网络功能：数据传输

ISO(国际标准化组织)

## OSI七层模型 ---网络通信工作流程的标准化

应用层：提供用户服务，具体功能由特定程序而定

表示层：提供数据的压缩优化，加密

会话层：建立应用级的连接，选择传输服务

传输层：提供不同的传输服务。流量控制

网络层：路由选择，网络互联

链接层：进行数据转换，具体消息的发送，链路连接【帧二进制】

物理层：物理硬件，接口设定，网卡路由交换机等

Cookie

高内聚：模块功能尽可能单一，不要掺杂

低耦合：模块之间尽可能减少关联和影响

OSI七层模型优点：

1. 将工作流程标准化
2. .降低了模块间的耦合度，每一部分可以独立开发，单独工作

四层模型（TCP/IP模型）：

应用层

传输层

网络层

物理链路层

## 数据传输流程

1. 发送端由应用层到物理层逐渐添加信息头（首部），最终在物理层发送
2. 中间经过节点（交换机，路由器等）转发，发送到接收端
3. 在接收端根据发送端的每个信息头进行解析，最终消息到应用层展示给用户

网络协议（信息头）：在网络通信中双方都遵循的规定。包括建立什么样的网络结构，消息结构，标识代表什么等。

应用层：TFTP HTTP DNS SMTP

传输层：TCP UDP

网络层：IP

物理层：IEEE

## 网络相关概念

网络主机：在网络中标识一台计算机 **HOST**

本地使用：’localhost’

‘127.0.0.1’

（ifconfig查看）

网络使用：’0.0.0.0’-自动选择一个ip

‘192.168.207.129’

获取计算机名：

host\_name = socket.gethostname()

获取计算机地址：

ip = socket.gethostbyname(host\_name)

### IP地址：网络上确定一台主机网络位置的地址

IPv4：点分十进制 192.168.1.2 0-255 32位

IPv6: 128位

ping ip：测试和某台网络主机是否连通

特殊IP地址

127.0.0.1 --本地测试IP

0.0.0.0 --使用本机可用IP

192.168.1.0 --表示网段IP

192.168.1.1 --默认网关IP（路由器IP）

192.168.1.255 --广播地址

**通过地址获取获取主机网络信息：socket.gethostbyaddr('localhost')**

**('localhost', [], ['127.0.0.1'])主机名，别名，网络地址**

**点分十进制转为二进制：socket.inet\_aton('192.168.1.2') –>b'\xc0\xa8\x01\x02'**

**二进制地址转为十进制：socket.inet\_ntoa(b'\xc0\xa8\x01\x02') 🡪 192.168.1.2**

## 域名：网络服务器地址的名称

名称如：[www.baidu.com](http://www.baidu.com) <> 119.75.217.26

1. 方便记忆
2. 名称表达一定含义

### 网络端口号

端口是网络地址的一部分，用于区分一个网络主机上的网络应用

\*在一个操作系统中不同的网络应用监听不同的端口号

取值范围：1—65535

1—255 一些众所周知的通用端口

256-1023 系统应用端口

1024-65535 自用端口

建议使用 >10000

**获取一个应用端口信息：socket.getservbyname('mysql') 🡪3306**

**网络字节序：数据在网络中的传输格式**

# 传输层服务

## 面向连接的传输服务

### 基于TCP协议的数据传输

**传输特征：提供可靠的数据传输，可靠性-数据传输过程无丢失、无失序、无差错、无重复**

**实现手段：数据传输断开前都需要进行连接和断开的确认**

**三次握手：tcp传输在数据传输前建立的过程**

1. **客户端向服务器发送连接请求**
2. **服务器收到请求后，回复确认消息，表示允许连接**
3. **客户端收到服务器回复，进行最终标志发送确认**

**四次挥手：tcp传输在连接断开前进行断开确认的过程**

1. **主动方发送报文告知对方要断开连接**
2. 被动方收到请求后立即返回报文告知已经准备断开
3. 被动方准备就绪后再次发送报文告知可以断开
4. 主动方发送消息，确认最终断开

