

**⽹际层也称为：⽹络层 ⽹络接⼝层也称为：链路层**

**互联⽹协议包含了上百种协议标准，但是最重要的两个协议是*TCP和IP协议*，所以，⼤家把互联⽹的协议简称TCP/IP协议 22端口ssh**

**协议是完成进程之间通讯的规范 TCP/IP是协议族**

**TCP/IP协议中的端⼝**指的是什么呢？端⼝就好⼀个房⼦的⻔，是出⼊这间房⼦的必经之路。

知道**pid**可以直接发送（只能在一台电脑区分，多台不可以） **端口是用来唯一确定一个进程的方式，标记用来确定地址**

**⽤“netstat －an”查看端⼝状态 linux系统中，端⼝可以有65536（2的16次⽅）个**

**端⼝是通过端⼝号来标记的，端⼝号只有整数，范围是从0到65535**

**端口分为知名端⼝和动态端⼝** ⼀般情况下，如果⼀个**程序需要使⽤知名端⼝的需要有root权限**

知名端⼝是众所周知的端⼝号，范围从0到1023 **动态端口**是1024到65535

**动态端口它⼀般不固定分配某种服务，⽽是动态分配。动态分配是指当⼀个系统进程或应⽤程序进程需要⽹络通信时，它向主机申请⼀个端⼝，主机从可⽤的端⼝号中分配⼀个供它使⽤。当这个进程关闭时，同时也就释放了所占⽤的端⼝号。**

**ip**地址：⽤来在⽹络中标记⼀台电脑，⽐如**192.168.1.1**；在本

地局域⽹上是唯⼀的。

**80端⼝分配给HTTP服务 （提供网站）**

**21端⼝分配给FTP服务（文件下的）**

**IP地址与⽹络服务的关系是⼀对多的关系。实际上是通过“IP地址+端⼝号”来区分不同的服务的。IP地址逻辑上唯一的标记一台电脑**

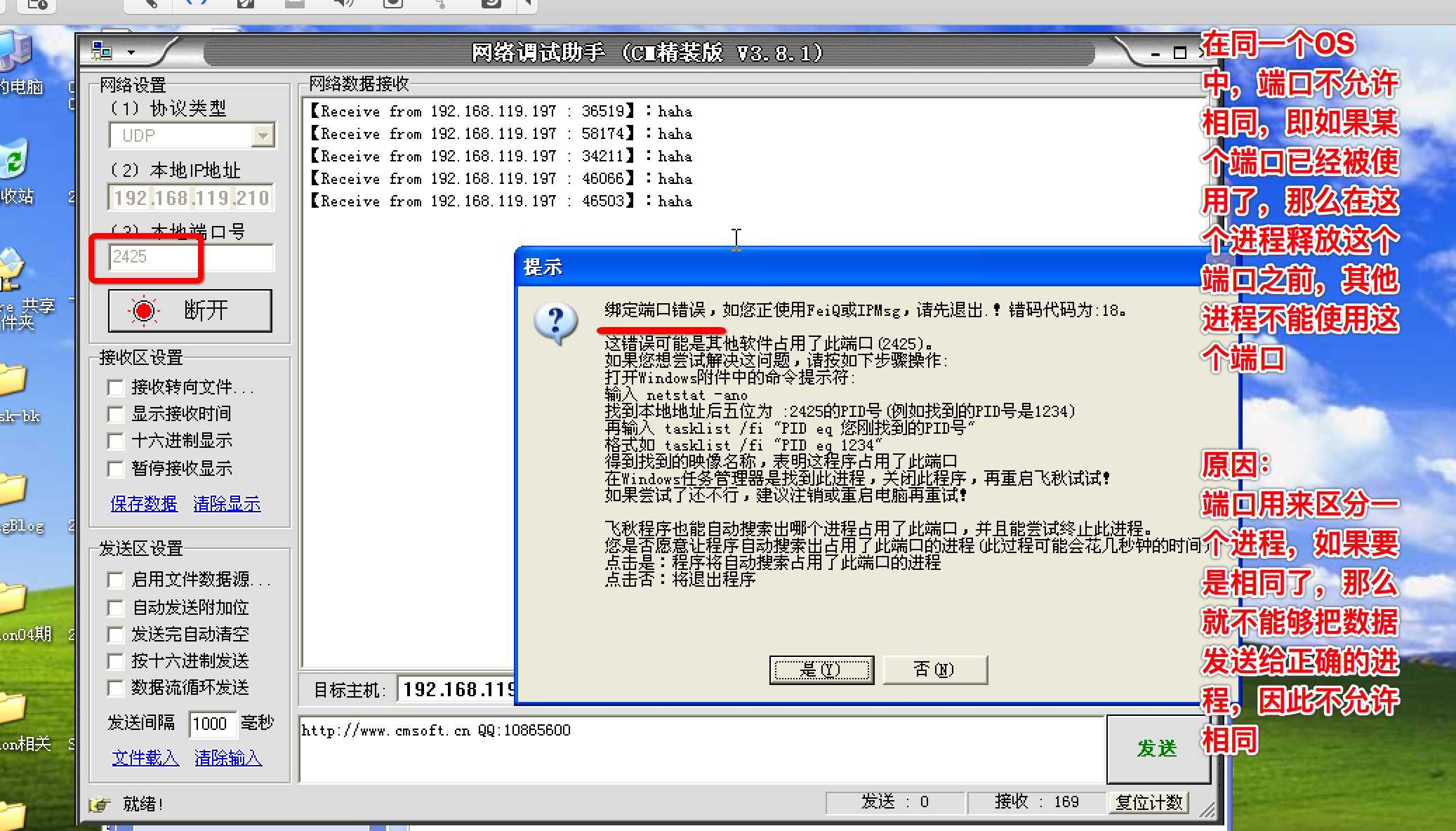
**ip地址的分类 ipv4有2的32次方 4G 4个字节 二的八次方 0-255 最小是当前网络号，最大是广播 2的10次方是k，20次方是M，30次方是G**

