**一些经验贴**

IMG_256IMG_256

1. **我为进软院在大学做过啥准备**
2. **中断的处理过程**

中断在操作系统中有着重要地位，它是多道程序得以实现的基础，没有中断就没有多道程序，因此进程之间的切换是通过中断来完成的。此外，为了提高处理机的利用率和实现CPU与I/O设备的并行执行，也必须要有中断的支持。

中断和陷入

中断：是指CPU对I/O设备发来的中断信号的一种响应。CPU保护暂停当前进程，转而去处理引发中断的进程。执行完后，再返回到原进程，继续执行。I/O设备可以是字符设备（键盘），块设备（磁盘）或通信设备（网络）等。由于中断是外部造成的，所以又叫外中断。

陷入：显然，陷入是由CPU进程内部事件引发的中断，因此叫陷入或内中断。其产生的原因往往是由于进程本身出现运行时异常。

两者的区别仅仅是信号的来源

中断向量表和中断优先级

1.中断向量表

用于记录相应中断处理程序入口地址的映射表。表中有为中断类型分配的中断号和相对应的中断处理程序的入口地址。当中断来临时，由中断控制器确定中断号，再到表中寻找程序。

2.中断优先级

对于并发的中断请求，必须要给出合理的中断顺序，比如键盘中断和磁盘中断谁更重要等，要给出解决中断的优先级。

多中断的处理方式

试想一种情况：当处理机正在处理键盘引起的终端，此时又收到了高优先级的磁盘中断，应该如何处理？

1.屏蔽（禁止）中断

对于新来的中断，在当前中断未完成之前，对于其他中断不予理睬，直到当前处理完。采用先来先服务的方式。

2.嵌套中断

中断的中断。中断处理程序也是进程，当更高优先级的中断处理程序到来时，可以剥夺当前中断处理进程的处理机，及中断的中断。此种情况下，CPU总是优先响应更高优先级的中断请求。

中断处理程序的处理过程

1.测定是否有未响应的中断信号。当设备完成I/O后，设备控制器会向处理机发送中断信号，告诉I/O完成。之后要测试在中断请求队列中是否有未响应的请求。

2.保护被中断进程的CPU环境。保存现场，将进程的CPU信息全部载入进程PCB中（堆栈区），然后切换进程。

3.转入相应的设备处理程序。根据中断号找到相应程序的入口，交给设备驱动程序。

4.中断处理。检测设备控制器中相应设备的状态（正常、异常），并给予相应的解决。

5.恢复CPU现场并退出中断。如果是屏蔽中断，返回被中断进程；嵌套中断，检查优先级队列，根据是否有更高优先级中断请求，选择返回原中断还是处理更高优先级中断；

6.检测中断是否处理正确，向上层软件确定结果，报告等后续操作。

2.微机：**512\*4的芯片，要组成4M的存储空间要用几个芯片级联，具体用多少引脚。**

3.电子线路：逻辑电路去异步时序电路的区别

后来老师告诉第二题的答案，我问了一句,能不能麻烦老师告诉我第三题的答案啊，这个想不起来了好遗憾，

4.8259是干什么，具体内部怎么实现的

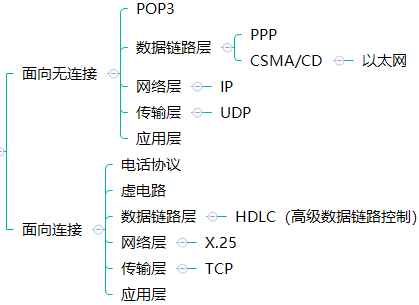
答：可编程中断控制器。   再问，那具体内部怎么实现的，我说了它与CPU连接的原理，老师说不对要内部的。。。我就不会了。。。后来老师说就到这里可以了

然后就是一顿自我介绍，注意了**自我介绍很重要**，由于那天人很多你们自我介绍一定要突出重点，本人就是得了个c语言竞赛的奖，还有学习成绩什么的，还有个中级设计师证书，然后表示对软件有兴趣，学习态度也好，很期待来软院充实的学习。

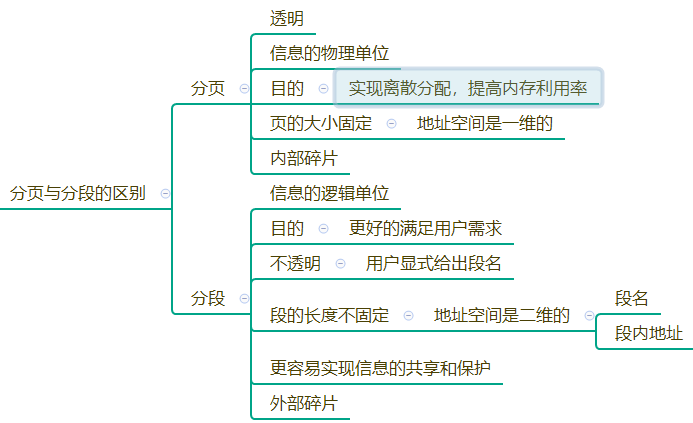
1. **说出网络任意一层**



|  |  |  |
| --- | --- | --- |
| **物理层** | 协议 | CDMA（码分多路复用） |
| 任务 | 1.解决如何在连接各个计算机的传输媒体上传输数据比特流，而不是指具体的传输媒体  2.确定与传输媒体接口有关的一些特性 |
| **数据链路层** | 协议 | HDLC（高级数据链路控制）【特点：1.面向比特2.同步】  PPP |
| 任务 | 1.将源自网络层的数据可靠的传输到相邻结点的目标网络层  2.加强物理层传输原始比特流的功能  3.将物理层提供的可能出错的物理连接改造成逻辑上无差错的数据链路  4.使之对网络层表现为一条无差错的链路 |
| **网络层** | 协议 | ICMP（网际报文控制协议），IGMP，IP，ARP，RARP |
| 任务 | 1.把分组从源端传递到目的端  2.为分组交换网上的主机提供通信服务  3.单位：数据报  4.提供主机与主机之间的逻辑通信  5.功能  1.路由选择与分组转发（最佳路径）  2.异构网络互连  3.拥塞控制  1.开环控制  2.闭环控制 |
| **传输层** | 协议 | TCP，OSPF，UDP  SCTP（流控制传输协议）【可靠，端点到端点】 |
| 任务 | 1.使用网络层的服务，为应用层提供通信服务  功能  1.提供进程和进程之间的逻辑通信  2.复用和分用  3.对收到的报文进行差错检测  4.两种协议 |
| **应用层** | 协议 | DHCP（动态主机配置协议）  BGP，RIP，HTTP，FTP，SMTP  TFTP（简单文件传输协议）【基于UDP，端口号：69，传输8位数据，功能有限】 |
| 任务 | 1.对应用程序的通信提供服务  功能  1.文件传输，访问和管理  2.电子邮件  3.虚拟终端  4.查询服务和远程作业登录 |