应用情况：文件下载、网页获取、邮件传输、登录注册。

适用于传输较大文件、网络情况良好，需要保证传输可靠性的情况。

## 面向无连接的传输服务

### 基于udp协议的传输

**传输特征：不保证可靠的数据传输，传输过程无连接断开的流程，数据收发自由。**

**实现手段：数据传输断开前都需要进行连接和断开的确认**

应用情况：网络情况较差，对传输可靠性要求不高，需要提升传输效率。不便连接，需要灵活收发消息。

比如：网站视频、群聊、广播发送

### 广播

* 服务器端
  + # 设置可以发送接收广播
  + s.setsockopt(SOL\_SOCKET,SO\_BROADCAST,True)
  + s.sendto(data.encode(),addr)
* 客户端
  + # 选择接收地址
  + s.bind(('0.0.0.0',9999))
  + msg,addr = s.recvfrom(1024)

问题：

Osi七层模型？tcp/ip呢？

Tcp服务和udp服务有什么区别

三次握手和四次挥手是啥

# Socket套接字编程

流式套接字（SOCK\_STREAM）：传输层基于tcp协议的套接字编程方案

数据报套接字（SOCK\_DGRAM）：传输层基于udp协议的套接字编程方案

底层(SOCK\_RAM)：访问底层协议的套接字编程

### Tcp套接字服务端流程

Linux建立客户端访问服务器端- telnet 127.0.0.1 8848

### 网络收发缓存区

缓冲区作用：协调收发（处理）速度、减少交互次数

send和recv实际上是和缓存区进行交互，发送缓冲区空时就堵塞无法发送，接收缓冲区空时recv才阻塞

### TCP粘包

产生原因：tcp套接字以字节流方式传输，没有消息边界，发送和接收并不能保证每次发送都及时的被接收

影响：如果每次发送内容表达一个独立的含义，可能需要粘包

处理方法：

1. 每次发送的消息添加结尾标志（人为增加消息边界）
2. 发送数据结构体 如：“Y 18 abcd….”
3. 协调收发速度，每次发送后都预留接收时间

sys.argv 属性

功能：获取命令行参数，得到一个列表

命令本事是argv[0]，后面的参数从avgv[1]开始（默认以空格分隔，使用引号引起来的内容算一个整体）

在程序的第一行加

修改程序的执行权限

chmod 755 file.py

使用./file.py运行程序

### TCP套接字和udp套接字的区别

1. tcp需要进行listen accept 操作、udp不需要
2. tcp收发消息使用新的套接字，recv send。Udp recvfrom sendto

文件描述符：

每一个IO事件操作系统都会分配一个正整数作为编号，该正整数即为这个IO文件描述符

\*文件描述符是操作系统识别IO的唯一标志

sys.stdin 🡪0标准输入流

sys.stdout🡪1

sys.stdrr🡪2标准输错流

s.setsockopt(level,option,value)

功能：设置套接字选项，丰富或者修改套接字属性功能

参数：level 选项类别 SOL\_SOCKET

option具体选项

value选项值

HTTP协议：超文本传输协议，是一个应用层协议

用途：网页数据的传输、数据传输的方法

特点：应用层协议，传输层使用tcp服务

简单、灵活，多种语言都有http相关操作接口

无状态的协议，即不记录用户传输的信息

http1.1 支持持久连接

请求格式：

GET / HTTP/1.1

请求类别 请求内容（/主页）

**请求类别：**

**GET、POST**

**HEAD 响应头**

**PUT 更新服务器资源**

**DELETE 删除服务器资源**

**CONNECT**

**TRACE 用于测试**

**OPTIONS 获取服务器性能信息**

**请求头：对请求内容的具体描述信息（键值对形式）**

Accept-Encoding: gzip, deflate

Accept-Language: zh-CN,zh;q=0.9

Cache-Control: max-age=0

Connection: keep-alive

**空行**

**请求体：请求参数或者是提交内容**

**相应格式**

**响应行：反馈响应的情况**

**格式： HTTP/1.1 200 OK**

**响应码 附加信息**

**响应码：**

**1XX:提示信息，表示请求成功**

**2XX:响应成功**

**3XX:响应需要重定向**

**4XX:客户端错误**

**5XX:服务端错误**

**响应头：对响应内容的具体描述**

Connection: keep-alive

Content-Encoding: gzip

Content-Type: text/html

Date: Tue, 05 Mar 2019 08:37:04 GMT

**空行：**

**响应体：返回给请求端的具体内容**

## IO

* **在内存中存在数据交换的操作都可以认为是IO操作**
* **和终端交换：input print**
* **和磁盘交互：read write**
* **和网络交互：recv send**

### 程序分类

* **IO密集型程序：在程序执行过程中存在大量IO操作，而cpu运算操作较少。运行速度慢**
* **计算密集型程序（cpu密集程序）：在程序执行中cpu运算较多，IO操作相对较少。运行速度较快。**

