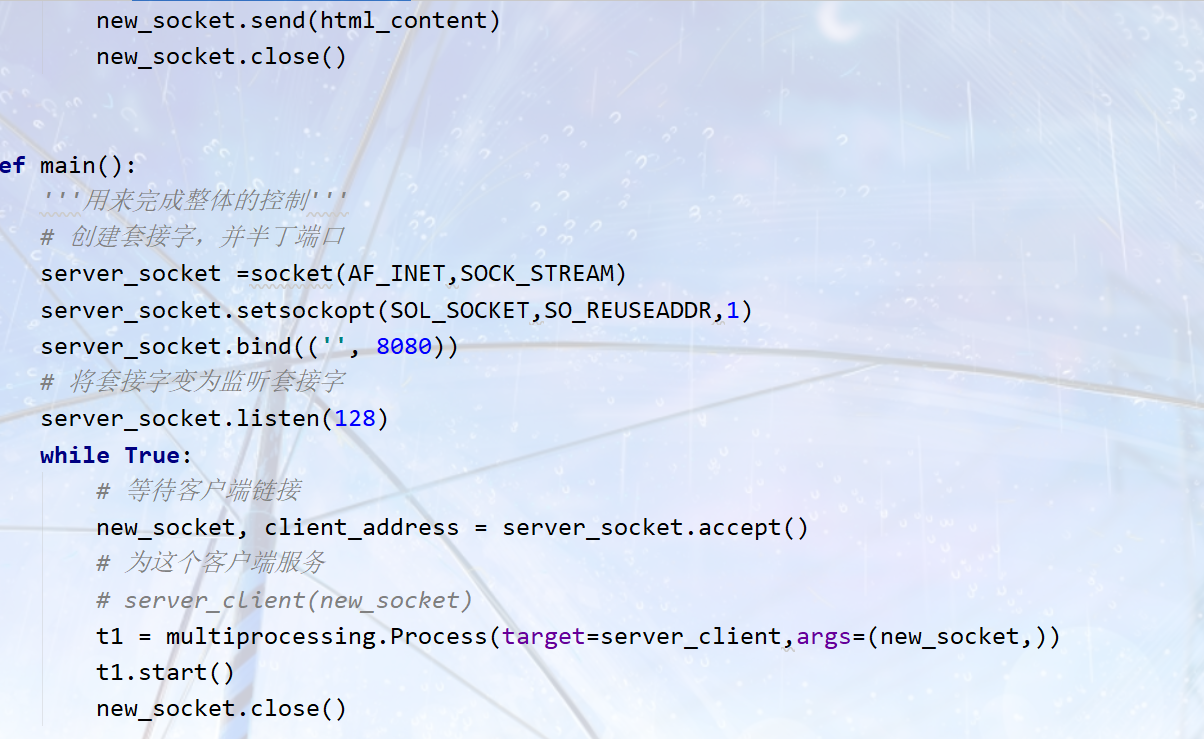
所有的Linux里面的东西都对应一个文件

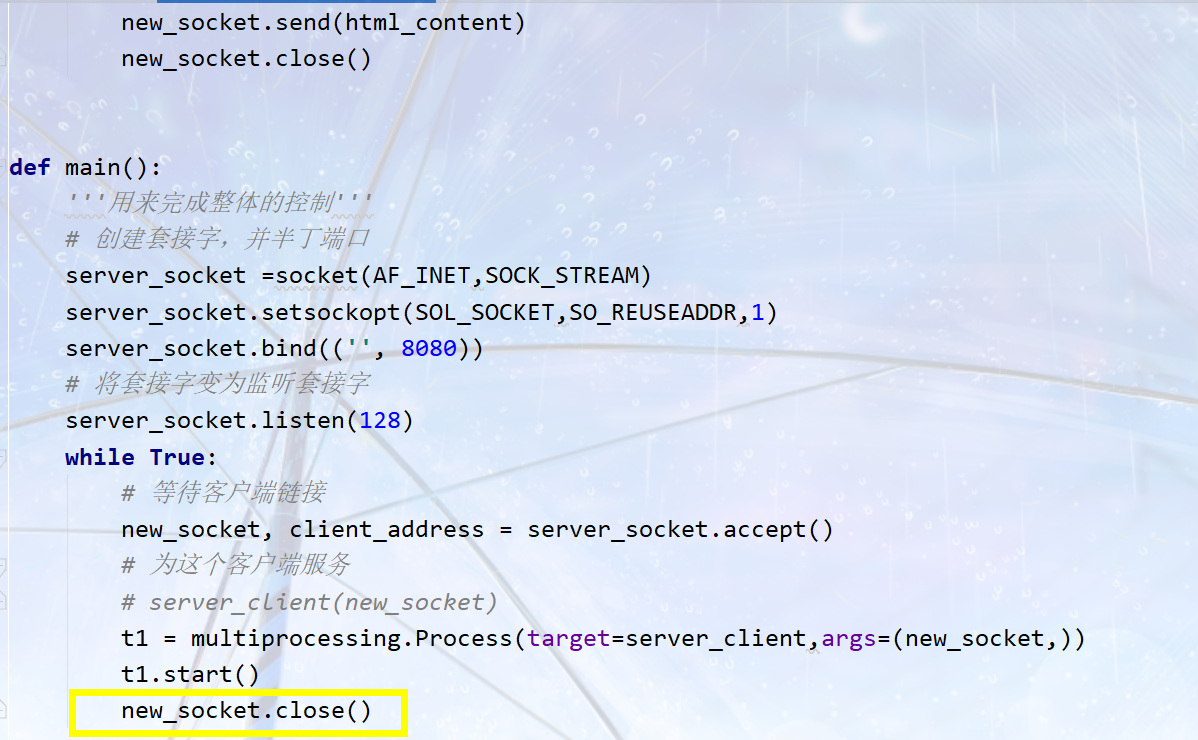
以下代码是多线程实现http服务器



以下代码是使用多进程实现http服务器



两种方式的差别即



在多进程实现中，主进程中多了一个new\_socket.close()

原因：之前所学，在多进程，创建一个子进程中会复制原有的代码，那么相当于有两个new\_socket()的来指向new\_socket，那么如果只调用一次close，那么还有一个new\_socket()，则这个客户端的端口无法关闭，即无法执行四次挥手，也就是意味着客户端会一直等待服务器发送信息，而服务器却不再发送

以下是使用协程实现http服务器(注意：join不可以少，并且需要导入monkey)



一般来说：在现实工作中，协程会优于线程和进程

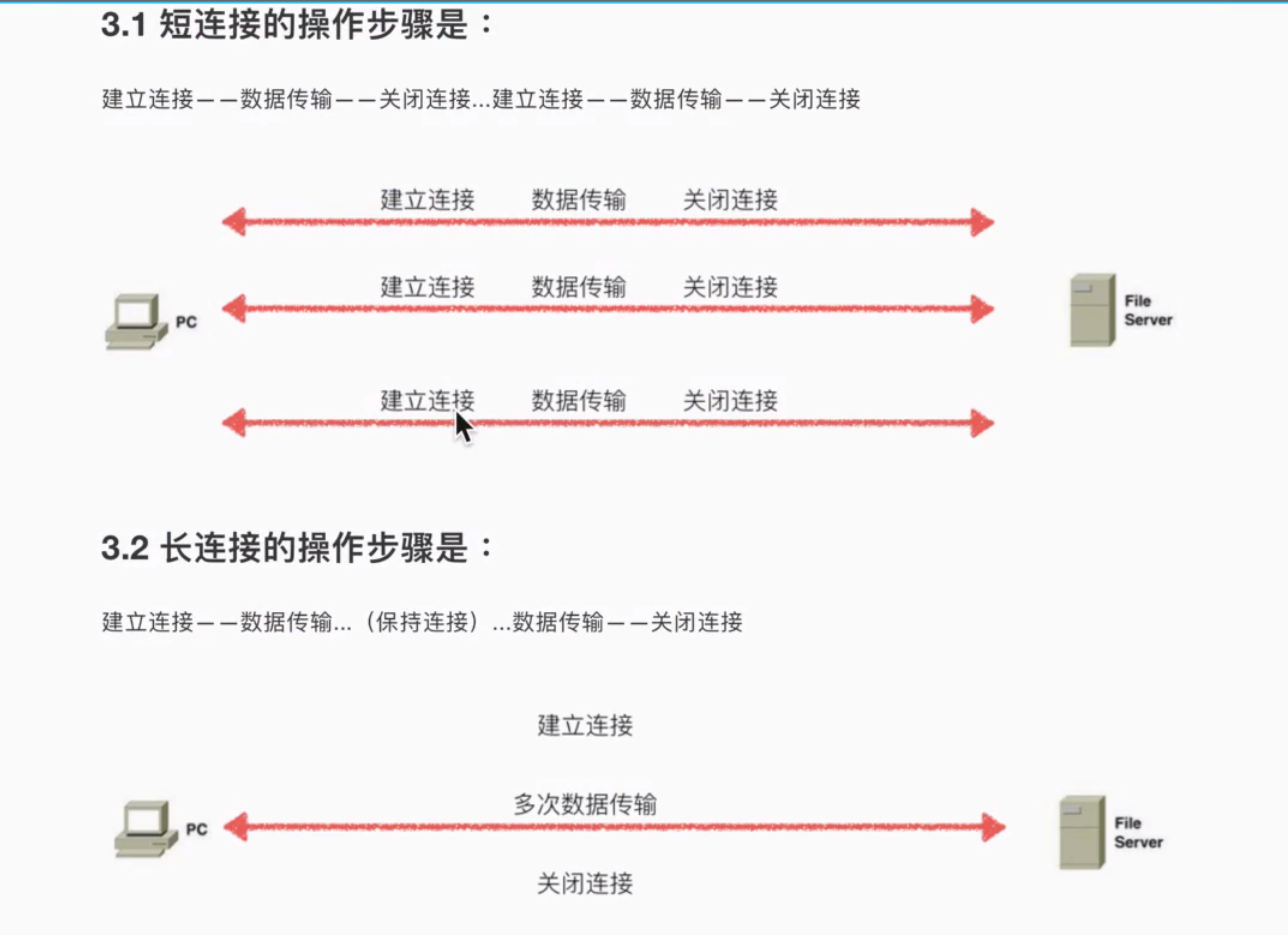
注意：server\_socket.setblocking(False) # 设置套接字为非堵塞的方式

