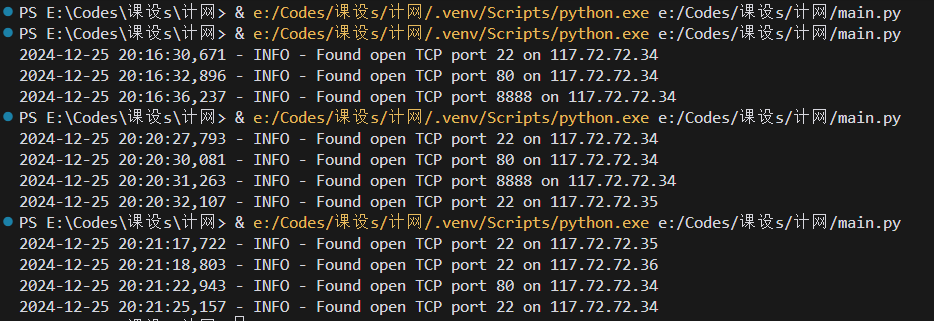
图形用户界面, 文本, 应用程序

描述已自动生成

图形用户界面, 文本

描述已自动生成

**扫描我的个人服务器结果**

 **终端日志打印效果**

(9) 附录