1. **操作系统页表和段表的区别**



1. **数据库6个设计阶段**

系统需求分析阶段；   
 概念结构设计阶段；   
 逻辑结构设计阶段；   
 物理结构设计阶段；   
 数据库实施阶段；   
 数据库运行与维护阶段；

1. **编译时候静态编译和动态编译有什么区别**。最后把表3交给美女就可以走了。

**动态编译**的[可执行文件](http://baike.baidu.com/view/159830.htm" \t "https://blog.csdn.net/Stephen_yu/article/details/_blank)需要附带一个的[动态链接库](http://baike.baidu.com/view/887.htm" \t "https://blog.csdn.net/Stephen_yu/article/details/_blank)，在执行时，需要调用其对应动态链接库中的命令。

优点一方面是缩小了执行文件本身的体积，一方面是加快了编译速度，节省了[系统资源](http://baike.baidu.com/view/53557.htm" \t "https://blog.csdn.net/Stephen_yu/article/details/_blank)。缺点一是哪怕是很简单的程序，只用到了链接库中的一两条命令，也需要附带一个相对庞大的链接库；二是如果其他计算机上没有安装对应的[运行库](http://baike.baidu.com/view/1032404.htm" \t "https://blog.csdn.net/Stephen_yu/article/details/_blank)，则用动态编译的可执行文件就不能运行。

**静态编译**就是[编译器](http://baike.baidu.com/view/487018.htm" \t "https://blog.csdn.net/Stephen_yu/article/details/_blank)在编译可执行文件的时候，将可执行文件需要调用的对应动态链接库(.so)中的部分提取出来，链接到可执行文件中去，使可执行文件在运行的时候不依赖于动态链接库。编译慢，兼容性好。

B组下午

1. **CSMA/CA和CSMA/CD的英文**

**Carrier Sense Multiple Access with Collision Detection**，意思是**载波侦听多路访问／冲突检测**。

采用该协议要求设备在发送帧的同时要对信道进行侦听，以确定是否会发生信道冲突，若在发送数据过程中检测到冲突，则需要进行冲突处理。整个协议的处理规程如下：

1 监听当前信道上是否有数据再发送，如果信道空闲，直接发送数据，如果信道忙，则按照一定的退避算法进行延时监听。

2 当信道允许发送数据时，发送数据。

3 数据发送过程中，边发送边监听，如果发送过程中检测到冲突，则停止发送数据，并发送阻塞信息，强化冲突，并转入1。

---------------------

**Carrier Sense Multiple Access with Collision Avoidance**，意思是**载波侦听多路访问／冲突避免。**

采用该协议要求设备要主动避免冲突而非被动侦测的方式来解决冲突问题。避免冲突的方法主要有两个：

一是监听到信道空闲时，并不是立即发送，而是等待一段时间再发送数据。

二是先发送一个很小的信道侦测帧RTS，如果收到最近的接入点返回的CTS，就认为信道是空闲的，然后再发送数据。

协议的主要流程如下：

1 首先检测信道是否有使用，如果检测出信道空闲，则等待一段随机时间后，才送出数据。

2 接收端如果正确收到此帧，则经过一段时间间隔后，向发送端发送确认帧ACK。

3 发送端收到ACK帧，确定数据正确传输，在经历一段时间间隔后，再发送数据。

---------------------

区别如下:

1.**两者的传输介质不同**,CSMA/CD用于总线式以太网,而[CSMA/CA](https://www.baidu.com/s?wd=CSMA/CA&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)则用于无线局域网802.11a/b/g/n等等；

2.**检测方式不同**,CSMA/CD通过电缆中电压的变化来检测,当数据发生碰撞时,电缆中的电压就会随着发生变化；而[CSMA/CA](https://www.baidu.com/s?wd=CSMA/CA&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)采用能量检测(ED)、载波检测(CS)和能量载波混合检测三种检测信道空闲的方式。

3.CSMA/CD:带有[冲突检测](https://www.baidu.com/s?wd=%E5%86%B2%E7%AA%81%E6%A3%80%E6%B5%8B&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)的载波监听多路访问，可以检测冲突，但无法“避免”。

[CSMA/CA](https://www.baidu.com/s?wd=CSMA/CA&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)：带有冲突避免的载波监听多路访问，发送包的同时不能检测到信道上有无冲突，只能尽量“避免”。

1. **操作系统中进程和线程的区别**

|  |  |
| --- | --- |
| 进程 | 1. 进程是资源分配的基本单位，进程是进程实体的运行过程，是系统进行资源分配和调度的独立单位 2. 进程的特征：动态性，并发性，独立性，异步性，结构性【程序段，数据段，进程控制块PCB（）】 |
| 线程 | 1.线程是独立调度的基本单位 |
| 区别 | 123 |

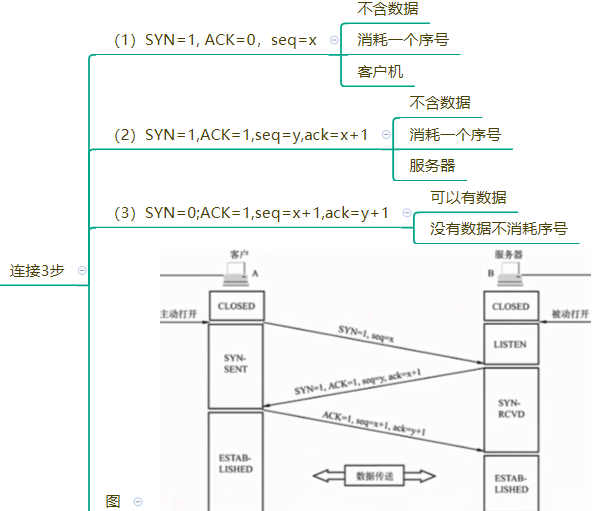
1. **C语言中传值和传址的区别**

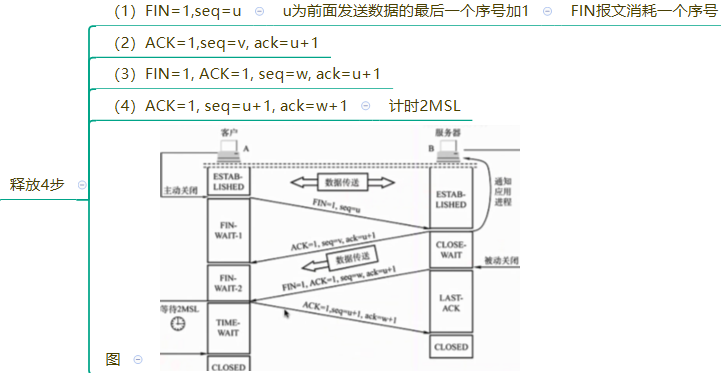
传值，实际是把实参的值赋值给行参，相当于copy。那么对形参的修改，不会影响实参的值 。

