# 操作系统

**页面置换应用场景，算法有哪些**

应用场景：需要调入新页到内存，但内存已满，选择页进行换出

先进先出算法（first in first out, FIFO）

最久未使用算法（Least Recently Used, LRU）

最不常用算法（Least Frequently Used, LFU）

时钟置换算法（clock）

**物理内存，虚拟内存和共享内存的区别**

物理内存：系统的物理内存被划分为许多大小相同的部分，也叫内存

虚拟内存：当线程启动，每个进程会建立一个虚拟内存地址，存放堆栈等数据

共享内存：进程运行共存中，会加载一些公共库，是每个进程公用的，这部分叫共性内存

**32位和64位操作系统的区别**

指CPU一次处理的最大位数

**硬中断和软中断的区别**

硬中断：由外部计算机的硬件产生的中断，是随机的，使用场景：

软中断：执行中断指令产生的中断，是程序安排的，使用场景：

**CPU占用率和负载的区别**

cpu占用率(使用率)：指程序在运行期间实际占用CPU的百分比

cpu负载：

**键盘敲入 A 字母时，操作系统期间发生了什么**

I/O获取

**内/外存读写速度**

内存60GB/s,外存100MB/s

**异步IO**

发起I/O请求（如读文件、收网络数据）后，不等待结果，而是立即返回并继续执行其他逻辑。I/O完成后，操作系统通过回调函数、事件通知等方式通知程序，程序再去处理结果

# 计算机网络

**http请求有几种方式**

幂等性：多次执行的效果与执行一次的效果相同

GET方式：用来获取数据(幂等)

POST方式：用来提交表单（不幂等）

PUT方式：用来修改数据(幂等)

DELETE方式：用来删除数据(幂等)

**什么是BS架构，CS架构**

B/S(Browser/Server)：浏览器/服务器程序

C/S(Client/Server)：客户端/服务端程序

**网络协议有哪些**

http：超文本传输协议

ftp：文件传输协议

smpt：简单邮件协议

pop3：邮件读取协议

**OSI五层模型**

应用层：为应用程序提供服务

会话层：会话管理和同步，设置检查点

表示层：加解密，格式转换，压缩

运输层：端到端传输， udp, tcp

网络层：ip选址及路由选择

ARP地址解析协议：从网络层的IP地址解析出在数据链路层使用的硬件地址

链路层：可靠传输，将数据报封装成帧，透明传输，差错控制,传输单位：帧

PPP点对点协议：只检测差错而不纠正差错，不进行流量控制，

物理层：物理线上传输(双绞线，光缆)

**IPv4和IPv6的区别**

IPv4的IP地址长度是32，IPv6的IP地址长度是128,安全性更高

**常见的通信方式**

单工通信：消息只能单方向传输(只能接/发)

半双工通信：可以实现双向的通信(能用来接/发，同一时刻只能是一种方式)

全双工通信：可以实现双向的通信(能用来接/发，可以同时进行)

**UDP和TCP**

UDP用户数据报协议:传输数据报，面向无连接，尽最大努力交付，不保证可靠传输，多对多

TCP传输控制协议:传输报文，面向连接，一对一，先建立链接，再进行可靠传输(无差错，不丢失，不重复，按需到达)