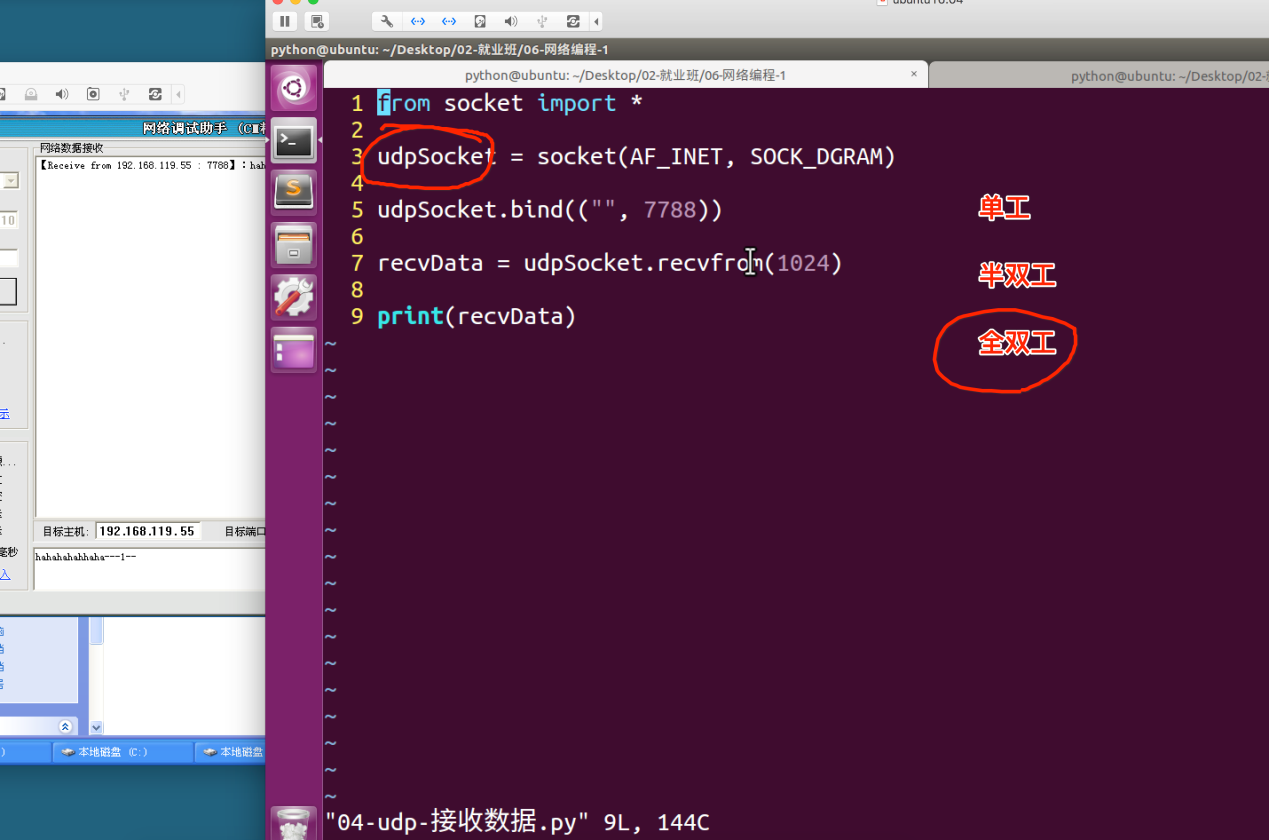
**ipv6 十六进 8个字节 ipv4 4个字节**  **每⼀个IP地址包括两部分：⽹络地址和主机地址**