传址： 实际是传值的一种特殊方式，只是他传递的是地址，不是普通的赋值，那么传地址以后，**实参和形参都指向同一个对象**，因此对形参的修改会影响到实参。

专有名词：Internet of Things，缩写IoT

1. **tcp三次握手的过程**





**13图论**

**14生成树    最小生成树**

一个有 n 个结点的[连通图](https://baike.baidu.com/item/%E8%BF%9E%E9%80%9A%E5%9B%BE/6460995" \t "https://baike.baidu.com/item/%E6%9C%80%E5%B0%8F%E7%94%9F%E6%88%90%E6%A0%91/_blank)的生成树是原图的极小连通子图，且包含原图中的所有 n 个结点，并且有保持图连通的最少的边。最小生成树可以用[kruskal](https://baike.baidu.com/item/kruskal/10242089" \t "https://baike.baidu.com/item/%E6%9C%80%E5%B0%8F%E7%94%9F%E6%88%90%E6%A0%91/_blank)（克鲁斯卡尔）算法或[prim](https://baike.baidu.com/item/prim/10242166" \t "https://baike.baidu.com/item/%E6%9C%80%E5%B0%8F%E7%94%9F%E6%88%90%E6%A0%91/_blank)（普里姆）算法求出。



1. **什么是形参，什么是实参及其区别**

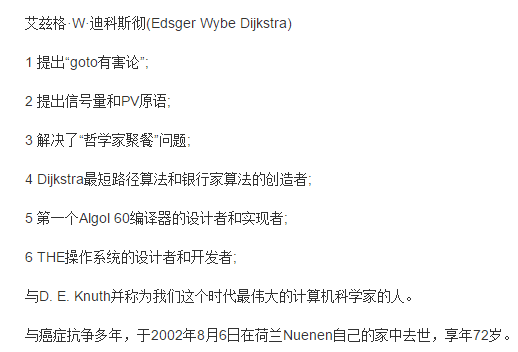
传值，实际是把实参的值赋值给行参，相当于copy。那么对形参的修改，不会影响实参的值 。

传址： 实际是传值的一种特殊方式，只是他传递的是地址，不是普通的赋值，那么传地址以后，**实参和形参都指向同一个对象**，因此对形参的修改会影响到实参。

1. **C语言能传地址吗？**

不能，可以传形参和指针。Cpp可以传地址

16.1.迪杰斯特拉算法



1. **什么是二叉排序树，什么是平衡二叉树，如何调整**



**18.计算机网络分层的缺点**    说分层的缺点是不稳定，下层出了问题会影响上层

**19.http的英文全称和acid是什么**

**HTTP--Hyper Text Transfer Protocol**，超文本传输协议，是一种建立在TCP上的无状态连接，整个基本的工作流程是客户端发送一个HTTP请求，说明客户端想要访问的资源和请求的动作，服务端收到请求之后，服务端开始处理请求，并根据请求做出相应的动作访问服务器资源，最后通过发送HTTP响应把结果返回给客户端。其中一个请求的开始到一个响应的结束称为事务，当一个事物结束后还会在服务端添加一条日志条目。

ACID，指数据库事务正确执行的四个基本要素的缩写。包含：

**原子性（Atomicity）**整个事务中的所有操作，要么全部完成，要么全部不完成，不可能停滞在中间某个环节。事务在执行过程中发生错误，会被[回滚](https://baike.baidu.com/item/%E5%9B%9E%E6%BB%9A" \t "https://baike.baidu.com/item/acid/_blank)（Rollback）到事务开始前的状态，就像这个事务从来没有执行过一样。

**一致性（Consistency）**事务必须始终保持系统处于一致的状态

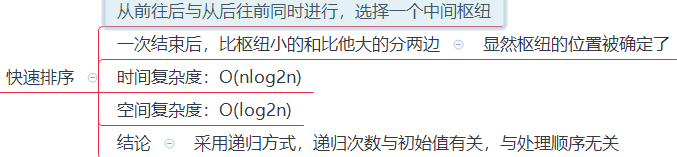
**隔离性（Isolation）**使得在同一时间仅有一个请求用于同一数据。

**持久性（Durability）**在事务完成以后，该事务对数据库所作的更改便持久的保存在数据库之中，并不会被回滚。

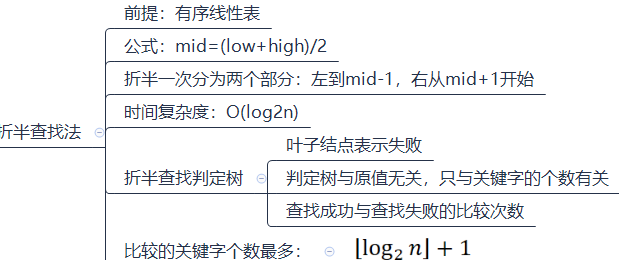
一个支持事务（Transaction）的数据库，必须要具有这四种[特性](https://baike.baidu.com/item/%E7%89%B9%E6%80%A7/3128227" \t "https://baike.baidu.com/item/acid/_blank)，否则在事务过程（Transaction processing）当中无法保证数据的正确性，交易过程极可能达不到交易方的要求。

1. **快速排序和C语言的函数调用**

# C语言函数调用三种方式：传值调用，引用调用和传地址调用



1. **折半查找的过程**



1. **PV操作是什么**

在计算机操作系统中，PV操作是进程管理中的难点。  
首先应弄清PV操作的含义：PV操作由P操作原语和V操作原语组成（原语是不可中断的过程），对信号量进行操作，具体定义如下：  
    P（S）：①将信号量S的值减1，即S=S-1；  
           ②如果S³0，则该进程继续执行；否则该进程置为等待状态，排入等待队列。  
    V（S）：①将信号量S的值加1，即S=S+1；  
           ②如果S>0，则该进程继续执行；否则释放队列中第一个等待信号量的进程。  
PV操作的意义：我们用信号量及PV操作来实现进程的同步和互斥。PV操作属于进程的低级通信。

什么是信号量？信号量（semaphore）的数据结构为一个值和一个指针，指针指向等待该信号量的下一个进程。信号量的值与相应资源的使用情况有关。当它的值大于0时，表示当前可用资源的数量；当它的值小于0时，其绝对值表示等待使用该资源的进程个数。注意，信号量的值仅能由PV操作来改变。  
     一般来说，信号量S³0时，S表示可用资源的数量。执行一次P操作意味着请求分配一个单位资源，因此S的值减1；当S<0时，表示已经没有可用资源，请求者必须等待别的进程释放该类资源，它才能运行下去。而执行一个V操作意味着释放一个单位资源，因此S的值加1；若S£0，表示有某些进程正在等待该资源，因此要唤醒一个等待状态的进程，使之运行下去。

1. **根据网络的覆盖范围，把网络分为几类**
2. 计算机网络按覆盖范围分类可分为**广域网，城域网，局域网**。

局域网是将小区域内的各种[通信设备](https://www.baidu.com/s?wd=%E9%80%9A%E4%BF%A1%E8%AE%BE%E5%A4%87&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)互连在一起的通信网络。

2、广域网特点

1）覆盖的地理区域大，通常在几公里至几千、几万公里，网络可跨越市、地区、省、国家、洲洋乃至全球。

2）广域网连接常借用公用网络。

3）[传输速率](https://www.baidu.com/s?wd=%E4%BC%A0%E8%BE%93%E9%80%9F%E7%8E%87&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)比较低，

4）网络拓扑结构复杂。

3、城域网特点

是在**一个城市范围内所建立的[计算机通信](https://www.baidu.com/s?wd=%E8%AE%A1%E7%AE%97%E6%9C%BA%E9%80%9A%E4%BF%A1&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)网**，简称MAN。这是80年代末，在LAN的发展基础上提出的，在技术上与LAN有许多相似之处，而与广域网(WAN)区别较大。