**UDP是什么协议，应用场景有哪些**

面向无连接的，不可靠传输，数据报传输，尽最大努力交付

使用场景：语言和视频传输和广播信息(给每个点发送信息)等，允许一部分数据的丢失

**为什么UDP比TCP快**

udp无连接的，没有流量控制，没有三次握手等机制，只要发数据就行

**udp发送方发的太快，接收放接收不过来，导致丢包，怎么解决**

增加接收缓冲区大小；或者发一部分后，间隔一秒再发

**TCP是什么协议，应用场景有哪些**

面向连接的，可靠传输，字节流传输

使用场景：http请求，文件传输等，需要保证可靠传输

**TCP和UDP的区别**

tcp是连接的，udp是无连接的

tcp是字节流传输，udp是报文传输

tcp是可靠传输(不丢失，不重复，按序到达，效率低)，udp是不可靠传输（尽最大努力交付，效率高）

tcp是点到点的，udp支持一对多，多对多传输

tcp首部开销20字节，udp首部开销8字节

**TCP如何保证可靠传输（**6个机制）

校验和

序列号和确认应答

超时重传

连接管理：三次握手和四次挥手

流量控制

拥塞控制

**序列号(seq)和确认应答(ack)**

序列号：TCP传输时将每个字节的数据进行编号

确认应答：TCP传输中，接收到数据后，会发送ACK报文给发送方进行应答，其中包含确认序列号，表示收到哪些数据

**超时重传**

TCP发送端长时间未得到ACK报文会进行重传

情况1：由于网络原因，传输的包丢失，接收端没收到，重传后，接收方收到后会发出ack报文

情况2：由于网络原因，丢失的是ack报文，重传后，接收方发现了重复的包，直接丢弃，仍然发出ack报文

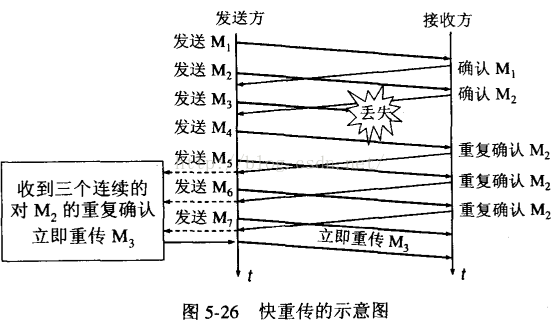
弊端：长等待时间影响传输效率，短了会导致频繁重复发送包，该时间为动态计算。

Linux中，以500ms为单位进行控制，每次重发后以等待时间\*2，指数递增，累计到一定重传次数后，TCP会认为网络或服务端 异常，强制关闭连接

**快重传**

由于超时重传中存在等待超时的过程，比较浪费时间

快重传，收到报文段后立即发出重复确认，未收到M3的报文段，则之后接受别的报文段都发ack消息确认收到M2，当连续出现3次冗余的M2重复确认,则客户端知道了M3报文段发生失败，立即重新发送M3报文段。



**TCP三次握手和四次挥手是什么**

三次握手(SYN建立联接，ACK确认)

客户端发送syn消息(seq(序列号)=j)给服务器，表示建立握手信号，等待服务器确认

服务器收到syn消息，发送ack消息(ack=j+1)和syn消息(seq=k),给客户端，服务器进入接收状态

客户端检查ack是否正确，发送ack包(ack=k+1)给服务器，服务器检查正确则建立连接

四次挥手

客户端发送fin消息(seq = j)，表示不发送报文了

服务端发送ack消息(seq=j+1),告知客户端已知道，同时需要等待服务端接收完所有数据

服务端发送,fin消息(seq = k),表示准备关闭

客户端发送 ack消息(seq=k+1)，确认关闭连接

**四次挥手能否合并步骤2，3改成3次握手**

关闭时服务端收到FIN后需要先发送ACK消息，让客户端等待服务端接受完所有的报文，然后再发送FIN消息给客户端。

步骤2发送ack消息时，只表示收到了关闭连接的请求，但是可能并未接收完所有报文，合并步骤2，3会导致一些包的丢失。

能否改为两次握手？服务端发送ack和syn消息给客户端，但客户端不会理会，会让服务端一直等待下去，浪费资源，所以不能改为二次握手

客户端第一次握手发送的syn消息因为网络问题而延迟到达客户端。

如果最后一次握手或者挥手时的包丢失了会发生什么

客户端发送的ack消息丢失了，服务端收不到ack消息，会超时重传ack消息，直到客户端再次发送ack消息成功，则建立/关闭连接

**TCP流量控制**

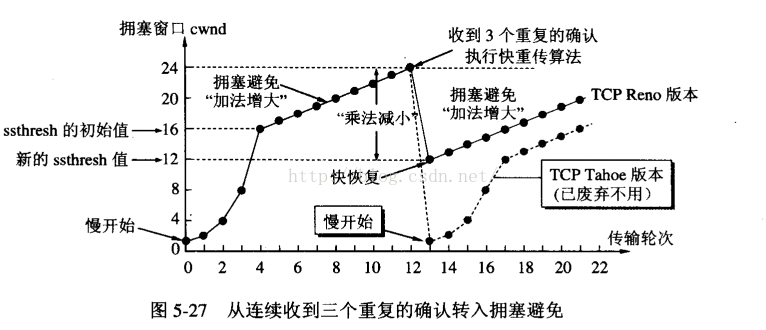
避免发送端发送太快，接收端接受不及时导致丢包

接收端给告知发送端最多允许发多少数据(发送窗口大小)，发送端发的数据不能超过发送窗口大小

**TCP现拥塞控制**

拥塞控制过程：慢启动过程中，拥塞窗口大小指数增长，到达慢启动阈值ssthresh时线性增长

重复收到3次冗余确认，触发快重传，同时进入快恢复状态，拥塞窗口大小变为原来的一半，然后线性增长



**http和https的区别**

HTTP是超文本传输协议，明文传输

HTTPS是具有安全性的SSL加密传输协议

**get和post的区别（都是TCP连接）**

get请求参数放在url中，有长度限制，比较不安全，主要向服务器请求数据

post请求参数放在request body中，无长度限制，比较安全，主要是向服务器传送数据（增删改）

**浏览器状态**

1XX：表示临时响应并需要请求者执行操作才能继续的状态代码。

2XX：请求成功

3XX：需要进行重定向

4XX:客户端错误

5XX:服务端错误

**浏览器页面是进程还是线程**

谷歌打开标签页是创建一个进程，其他浏览器是创建一个线程

创建进程的好处：进程更加安全，一个进程崩了不影响其他进程

创建进程的坏处：进程的启动，关闭，切换会比线程由更多的开销。

**浏览器输入网址到返回页面的过程，期间用到了哪些协议**

1.域名解析(DNS域名解析协议)

2.浏览器和目标服务器建立TCP连接

3.浏览器向服务器发送http请求（HTTP协议）

4.服务器接收请求并返回相应资源

应用层：调用接口返回报文

传输层：TCP协议进行传输

网络层：路由到对应IP地址上

数据链路层：将数据报封装成帧

5.释放TCP连接

6.浏览器解析接收到的资源(HTMl,CSS,JS，图片等)

7.浏览器进行页面渲染

**XXS攻击**

将恶意的js,sql代码注入到页面中，对系统造成损伤(比如：宕机)

**一个MTU最多能包含多少数据**

ip头部为20字节，数据最多：1480字节