**D类地址⽤于多点⼴播（视频会议）**IP地址127．0．0．1~127．255．255．255⽤于**回路测试**

**Socket是一种通过网络完成进程之间通信的方式**





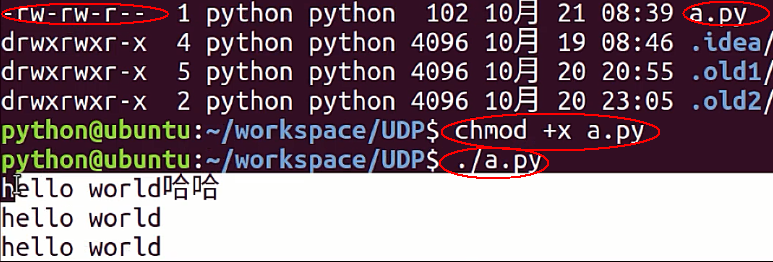
Ctrl shift **t** 在一个终端上开多个终端

交互式 vim a.py 编辑器把文件写好 然后在交换状态下，用Python运行 格式python a.py

Which python 查看python解释器所在位置







可执行是自己可以执行 **./当前路径下的a.py程序**

Python终端中 Ctrl  **d** 退出 ctrl **z** 切换到后台 fg查看后台 jobs查看所有后台

1. python程序的几种运行方式

交互式 Python ipython（自动提示）

场景 – 验证语法，简单功能

执行 vim/sublime + python解释器

Chmod +x \*.py  **./**a.py

IDE

1. 网络概论

形式

目的 实现信息共享和数据传递

无中间节点

网络工程师是做IP地址 路由 交换机工作 CCNA CCNP思科

127.0.0.1永远是本机

Unix /linux ifconfig查看IP

Windows ipifconfig 查看

Ping通则肯定通，不通不一定网络不通 ping [www.baidu.com](http://www.baidu.com)