1）传输速率高。

2）用户投入少，接入简单。

3）技术先进、安全。

**24.团队合作很重要，说说软件系统设计怎么通过合作进行**？

**25.const除了静态变量以外还有什么用处**

1、想要阻止一个变量被改变，可以使用const关键字。在定义该const关键字是，通常要对它进行初始化，因为以后再也没有机会去改变它。

2、对于指针来说，可以指定指针本省为const，也可以指定指针所指向的数据为const，或者二者同时指定为const。

3、在一个函数声明中，const可以修饰形参，表明它是一个输入参数，在函数内部不能改变其值。

4、对于类的成员函数，若指定为const，则表明其实一个常函数，不能修改类的成员变量。

5、对于类的成员函数，有时候必须制定其返回值为const，以使得其返回值不能为左值。

**26.快排是怎么实现的**

它的基本思想是：通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。

一趟快速排序的算法是：

1）设置两个变量i、j，排序开始的时候：i=0，j=N-1；

2）以第一个数组元素作为关键数据，赋值给key，即key=A[0]；

3）从j开始向前搜索，即由后开始向前搜索(j--)，找到第一个小于key的值A[j]，将A[j]和A[i]互换；

4）从i开始向后搜索，即由前开始向后搜索(i++)，找到第一个大于key的A[i]，将A[i]和A[j]互换；

5）重复第3、4步，直到i=j； (3,4步中，没找到符合条件的值，即3中A[j]不小于key,4中A[i]不大于key的时候改变j、i的值，使得j=j-1，i=i+1，直至找到为止。找到符合条件的值，进行交换的时候i， j指针位置不变。另外，i==j这一过程一定正好是i+或j-完成的时候，此时令循环结束）。

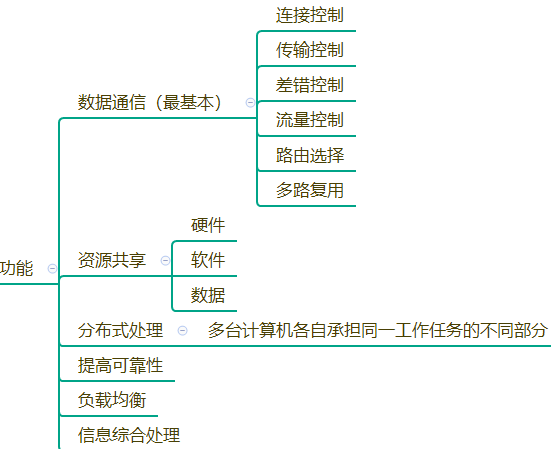
**27.网络操作系统做的课程设计**

**28.微机原理里面的输入输出方式有哪些**

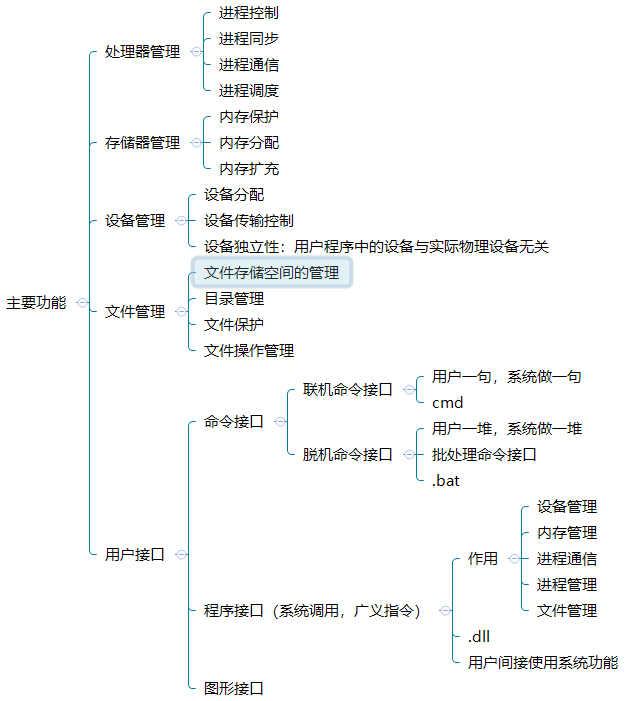
**29.DMA和通道方式的区别是什么？**

**通道控制（Channel Control）**方式与[DMA方式](https://www.baidu.com/s?wd=DMA%E6%96%B9%E5%BC%8F&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)类似，也是一种以内存为中心，实现设备和内存直接交换数据的控制方式。  
与[DMA方式](https://www.baidu.com/s?wd=DMA%E6%96%B9%E5%BC%8F&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)不同的是，在DMA方式中，数据的传送方向、存放数据的内存始址以及传送的数据块长度等都由[CPU控制](https://www.baidu.com/s?wd=CPU%E6%8E%A7%E5%88%B6&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)，而在通道方式中，这些都由通道来进行控制。另外，DMA方式每台设备至少需要一个[DMA控制器](https://www.baidu.com/s?wd=DMA%E6%8E%A7%E5%88%B6%E5%99%A8&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)，一个通道控制器可以控制多台设备。

1. **网络的功能是什么**

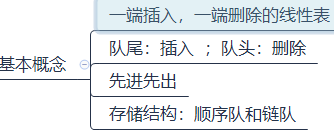
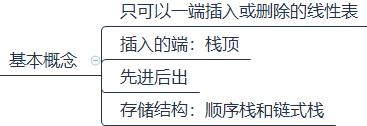


1. **操作系统的几大功能是什么**



**32.进程管理管理的是什么**

**33.栈和队列有什么区别**



1. **生活中有哪些常见栈和队列的应用**

**栈的运用**：1.使用栈检测括号是否匹配 2．转置应用。一串数据全部进入栈后再全部出栈，将得到一个次序完全相反的数据串，这是栈的特性之一。因此，栈可以应用于转置操作，例如，数的进制转换等。3．迷宫问题 4.网络浏览器多会将用户最近访问过的网址组织为一个栈。这样，用户每访问一个新页面，其地址就会被存放至栈顶；而用户每按下一次“后退”按钮，即可沿相反的次序访问此前刚访问过的页面。5.类似地，主流的文本编辑器也大都支持编辑操作的历史记录功能6.Java的虚拟机