#### IO模型

* **阻塞IO-IO默认形态，是效率较低的一种IO情形。**
  + **阻塞情况：**
    - **某种条件没达成（accept等待客户连接 input）**
    - **处理IO数据传输时间较长形成的阻塞（网络传输过程，文件读写过程）**
* **非阻塞IO：通过修改IO事件的属性，使其变为非阻塞状态。（让一些条件阻塞函数不再阻塞），需要手动捕获异常BlockingIOError，继续运行**
  + **.setblocking(False)**
  + **往往和循环判断一起使用**
* **超时检测：将原本阻塞的函数设置一个最长阻塞事件。如果时间内条件达成则正常运行，如果仍然阻塞则视为超时，继续向下运行或产生异常，需要手动捕获异常timeerror，继续运行**
  + **settimeout(5) 有些函数里有timeout参数 select([timeout])**
* **IO多路复用：同时监控多个IO事件，当哪个IO事件准备就绪就执行哪个IO事件。以此形成可用同时操作多个IO的并发行为，避免一个IO阻塞，造成所有IO都无法执行。**
* **IO准备就绪：是一种IO必然发生的零界状态。**
  + **select -- windows linux unix**
  + **poll -- linux unix**
  + **epoll -- linux unix**

### IO多路复用

#### select

* **rs,ws,xs = select(rlist,wlist,xlist[,timeout])**
* **功能：监控IO事件，阻塞等待IO事件发生**
  + **rlist --存放监控等待处理的IO事件 accpt()**
  + **wlist --存放要主动操作的IO事件 send()**
  + **xlist --要关注出错处理的IO事件**
* **返回值：任意IO准备就绪就返回对应IO列表**
* **wlist中有IO事件则select立即回返回为ws**
* **timeout超时时间设置**
* **在处理IO过程中不要处理一个客户端长期占有服务器端无法运行到select的情况**
* **IO多路复用占用计算机资源少，io效率**

**例： **

**telnet 172.40.80.250 8888**

#### 位运算

* **整数按照二进制位进行运算**
* **&按位与 |按位或 ^按位异或**
* **<<左移 >>右移10011**
  + - **11 -------1011**
    - **14 -------1110**
    - **& 1010 ---10**
    - **| 1111 – 15**
    - **^相同为0不同为1 101**
    - **11 << 2 101100 相当于\*（2\*\*）**
    - **14 >>2 10 相当于//（2\*\*）**

### poll

* **创建poll对象：p = select.poll()**
* **添加注册事件：p = p.register(s,POLLIN | POLLERR) 默认 所有事件**

**POLLIN1 POLLOUT POLLERR POLLHUP POLLNVAL**

**1 4 8 16 32**

**rlist wlist xlist 断开 无效数据**

* **p.unregister(s)从关注事件中移除**
* **阻塞等待IO发生**
  + - **events = p.poll() #阻塞等待IO发生**
    - **返回值：events是一个列表，列表中每个元素都是一个元组，代表一个发生的IO事件**
    - **[(fileno,event),(),()….]**
      * **(就绪IO的文件描述符,具体就绪事件)**
    - **通过文件描述符找到对应的IO对象：{s.fileno():s}**
* **处理具体的IO**

**例： **

### epoll

* **效率比select，poll高**
* **可以同时监控的IO数量比select poll多**
* **epoll的触发方式更多（EPOLLET边缘触发（事件未触发，下次在次触发的话一同处理）---反义：水平触发，事件触发必须处理）**

## Struct

* 生成结构化对象st = struct(fmt)
  + Fmt定制的数据结构
  + 如：要组织的数据 1 b’list’ 1.75 fmt:”4isf”
* 将一组数据按照指定格式打包转换：st.pack(v1,…)
  + v1…要打包的数据
  + 返回bytes字节串
* 将bytes按照指定格式解析：st.unpack(bytes\_data)
  + 解析后的数据元组
* struct.pack(fmt,v1,….) struct.unpack(fmt,bytes\_date)

### 本地套接字

* 本地两个程序之间的数据交换
* 通信原理：对一个内存对象进行读写操作，完成2个程序间的数据交互（代表内存的文件叫做套接字文件）
* s\_socked = socket(AF\_UNIX,SOCK\_STREAM)
* 绑定一个套接字文件：s\_socked.bind(file)
* 另外一端的客户端套接字也要使用这个套接字文件发送数据