ping 用来测试与目的IP地址的网络主机的网络是否通畅



端口

sudo lsof –i ：22 占用22号端口的相关服务的信息 ssh占22端口

飞秋端口 2425

HTTP 80 ssh22 端口作用 用以标示该系统的进程

socket（套接字） 原意-孔，插座

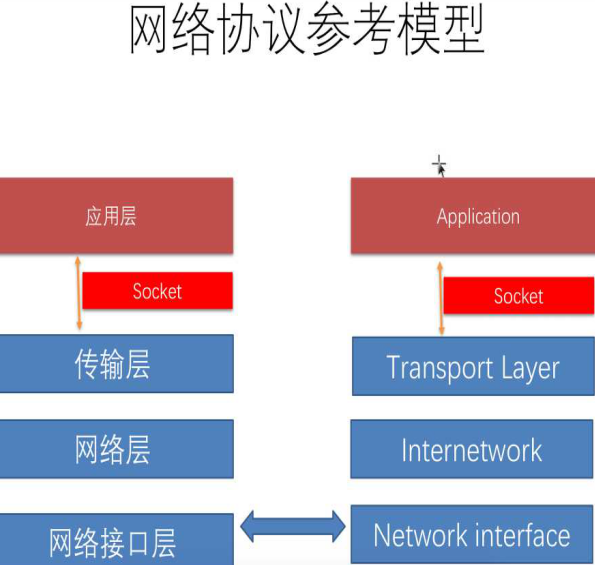
网络协议为什么要分层

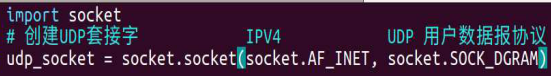
封装 下层封指定功能 上层直接使用

隔离 下层变动的时候还是实现指定的功能 不影响其他层使用

应用层--成寄快递 socket--朋友给你办 传输层--选择快递公司

网络层—选择运输方式 网络接口层—具体执行





记得关闭套接字 udp\_socket.close 一个进程所能打开的文件等系统资源是有限的

UDP创建 关闭

import socket

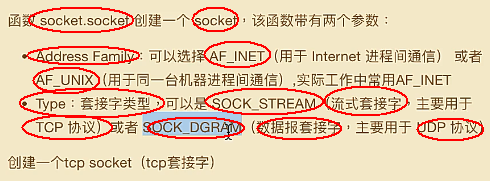
sock = socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM)创建一个套接字 返回的套接字对象 socket.AF\_INET is IPV4 socket.SOCK\_DGRAM是UDP用户数据报协议

sock.close()

sock.sendto()

飞鸽传书（IPMessenger） 免费软件，是一款局域网内即时通信软件，基于TCP/IP（UDP）；可运行于多种操作平台（Win、Mac、UNIX、Linux），并实现跨平台信息交流。不需要服务器支持 TCP用于文件下载传输 UDP用于发送消息





字符串的编码和解码

在linux中 系统默认编码都是UTF-8

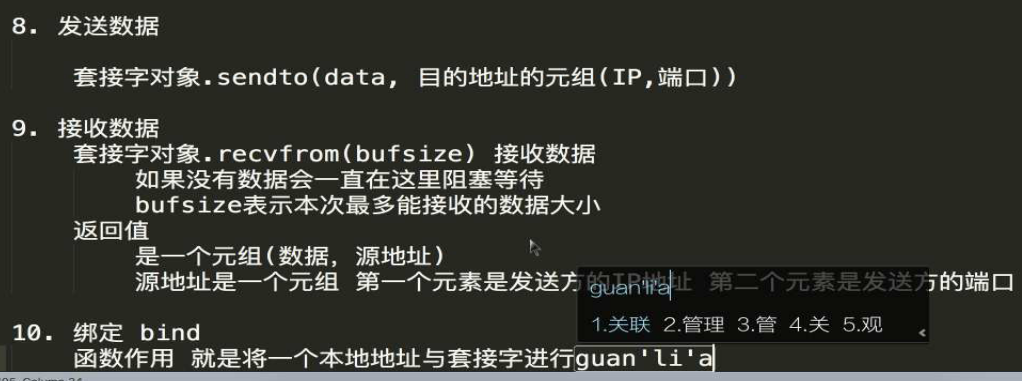
Str字符串类型

Python3中新增一个类型就是Bytes字节类型

Str ===””

Str –编码 str.encode()—》 bytes

《-----解码 bytes.decode()











import socket

# 创建UDP套接字 IPV4 UDP 用户数据报协议

udp\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

# 使用

data = input("请输入你要发送的数据:")

# 发送数据需要送到的 目的地址(IP, port)

dest\_address = ('192.168.20.157', 8080)

udp\_socket.sendto(data.encode(), dest\_address)