**队列的应用**：1．共享打印机 2．消息队列3.多进程共享的 CPU时间。4.无论是调度银行和医院的服务窗口5.还是管理轮耕的田地和轮伐的森林

1. **把字符串类型的“321”转换成int类型的321？**

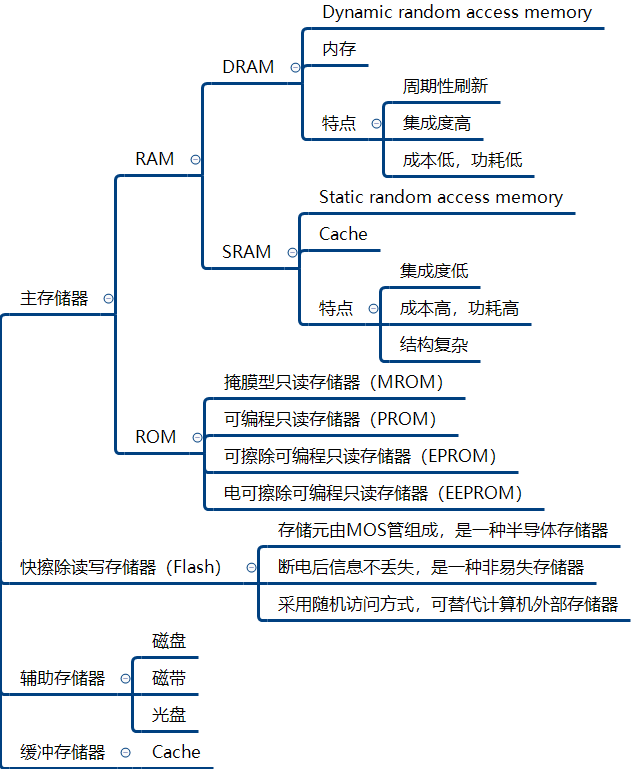
“那就用ASCII码，从头开始，每个字符减去‘0’然后赋给整型……”

1. **什么是操作系统里的‘锁’？”**

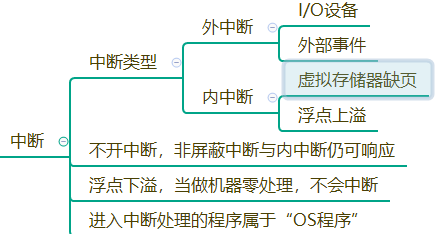
 为了保证数据的一致性，在多线程编程中我们会用到锁，使得在某一时间点，只有一个线程进入临界区代码。信号量机制就是实现进程同步与互斥的机制。通过PV操作。

37.**一些进程读了数据之后，另一些进程也读了数据，然后把修改后的数据写回去**，这个怎么办呢？

**38.“计算机存储介质有哪些？”这些介质哪些属于内存那些属于外存？**”



1. **可屏蔽中断和不可屏蔽中断区别是什么？”**



40.用c语言实现差错控制

41.cisc和risc应用不同在哪些点



RISC的设计重点在于降低由硬件执行指令的复杂度，因为软件比硬件容易提供更大的灵活性和更高的智能，因此RISC设计对编译器有更高的要求；CISC的设计则更侧重于硬件执行指令的功能，使CISC的指令变得很复杂。总之RISC对编译器的要求高，CISC强调硬件的复杂性，CPU的实现更复杂。

RISC设计思想准则：

1． 指令集----RISC处理器减少指令集的种类，通常一个周期一条指令，也就是说指令的周期是固定的，编译器或程序员通过几条指令完成一个复杂的操作；CISC的指令长度通常不固定。

2． 流水线----流水线的本质就是CPU并行运行，只是并行运行不像FPGA中的那么直接，它只是把一条指令分成几个更小的执行单元；CISC指令的执行需要调用一个微程序，明显没有RISC的指令吞吐量大。

3． 寄存器----RISC的寄存器拥有更多的通用寄存器，寄存器操作较多，例如ARM具有27个寄存器，CISC的寄存器都是用于特定目的的。

4． Load-store结构----处理器只处理寄存器中的数据，这是因为访问存储器很耗时，同时对外部存储器的读写会影响其寿命；CISC能够在存储器中直接运行

5． 寻址方式简化，不像CISC那样的复杂众多的寻址方式

42.我们知道有中间语言，那为什么要有中间语言，作用是什么

[编程语言](https://www.baidu.com/s?wd=%E7%BC%96%E7%A8%8B%E8%AF%AD%E8%A8%80&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)，要解决的问题有两个，一个是方便程序员”表达“，一个是让计算机”听懂“。  
围绕着这两个问题，[编程语言](https://www.baidu.com/s?wd=%E7%BC%96%E7%A8%8B%E8%AF%AD%E8%A8%80&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)大约经历了[机器语言](https://www.baidu.com/s?wd=%E6%9C%BA%E5%99%A8%E8%AF%AD%E8%A8%80&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)、[汇编语言](https://www.baidu.com/s?wd=%E6%B1%87%E7%BC%96%E8%AF%AD%E8%A8%80&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)、高级语言（C语言）、自然语言（比如SQL查询语言）。

43.在一个老师学生的关系中，学生学号与姓名一一对应，问采用什么数据结构能使通过学号查找姓名的速度最快？

我：链表吧，可以用一个结构体指针，将学生学号与姓名联系起来。。老师：链表可以，但不够快，用什么比较快

**Hash**

44堆和栈的区别

一个由c/C++编译的程序占用的内存分为以下几个部分  
1、栈区（stack）— 由编译器自动分配释放 ，存放函数的参数值，局部变量的值等。其操作方式类似于数据结构中的栈。  
2、堆区（heap） — 一般由程序员分配释放， 若程序员不释放，程序结束时可能由OS回收 。注意它与数据结构中的堆是两回事，分配方式倒是类似于链表，呵呵。  
3、全局区（静态区）（static）—，全局变量和静态变量的存储是放在一块的，初始化的全局变量和静态变量在一块区域， 未初始化的全局变量和未初始化的静态变量在相邻的另一块区域。 - 程序结束后有系统释放   
4、文字常量区—常量字符串就是放在这里的。 程序结束后由系统释放  
5、程序代码区—存放函数体的二进制代码。

45如何设计操作系统

46几亿个数据如何做聚类

47什么是UML

UML是[统一建模语言](https://www.baidu.com/s?wd=%E7%BB%9F%E4%B8%80%E5%BB%BA%E6%A8%A1%E8%AF%AD%E8%A8%80&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)（UML是 Unified Modeling Language的缩写）是用来对软件密集系统进行可视化建模的一种语言。UML为[面向对象](https://www.baidu.com/s?wd=%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)开发系统的产品进行说明、可视化、和编制文档的一种标准语言。  
[统一建模语言](https://www.baidu.com/s?wd=%E7%BB%9F%E4%B8%80%E5%BB%BA%E6%A8%A1%E8%AF%AD%E8%A8%80&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank) (UML)是非专利的第三代建模和规约语言。 UML是在开发阶段，说明，可视化，构建和书写一个[面向对象](https://www.baidu.com/s?wd=%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)软件密集系统的制品的开放方法。UML展现了一系列最佳工程实践，这些最佳实践在对大规模，复杂系统进行建模方面,特别是在软件架构层次已经被验证有效。  
UML可以贯穿软件开发周期中的每一个阶段。被OMG采纳作为业界的标准。  
UML最适于[数据建模](https://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E5%BB%BA%E6%A8%A1&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)，业务建模，对象建模，组件建模。  
UML作为一种模型语言，它使开发人员专注于建立产品的模型和结构，而不是选用什么程序语言和算法实现。当[模型建立](https://www.baidu.com/s?wd=%E6%A8%A1%E5%9E%8B%E5%BB%BA%E7%AB%8B&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)之后，模型可以被UML工具转化成指定的程序语言代码。

