IO网络编程

Tedu Python 教学部
Author：吕泽
Days：4天

- Linux操作系统有其组成
- shell命令
  - 文件操作命令
- IO
- 文件
  - 字节串（bytes）
  - 文件读写
  - 其他操作
    - 刷新缓冲区
    - 文件偏移量
    - 文件描述符
  - 文件管理函数
- 网络编程基础
  - OSI七层模型
  - 四层模型（TCP/IP模型）
    - 数据传输过程
    - 网络协议

      -网络基础概念
- 传输层服务
  - 面向连接的传输服务（基于TCP协议的数据传输）
  - 面向无连接的传输服务（基于UDP协议的数据传输）
- socket套接字编程
  - 套接字介绍
  - tcp套接字编程
    - 服务端流程
    - 客户端流程
    - tcp套接字数据传输特点
    - 网络收发缓冲区
    - aaa
    - 响应请求（request）
    - http响应（response）
  - struct模块的使用

Linux操作系统及其组成

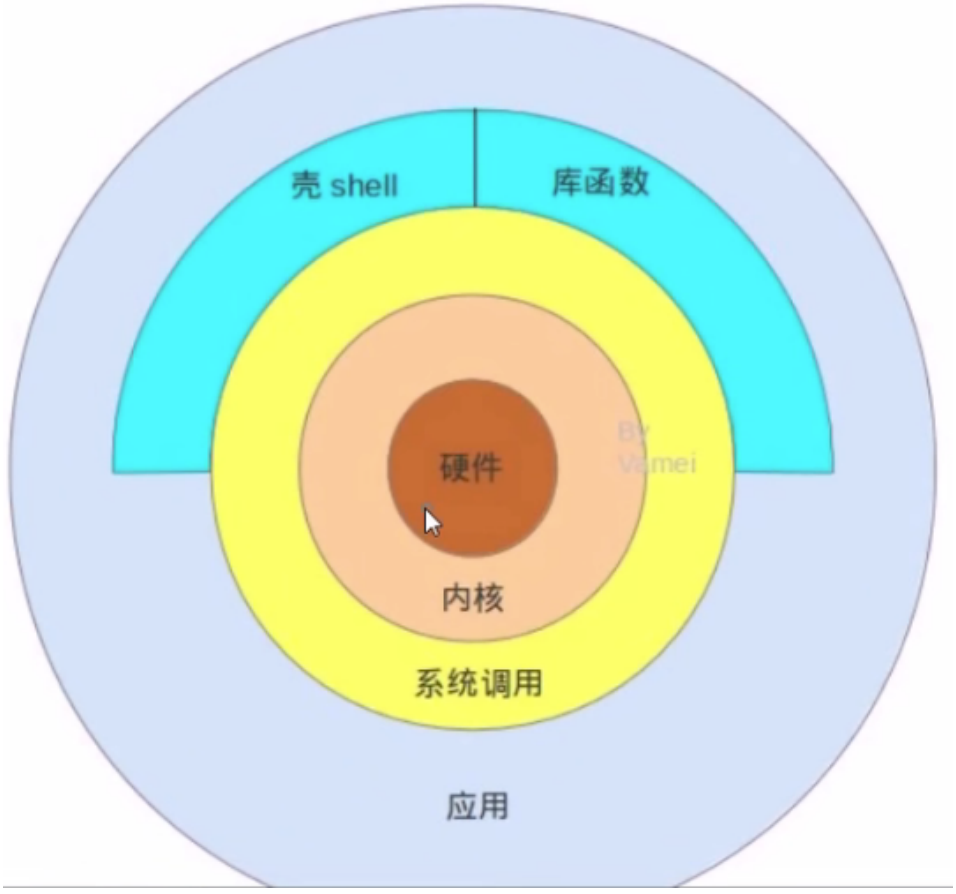
1.操作系统的作用

操作系统（ OS ）是管理计算机硬件与软件资源的计算机程序，同时也是计算机系统的内核与基石。操作系统需要处理如管理与配置内存、决定系统资源供需的优先次序、控制输入设备与输出设备、操作网络与管理文件系统等基本事务。操作系统也提供一个让用户与系统交互的操作界面。