# 记得关闭 10000个 10001一个进程所能打开的文件等系统资源是有限制的

# 使用系统资源 关闭套接字

udp\_socket.close()

import socket

# 创建UDP套接字 IPV4 UDP 用户数据报协议

udp\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

while True:

# 使用

data = input("请输入你要发送的数据:")

# 发送数据需要送到的 目的地址(IP, port)

dest\_address = ('192.168.20.132', 8080)

udp\_socket.sendto(data.encode(), dest\_address)

# 接收数据(b'hahahahha', ('192.168.20.132', 8080))

# 是bytes类型的数据 远程地址(IP PORT)

data, remote\_address = udp\_socket.recvfrom(4096)

print("收到来自%s的数据:%s" % (str(remote\_address), data))

# 记得关闭 10000个 10001一个进程所能打开的文件等系统资源是有限制的

# 使用系统资源 关闭套接字

udp\_socket.close()

import socket

#echo 回射

udp\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

# 将本地端口８８８８占用　　放置操作系统随机分配

udp\_socket.bind(('', 8888))

while True:

data, remote\_address = udp\_socket.recvfrom(4096)

print("收到来自%s的数据:%s" % (str(remote\_address), data))

# udp\_socket.sendto("1234567890".encode(), ("192.168.20.132", 8080))

udp\_socket.sendto(data, remote\_address)

udp\_socket.close()

import socket

"""

在一个电脑中编写1个程序，有2个功能

1.获取键盘数据，并将其发送给对方

2.接收数据并显示

并且功能数据进行选择以上的2个功能调用

"""

# print(\_\_name\_\_)

def send\_msg(udp\_socket):

# 数据　ＩＰ　ＰＯＲＴ

data = input("数据:")

ip\_address = input("目的ＩＰ:")

port = int(input("目的端口:"))

dest\_address = (ip\_address, port)

udp\_socket.sendto(data.encode(), dest\_address)

def recv\_msg(udp\_socket):

data, remote\_addr = udp\_socket.recvfrom(4096)

print("接收到来自%s的数据:%s" % (str(remote\_addr), data.decode()))

if \_\_name\_\_ == '\_\_main\_\_':

# 创建ＵＤＰ套接字

udp\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

udp\_socket.bind(('', 8888))

while True:

op = input("１　发送数据\n2　接收数据　请选择:")

if op == '1':

send\_msg(udp\_socket)

elif op == '2':

recv\_msg(udp\_socket)

else:

print("不好意思　　你的请求去了火星")

TCP协议，传输控制协议（英语：Transmission Control Protocol，缩写为 TCP）是⼀种面向连接的、可靠的、基于字节流的传输层通信协议，由IETF的RFC 793定义。

TCP通信需要经过创建连接、数据传送、终止连接三个步骤

连接

单播 1对1 TCP UDP都支持

多播 组播 1对多 UDP

**TCP与UDP的不同点**

面向连接（确认有创建三方交握，连接已创建才作传输。）

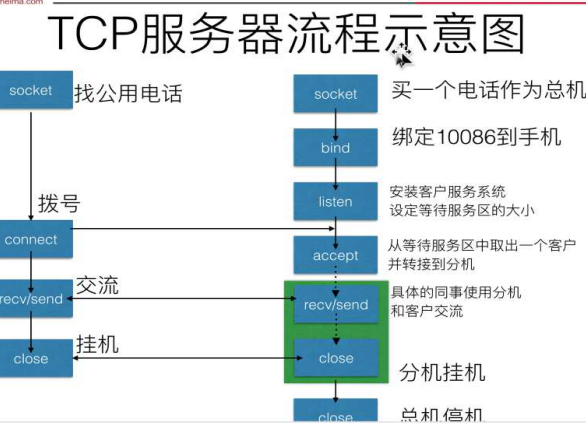
有序数据传输

重发丢失的数据包

舍弃重复的数据包

无差错的数据传输

阻塞/流量控制



1. 自动捕获异常
2. 在

主动断开连接的一方存在TIME\_WAIT

该TCP占有相关资源会保持几分钟 不能再使用

2MSL 倍 最大生命周期