48拥塞控制    流量控制

在某段时间，若对网络中的某一资源的需求超过了该资源所能提供的可用部分，网络的性能就要变化，这种情况叫做拥塞。网络拥塞往往是由许多因素引起的，简单的提高节点处理机的速度或者扩大结点缓存的存储空间并不能解决拥塞问题。拥塞问题的是指往往是整个系统的各个部分不匹配，只有各个部分平衡了，问题才会得到解决。

开环控制和闭环控制两种方法

慢开始（Slow-start)、拥塞避免（Congestion Avoidance)、快重传（Fast Restrangsmit)和快回复（Fast Recovery）

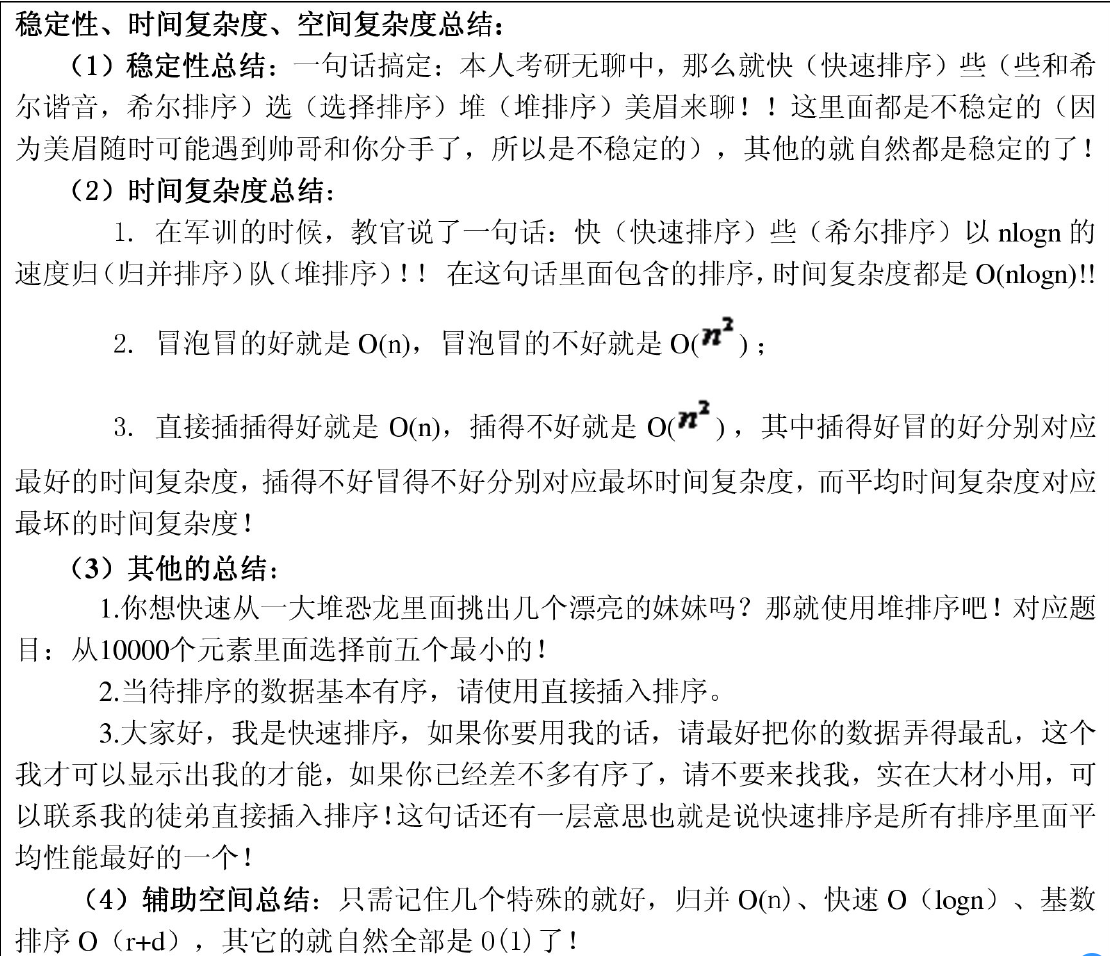
所谓拥塞控制就是防止过多的数据注入到网络中，这样可以使网络中的路由器或链路不致过载。拥塞控制所要做的都有一个前提，就是网络能承受现有的网络负荷。拥塞问题是一个全局性的问题,涉及到所有的主机、所有的路由器、以及与降低网络传输性能有关的所有因素。流量控制往往指的是点对点通信量的控制，是个端到端的问题。流量控制所要做的就是控制发送端发送数据的速率，以便使接收端来得及接受。

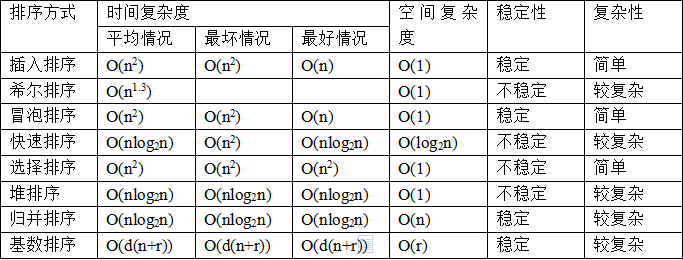
49慢启动为啥是慢启动

 慢开始算法的思路就是：最初的TCP在连接建立成功后会向网络中发送大量的数据包，这样很容易导致网络中路由器缓存空间耗尽，从而发生拥塞。因此新建立的连接不能够一开始就大量发送数据包，而只能根据网络情况逐步增加每次发送的数据量，以避免上述现象的发生。具体来说，当新建连接时，cwnd初始化为1个最大报文段(MSS)大小，发送端开始按照拥塞窗口大小发送数据，每当有一个报文段被确认，cwnd就增加至多1个MSS大小。用这样的方法来逐步增大拥塞窗口CWND。

50几种排序时间复杂度，又问了树和图，最后问了快排为啥是nlogn，什么原理，运用了什么思想







51分页式和分段式存储的区别

页和分段系统有许多相似之处，但在概念上两者完全不同，主要表现在：

1、页是信息的物理单位，分页是为实现离散分配方式，以消减内存的外零头，提高内存的利用率；或者说，分页仅仅是由于系统管理的需要，而不是用户的需要。

段是信息的逻辑单位，它含有一组其意义相对完整的信息。分段的目的是为了能更好的满足用户的需要。

2、页的大小固定且由系统确定，把逻辑地址划分为页号和页内地址两部分，是由机器硬件实现的，因而一个系统只能有一种大小的页面。

段的长度却不固定，决定于用户所编写的程序，通常由编辑程序在对源程序进行编辑时，根据信息的性质来划分。

3、分页的作业地址空间是维一的，即单一的线性空间，程序员只须利用一个记忆符，即可表示一地址。

分段的作业地址空间是二维的，程序员在标识一个地址时，既需给出段名，又需给出段内地址。

1. 地址能左移四位再加上偏移地址得到物理地址

我就说因为物理地址20位啊bulubulu，然而老师好像是要我回答因为左移后右边四位是0

53行优先和列优先有什么区别

54你这说的种区别是定义上的区别，我要你从效率上解释，比如如果A和B相乘，你选择行优先还是列优先，为什么？”。

我说：“我选择行优先，因为效率高，不用跳着存储数据”