. new\_socket.setblocking(False) # 设置套接字为非堵塞的方式

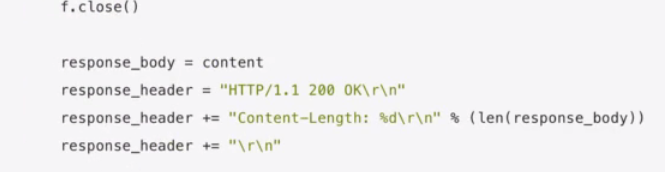
该方式不仅仅可以适用于创建的tcp套接字，还可以作用于客户端的套接字，如果该套接字堵塞了，那么就会报错

HTTP/1.1 采用的是长连接

HTTP/1.0 采用的是短连接

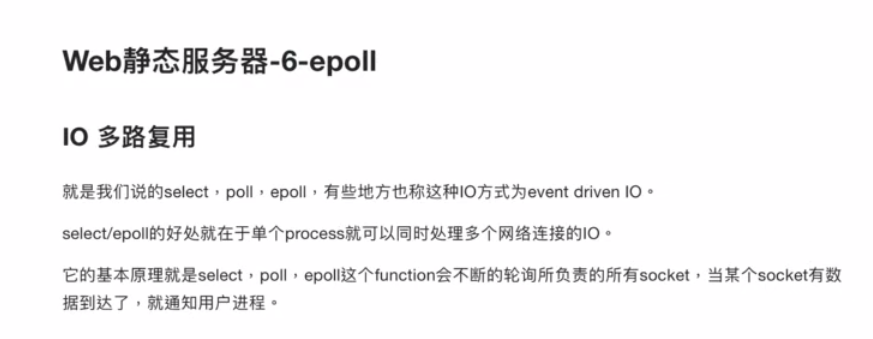


注意：在长连接过程中如何让浏览器知道本次数据已经发送完毕？



在HTTP发送的response中有一个“Content-Length:”