2.Linux操作系统组成

一个典型的Linux操作系统组成为:Linux内核，文件系统，命令行shell，图形界面和桌面环境，并包括各种工具和应用软件。

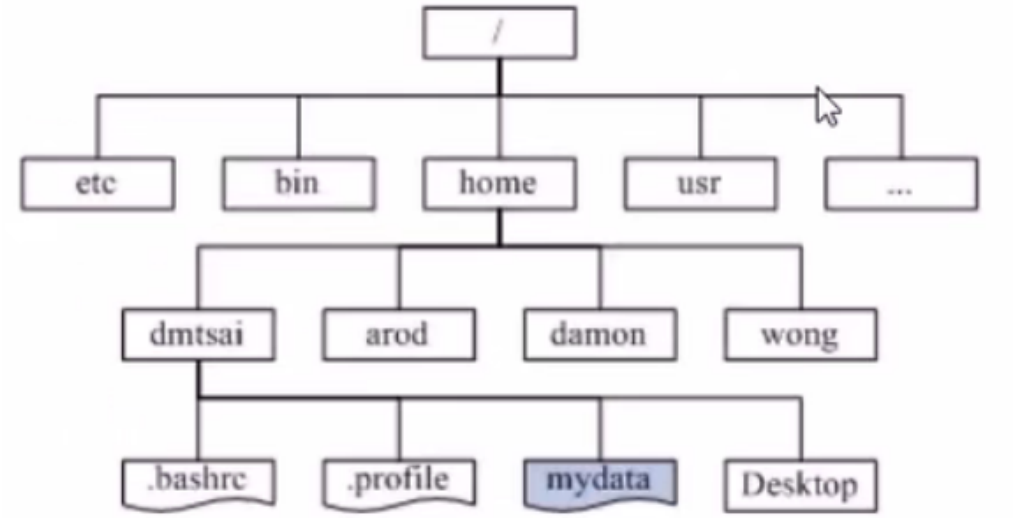
- Linux内核: Linux操作系统的核心代码
- 文件系统：通常指称管理磁盘数据的系统，可将数据以目录或文件的型式存储。每个文件系统都有自己的特殊格式与功能
- shell命令：接收用户命令，然后调用相应的应用程序，并根据用户输入的指令来反馈给用户指定的信息。



shell命令

文件操作命令

- linux下的目录结构



作用	命令
切换工作目录	cd
查看文件	ls, ls -l, ls -a
复制文件	cp -r
移动文件	mv
删除文件	rm -rf, rmdir
创建文件地夹	mkdir -p
创建文件	touch
查看文件内容	cat

- linux下凡是以点开头的文件或文件夹都是隐藏文件。
- linux的一条命令通常是由命令、选项、参数构成。

IO

1.定义

> 在内存中存在数据交换的操作认为是IO操作，比如和终端交互，和磁盘交互，和网络交互等

2.程序分类

> - IO密集型程序：在程序执行中有大量IO操作，而CPU运算较少。消耗CPU较少，耗时长。
> - 计算密集型程序：程序运行中计算较多，IO操作相对较少。CPU消耗多，执行速度快，几乎没有阻塞。

文件

文件是保存在持久化存储设备(硬盘、U盘、光盘..)上的一段数据。从功能角度分为文本文件(打开后会自动解码为字符)、二进制文件(视频、音频等)。在Python里把文件视作一种类型的对象，类似之前学习过的其它类型。

字节串（bytes)

在python3中引入了字节串的概念，与str不同，字节串以字节序列值表达数据，更方便用来处理二进程数据。因此在python3中字节串是常见的二进制数据展现方式。

- 普通的ascii编码字符串可以在前面加b转换为字节串，例如：b'hello'
- 字符串转换为字节串方法：str.encode()
- 字节串转换为字符串方法: bytes.decode()

文件读写

对文件实现读写的基本操作步骤为：打开文件，读写文件，关闭文件

代码实现：day1/file_open.py
代码实现：day1/file_read.py
代码实现：day1/file_write.py

1. 打开文件

```
file_object = open(file_name, access_mode='r', buffering=-1)
功能：打开一个文件，返回一个文件对象。
参数：file_name 文件名：
     access_mode 打开文件的方式，如果不写默认为'r'
           文件模式                          操作
               r                    以读方式打开 文件必须存在
               w                    以写方式打开
                                    文件不存在则创建，存在清空原有内容
               a                    以追加模式打开
               r+                   以读写模式打开 文件必须存在
               w+                   以读写模式打开文件
                                    不存在则创建，存在清空原有内容
               a+                   以读写模式打开 追加模式
               rb                   以二进制读模式打开 同r
               wb                   以二进制写模式打开 同w
               ab                   以二进制追加模式打开 同a
               rb+                  以二进制读写模式打开 同r+
               wb+                  以二进制读写模式打开 同w+
               ab+                  以二进制读写模式打开 同a+
     buffering 1表示有行缓冲，默认则表示使用系统默认提供的缓冲机制
返回值：成功返回文件操作对象。
```

