课程回顾:

昨天 : TCP编码,udp编码,pycharm输出带颜色,自定义类解决解码编码的问题

ACK : 确认收到

SYN : 请求连接的这么一个标识

FIN : 请求断开的这么一个标识

三次握手:客户端先发起

客户端先发起连接请求

服务器回复确认收到, 连接客户端的请求

客户端回复收到请求, 可以连接

四次挥手:谁都可以先发起请求

客户端发起一个请求,代表我没有数据继续发送了,但是如果你有数据继续发,我可以继续接收

服务器发送一个确认收到的ACK

服务器再发送一个断开连接的请求,表示可以断开连接了

客户端回复一个确认收到

UDP特点: 不面向连接,不可靠,面向数据报,速度快

TCP特点 : 可靠的,基于连接的,面向字节流形式的

OSI五层模型 : 应用层,传输层,网络层,数据链路层,物理层

socket是一个模块,是一个套接字,是一个类,是传输层和应用层之间的一个抽象层

子网掩码 : 子网掩码 & IP地址 得到网段

今日内容:

1 执行命令:

在py代码中如何去调用操作系统的命令

新模块 : subprocess

r = subprocess.Popen(**'ls'**,

shell=**True**,

stdout=subprocess.PIPE,

stderr=subprocess.PIPE)  
*# subprocess.Popen(cmd,shell=True,subprocess.stdout,subprocess.stderr)  
# cmd : 代表系统命令  
# shell = True 代表这条命令是 系统命令,告诉操作系统,将cmd当成系统命令去执行  
# stdout 是执行完系统命令之后,用于保存结果的一个管道  
# stderr 是执行完系统命令之后,用于保存错误结果的一个管道*

print(r.stdout.read().decode(**'gbk'**))

print(r.stderr.read().decode(**'gbk'**))

**粘包问题** : 只有tcp协议才会发送粘包,udp不会发生

EX: 发送端发送数据,接收端不知道应该如何去接收,造成的一种数据混乱的现象

在tcp协议中,

有一个合包机制(nagle算法),将多次连续发送且间隔较小的数据,进行打包成一块数据传送.

还有一个机制是拆包机制,在发送端,因为受到网卡的MTU限制,会将大的超过MTU限制的数据,进行拆分,拆分成多个小的数据,进行传输. 当传输到目标主机的操作系统层时,会重新将多个小的数据合并成原本的数据

针对 使用udp协议发送数据,一次收发大小究竟多少合适?

udp不会发生粘包,udp协议本层对一次收发数据大小的限制是:

65535 - ip包头(20) - udp包头(8) = 65507

站在数据链路层,因为网卡的MTU一般被限制在了1500,所以对于数据链路层来说,一次收发数据的大小被限制在 1500 - ip包头(20) - udp包头(8) = 1472

得到结论:

如果sendto(num)

num > 65507 报错

1472 < num < 65507 会在数据链路层拆包,而udp本身就是不可靠协议,所以一旦拆包之后,造成的多个小数据包在网络传输中,如果丢任何一个,那么此次数据传输失败

num < 1472 是比较理想的状态

作业:

0 整理博客,把粘包现象 + 两个机制 一定要搞懂

1 解决subprocess执行命令时,产生的粘包现象

2 把文件上传下载的 做完

3 做一个切换目录的功能

用户输入一个绝对路径,以此路径为基础(此时服务器应该返回这个目录下的所有文件及文件夹)

如果用户继续输入 .. 服务器要返回上一层的所有文件及文件夹

如果用户输入的是cd 后边一定要求用户继续输入一个当前目录下的文件夹的名字. 此时 服务器需要给用户返回该文件夹下的所有文件及文件夹