而这个就是用来表示本次发送body的长度，那么浏览器接收到之后就会停止等待本次等待



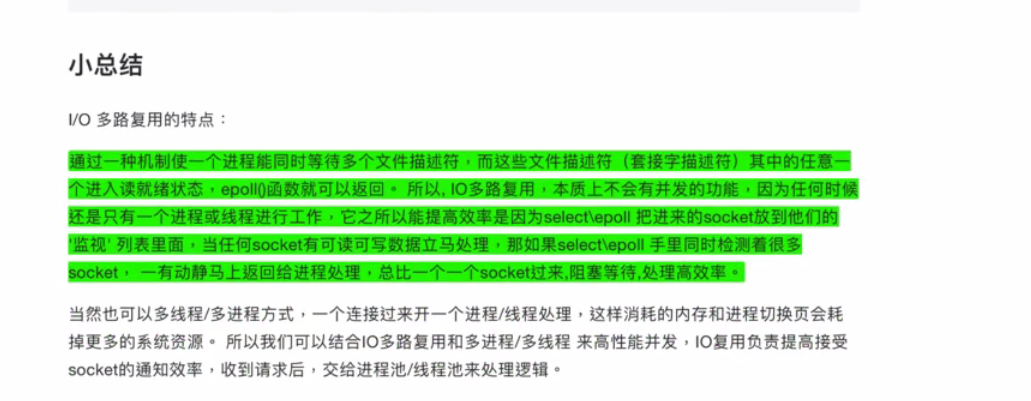
我们在开发过程中，把第一个遍历到最后一个的方法叫轮询，轮询的效率相当低

比轮询效率更高的叫事件通知

通俗理解为：一厨师问谁饿了，轮询的方式就是一个一个问过去，而事件通知的方式就是谁饿了谁告诉我

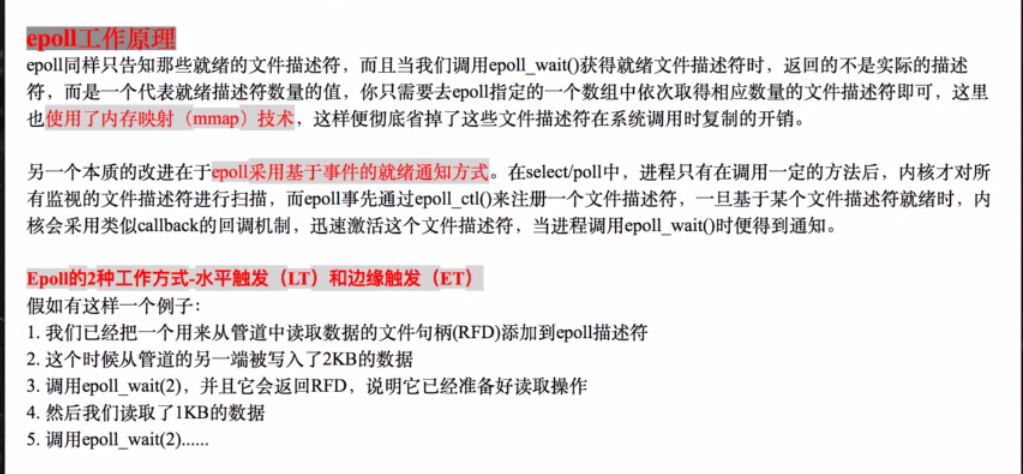
Epoll的特点：

1. 共享内存，即服务器和操作系统共享内存
2. 以事件通知的方式来替换轮询的方式



如想了解，可以参考以下网站





在python中有个select模块

可以通过epl = select.epoll()来创建一个epoll对象

通过epl.register()可以将指定内容传入epoll

注意，传入的必须是对应的文件

将监听套接字传入：

epl.register(tcp\_server\_socket.fileno(),select.EPOLLIN|select.EPOLLET)

在一个对象后加上.fileno()，可以找到指定对象对应的文件

第二个参数，select.EPOLLIN表示有消息进来就事件通知

Fd\_event\_list = eql.poll() # 创建一个poll对象，默认是堵塞的，直到os检测到数据到来，通过事件通知方式告诉这个程序，此时才会解堵塞

poll()返回的是一个由fd和event组成的列表

返回的列表对象大致类型由元组组成

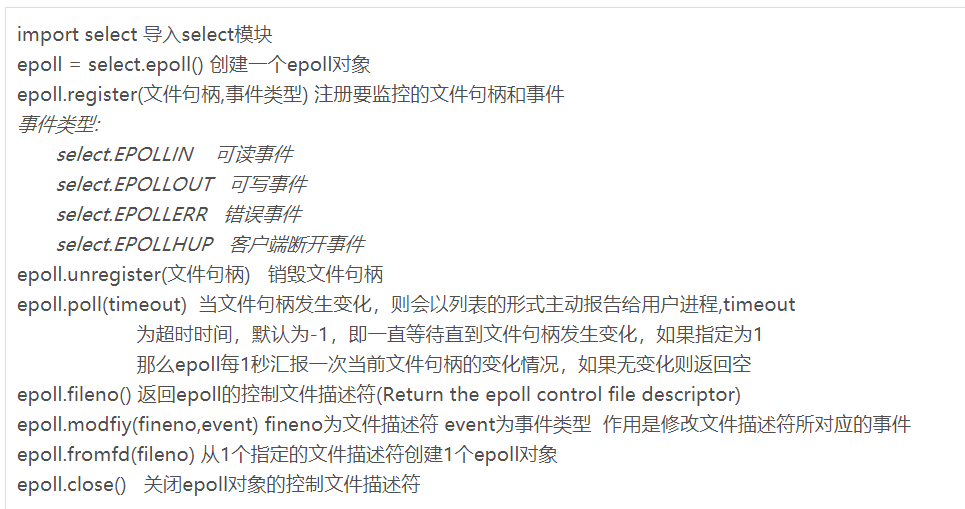
[(fd,event),（fd1，event1），（fd2，event2）……]

Fd表示：套接字对应的文件描述符

Event表示这个文件描述符到底是什么事件：例如可以调用recv接收等

注意：给操作系统一个套接字，它可以通过fileno()找到对应的文件，反之，给操作系统一个fileno()文件，操作系统无法找到对应的套接字

Register是将对象放入epoll里面，而unregister是将对象从epoll里面拿出



Python中的epoll一般应用于Linux

因此，在win下的select中是没有epoll的