> 缓冲：系统自动的在内存中为每一个正在使用的文件开辟一个缓冲区，从内存向磁盘输出数据必须先送到内存缓冲区，再由缓冲区送到磁盘中去。从磁盘中读数据，则一次从磁盘文件将一批数据读入到内存缓冲区中，然后再从缓冲区将数据送到程序的数据区。

1.读取文件

- read([size])

> 功能：用来直接读取文件中的字符。
> 参数：如果没有给定size参数（默认值为-1）或者size值为负，文件将被读取直到末尾，给定size最多读取给定数目个字符（字节）。
> 返回值：返回读取到的内容

> - 注意：文件过大时候不建议直接读取到文件末尾，读到文件结尾会返回空字符串。

- readline([size])

> 功能：用来读取文件中一行
> 参数：如果没有给定size参数（默认值为-1）或者size值为负，表示读取一行，给定size表示最多读取指定的字符（字节）。
> 返回值：返回读取到的内容

- readlines([sizeint])

> 功能：用来读取文件中的每一行作为列表的一项。
> 参数：如果没有给定size参数（默认值为-1）或者size值为负，文件将被读取直到末尾，给定size表示读取到size字符所在行为止。
> 返回值：返回读取到的内容列表

文件对象本身也是一个可迭代对象，在for循环中可以迭代文件的每一行。

```
for line in f:
    print(line)
```

3.写入文件

write(string)

> 功能：把文本数据或二进制数据块的字符串写入到文件中去
> 参数：要写入的内容
>
> - 如果需要换行要自己在写入内容中添加\n

writelines(str_list)

> 功能：接受一个字符串列表作为参数，将它们写入文件。
> 参数：要写入的内容列表

4.关闭文件

打开一个文件后我们就可以通过文件对象对文件进行操作了，当操作结束后使用close()关闭这个对象可以防止一些误操作，也可以节省资源。

> file_object.close()

5.with操作

python中的with语句使用于对资源进行访问的场合，保证不管处理中是否发生错误或者异常都会执行规定的“清理”操作，释放被访问的资源，比如有文件读写后自动关闭、线程中锁的自动获取和释放等。

with语句的语法格式如下：

```
with context_expression [as target(s)]:
    with-body
```

通过with方法可以不用close()，因为with生成的对象在语句块结束后会自动处理，所以也就不需要close了，但是这个文件对象只能在with语句块内使用。

```
with open("file", "r+") as f:
    f.read()
```

> 注意
>
> > 1.加b的打开方式读写要求必须都是字节串
> > 2.无论什么缓冲，当程序结束或者文件被关闭时，都会将缓冲区内容写入磁盘

其它操作

刷新缓冲区

代码实现：day02/buffer.py

> flush()
> 该函数调用后会进行一次磁盘交互，将缓冲区中的内容写入到磁盘

文件偏移量

代码实现：day02/seek.py

1.定义

> 打开一个文件进行操作时系统会自动生成一个记录，记录中描述了我们对文件的一系列操作。其中包括每次操作到的文件位置。文件的读写操作都是从这个位置开始进行的。

2.基本操作

tell()

> 功能：获取文件偏移量大小

seek(offset[,whence])

> 功能：移动文件偏移量位置
> 参数：offset代表相对于某个位置移动的字节数。负数表示向前移动，正数表示向后移动。
> whence是基准位置的默认值为0，代表从文件开头算起，1代表从当前位置算起，2代表从文件末尾算起。
>
> - 必须以二进制方式打开文件时基准位置才能是1或者2

文件描述符

1.定义

> 系统中每一个IO操作都会分配一个整数作为编号，该整数即这个IO操作的文件描述符。

2.获取文件描述符

fileno()

> 通过IO对象获取对应的文件描述符

文件管理函数

1.获取文件大小

os.path.getsize(file)

2.查看文件列表

os.listdir(dir)

3.查看文件是否存在

os.path.exists(file)

4.判断文件类型

os.path.isfile(file)

5.删除文件

os.remove(file)

网络编程基础

计算机网络功能主要包括实现资源共享，实现数据信息的快速传递。

OSI七层模型

> 制定组织：ISO（国际标准化组织）
>
> 作用：使网络通信工作流程标准化
>
> 应用层：提供用户服务，具体功能有应用程序实现
> 表示层：数据的压缩优化加密
> 会话层：建立用户级的连接，选择适当的传输服务
> 传输层：提供传输服务
> 网络层：路由选择，网络互联
> 链路层：进行数据交换，控制具体数据的发送
> 物理层：提供数据传输的硬件保证，网上接口，传输介质

优点

> 1. 建立了统一的工作流程
> 2. 分部清晰，各司其职，每个步骤分工明确
> 3. 降低了各个模块之间的耦合度，便于开发

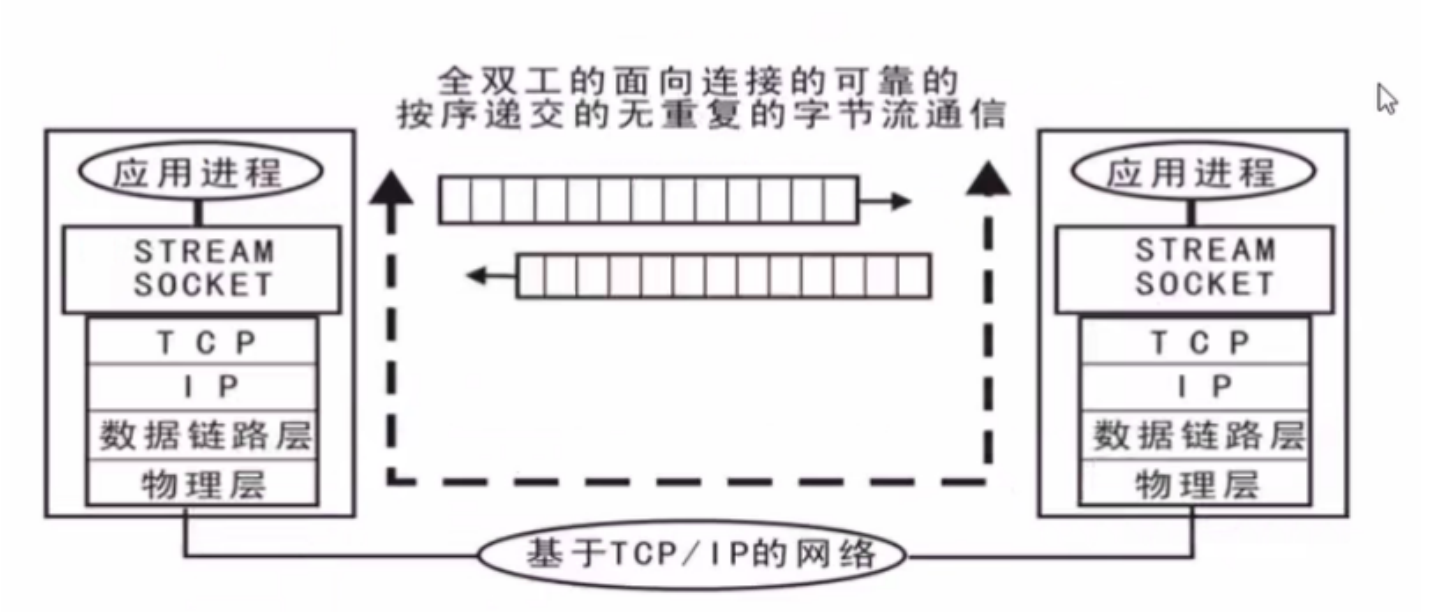
四层模型

背景：实际工作中工程师无法完全按照七层模型要求操作，逐渐演化成更符合实际情况的四层

OSI七层网络模型	TCP/IP四层概念模型	对应网络协议
应用层	应用层	TFTP,FTP,NFS,WAIS
表示层	应用层	Telnet,Rlogin,SNMP,Gopher
会话层	应用层	SMTP,DNS
传输层	传输层	TCP,UDP
网络层	网际层	IP,ICMP,ARP,RARP,AKP,UUCP
数据链路层	网络接口	FDDI,Ethernet,Arpanet,PDN,SLIP,PPP
物理层	网络接口	IEEE 802.1a,IEEE 802.2到IEEE 802.11

数据传输过程

1. 发送端由应用程序发送消息，逐层添加首部信息，最终在物理层发送消息包。
2. 发送的消息经过多个节点（交换机，路由器）传输，最终到达目标主机。
3. 目标主机由物理层逐层解析首部消息包，最终到应用程序呈现消息。



网络协议

> 在网络数据传输中，都遵循的规定，包括建立什么样的数据结构，什么样的特殊标志等。

网络基础概念