55操作系统中进程死锁怎么形成的？你举出个例子。”

**56static变量，auto变量，extern变量，register变量**

1.extern   外部变量声明，是指这是一个已在别的地方定义过的对象，这里只是对变量的一次重复引用，不会产生新的变量。

2.static   静态数据，数据存放在全局数据区，但作用域只是本 文件/函数 中，所以你可以在两个不同的文件/函数内部申明同名的static变量，但是它们是两个不同的全局变量。 如果是定义在函数内，那么该对象具有无链接，函数外不能对其访问。如果是定义在函数外，那么该对象具有内部链接，其它程序文件不能对其访问

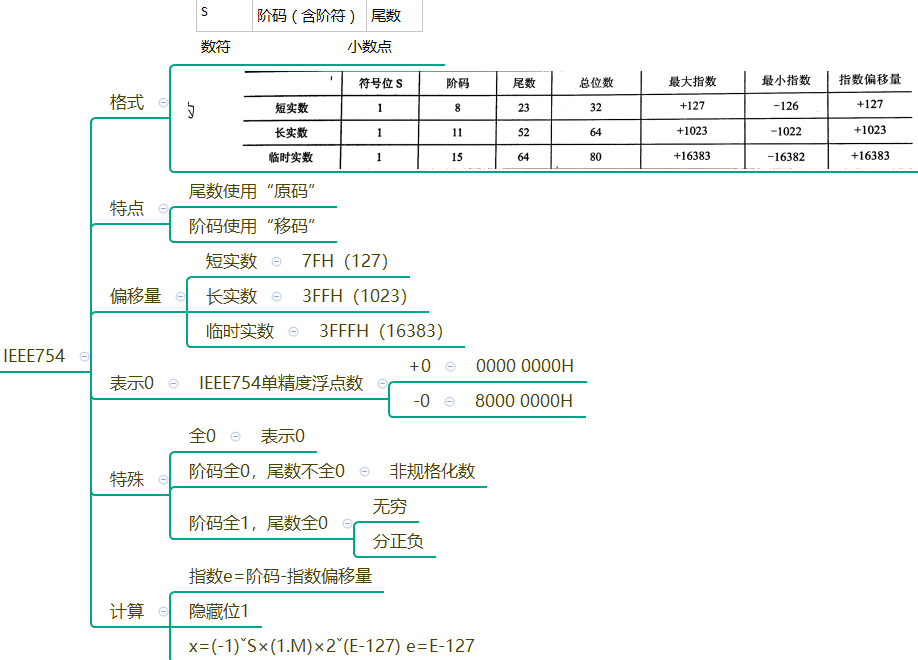
3.auto     普通局部栈变量，是自动存储，这种对象会自动创建和销毁 ，建议这个变量要放在堆栈上面，调用函数时分配内存，函数结束时释放内存。一般隐藏auto默认为自动存储类别。我们程序都变量大多是自动变量。

4.register  寄存器变量，请求编译器将这个变量保存在CPU的寄存器中，从而加快程序的运行.系统的寄存器是有限制的,声明变量时如:register int i.这种存储类型可以用于频繁使用的变量

57“折半查找用数组存数据，如果对数据进行删除和插入，就很麻烦，需要移动位置，那你设计一种数据结构解决这种问题。”    二叉排序树

58一个数字的字符串转成int

59一个高精度的数在计算机中是怎么存储的



60操作系统中都有什么调度算法     进程调度算法    页面调度算法

61存储器的分类

 62**矩阵的相加不是两个矩阵都要遍历一遍元素吗，我们可以把纵向遍历用变量i表示，横向遍历用变量j表示，现在我先写外层循环，用变量i做循环，在写内层循环，用变量j做循环，这样可以矩阵都遍历一遍。那如果我外层写j做循环，内层写i做循环，就是外层横向遍历，内层纵向遍历，有啥区别呢？**

       我：（被问到真的一脸懵逼）因为矩阵用二维数组存储嘛，二维数组又是数组的数组，前者是先找到二维数组的第一个数组，然后就只要考虑这个数组内的问题，是一维问题，取数据只要取一次地址，后者要整体考虑，是二维问题，每个数据都要取两次地址，速度会慢一些。（我真的瞎答的，在面试室里太紧张，题目都听得稀里糊涂的）

63**如果安徽这时有考试，考生有50万人，考试成绩满分是100分，最低分是0分，你怎么给他们排序？**

       我：（群里之前讨论过外部排序，我一开始以为要问这个）50万的数据内存装不下吧。哦不对，也装的下，直接用内部排序就好了。

  64.**50万也不是什么大数据，你不要考虑这个，现在给你一个足够好的电脑，你怎么排序？**

       我：我本科做过实验，这样大的数据量使用归并排序效率很高，我会选择归并排序。

65.**你不要光讲数据结构上讲的排序算法，我要听你自己怎么考虑这个问题的。**

你现在就面临这个实际问题，你该怎么处理？

       我：（wtf？你的意思是让我现场设计排序算法？我没理解错吧）（思考半分钟）我要不这样，这个问题困难在数据量很大，如果我们把他分成小段降低问题的规模会容易很多，我们把0-10分的人分一段，10-20分的人分一段，这边的人肯定不多，然后到了50分左右人肯定多了，50-55一段，55-60一段这样子，差不多均分成好几段，然后排序再整合。（此时B老师对我点了点头，我感觉老师就想知道你面对问题时是怎样思考的，现在想想还可以说分段与排序并行执行，分段程序把对应的数据扔进对应的段里，段内排序使用插入排序，因为插入排序可以来一个数据就插入有序部分里面，不用整体数据都在，还支持缓冲队列）

       A：你说的分段还可以再优化一些，你都知道这个学生的分数了，干嘛还把他分进对应的段里呢，像60分肯定比59高啊，你直接把他放进60分的区域里就好了（大体这么说的吧，不太记得了）。

       我：哦，对了，这是**桶排序**，这里用桶排序很适合。

66**那我再问你，如果我现在要做一个程序，程序里有个核心语句要反复执行，例如多项式相加，怎么做优化？**

       我：（其实这个问题很简单，可能当时太紧张连问题都没听清，也有可能是上一个问题太难落差太大，我还卡壳了一会儿，之后才反应过来）把这个核心程序编成可以调用的函数，需要用的时候调用一下。

       B：嗯，那这样做有啥优点呢？

       我：程序以后可能还要修改的，如果以后这块核心代码要修改而不编成函数，就要修改很多地方，还容易改错哪里都发现不了，编成函数只要修改一次就好了。而且这样代码也更简洁明了，可读性高。

       B：嗯，那缺点呢？

       我：调用函数肯定要用到系统栈，进这个函数内部需要进下一层栈，对硬件提出了要求。

       B：其他呢？

       我：（其他我实在不知道了）是不是不太容易排错，哦不对，这样更容易排错才对（BC两个老师都笑了，之后B老师又问了个问题，我都没听懂啥意思，C老师总结了一下）。

67**又要代码精简，又要运行效率提高，分别在C语言和C++语言里面，你如何做优化？对于参加过算法比赛的学生来说这应该不难。**

     我不想这辈子只做这方面的工作，我想学更深一点的知识，起码要和数学相关的。

**68maven具体是用来做什么的**，我说了版本管理本地仓库架包依赖等等

Maven是一个构建工具,服务与构建.使用Maven配置好项目后,输入简单的命令,如:mvn clean install,Maven会帮我们处理那些繁琐的任务.  
Maven是跨平台的.  
Maven最大化的消除了构建的重复.  
Maven可以帮助我们标准化构建过程.所有的项目都是简单一致的,简化了学习成本.  
总之,Maven作为一个构建工具,不仅帮我们自动化构建,还能抽象构建过程,提供构建任务实现.他跨平台,对外提供一致的操作接口,这一切足以使他成为优秀的,流行的构建工具.  
但是Maven不仅是构建工具,他还是一个依赖管理工具和项目信息管理工具.他还提供了中央仓库,能帮我们自动下载构件.  
使用Maven还能享受一个额外的好处，即Maven对于项目目录结构、测试用例命名方式等内容都有既定的规则，只要遵循了这些成熟的规则，用户在项目间切换的时候就免去了额外的学习成本，可以说是约定优于配置（Convention Over Configuration）。

**69又问了springMVC**，MVC是什么意思我就说了Model模型层，Viewer视图层，Controller控制层，又讲了讲具体的流程，

MVC的全名是Model View Controller，是模型(model)－视图(view)－控制器(controller)的缩写，是一种软件设计典范。它是用一种业务逻辑、数据与界面显示分离的方法来组织代码，将众多的业务逻辑聚集到一个部件里面，在需要改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑，达到减少编码的时间。

V即View视图是指用户看到并与之交互的界面。比如由html元素组成的网页界面，或者软件的客户端界面。MVC的好处之一在于它能为应用程序处理很多不同的视图。在视图中其实没有真正的处理发生，它只是作为一种输出数据并允许用户操纵的方式。

M即model模型是指模型表示业务规则。在MVC的三个部件中，模型拥有最多的处理任务。被模型返回的数据是中立的，模型与数据格式无关，这样一个模型能为多个视图提供数据，由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。

C即controller控制器是指控制器接受用户的输入并调用模型和视图去完成用户的需求，控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后再确定用哪个视图来显示返回的数据。

**最典型的MVC就是jsp+servlet+javabean模式**。

JavaBean作为模型，既可以作为数据模型来封装业务数据，又可以作为业务逻辑模型来包含应用的业务操作。其中，数据模型用来存储或传递业务数据，而业务逻辑模型接收到控制器传过来的模型更新请求后，执行特定的业务逻辑处理，然后返回相应的执行结果。

JSP作为表现层，负责提供页面为用户展示数据，提供相应的表单（Form）来用于用户的请求，并在适当的时候（点击按钮）向控制器发出请求来请求模型进行更新。

Serlvet作为控制器，用来接收用户提交的请求，然后获取请求中的数据，将之转换为业务模型需要的数据模型，然后调用业务模型相应的业务方法进行更新，同时根据业务执行结果来选择要返回的视图。

[MVC的优点：](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "https://www.cnblogs.com/diyunfei/p/_blank)

[1.](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "https://www.cnblogs.com/diyunfei/p/_blank)[耦合性](http://baike.baidu.com/item/%E8%80%A6%E5%90%88%E6%80%A7" \t "https://www.cnblogs.com/diyunfei/p/_blank)**低**

视图层和业务层分离，这样就允许更改视图层代码而不用重新编译模型和控制器代码，同样，一个应用的业务流程或者业务规则的改变只需要改动MVC的模型层即可。因为模型与控制器和视图相分离，所以很容易改变应用程序的数据层和业务规则。

**2.重用性高**

[MVC模式](http://baike.baidu.com/item/MVC%E6%A8%A1%E5%BC%8F" \t "https://www.cnblogs.com/diyunfei/p/_blank)允许使用各种不同样式的视图来访问同一个服务器端的代码，因为多个视图能共享一个模型，它包括任何WEB（HTTP）浏览器或者无线浏览器（wap），比如，用户可以通过电脑也可通过手机来订购某样产品，虽然订购的方式不一样，但处理订购产品的方式是一样的。由于模型返回的数据没有进行格式化，所以同样的构件能被不同的界面使用。

**3.部署快，**[生命周期](http://baike.baidu.com/item/%E7%94%9F%E5%91%BD%E5%91%A8%E6%9C%9F" \t "https://www.cnblogs.com/diyunfei/p/_blank)**成本低**

MVC使开发和维护用户[接口](http://baike.baidu.com/item/%E6%8E%A5%E5%8F%A3" \t "https://www.cnblogs.com/diyunfei/p/_blank)的技术含量降低。使用MVC模式使开发时间得到相当大的缩减，它使程序员（Java开发人员）集中精力于业务逻辑，界面程序员（HTML和JSP开发人员）集中精力于表现形式上。

**4.可维护性高**

分离视图层和业务逻辑层也使得WEB应用更易于维护和修改。

**MVC的缺点：**

**1.完全理解MVC比较复杂。**

由于MVC模式提出的时间不长，加上同学们的实践经验不足，所以完全理解并掌握MVC不是一个很容易的过程。

**2.调试困难。**

因为模型和视图要严格的分离，这样也给调试应用程序带来了一定的困难，每个构件在使用之前都需要经过彻底的测试。

**3.不适合小型，中等规模的应用程序**

在一个中小型的应用程序中，强制性的使用MVC进行开发，往往会花费大量时间，并且不能体现MVC的优势，同时会使开发变得繁琐。

**4.增加系统结构和实现的复杂性**

对于简单的界面，严格遵循MVC，使模型、视图与控制器分离，会增加结构的复杂性，并可能产生过多的更新操作，降低运行效率。

**5.视图与控制器间的过于紧密的连接并且降低了视图对模型数据的访问**

视图与控制器是相互分离，但却是联系紧密的部件，视图没有控制器的存在，其应用是很有限的，反之亦然，这样就妨碍了他们的独立重用。

依据模型操作接口的不同，视图可能需要多次调用才能获得足够的显示数据。对未变化数据的不必要的频繁访问，也将损害操作性能。

**70然后又问了我为什么要分层**，我一时不知道怎么回答就说，总不能在页面里面写大段的java代码吧，分层便于功能扩充，模块之间相互独立这些

主要还是为了方便项目管理，增加代码的复用性，更好的开发与维护的作用

DAO —— Data Access Object数据访问对象（接口）

DAOImpl —— DAO的实现类

entity —— 数据对象的实体（有些地方叫model层）

Service（不是Server）——就是中间层、业务逻辑层（接口）

ServiceImpl —— Service的实现类

Util —— 自定义工具类

Servlet——JAVA WEB小应用（有时叫Controller层）

71贪心算法，原理是什么

使用贪心算法需要具备两个性质：贪心选择性和最优子结构。

贪心选择性是指所求问题的整体最优解可以通过一系列可以通过一系列局部最优的选择来达到。他总是作出当前最好的选择，该选择可以依赖于之前的选择，但绝不依赖于将来的选择和子问题的选择，这是他与动态规划的重要区别。

最优子结构是指一个问题的最优解包含其子结构的最优解。

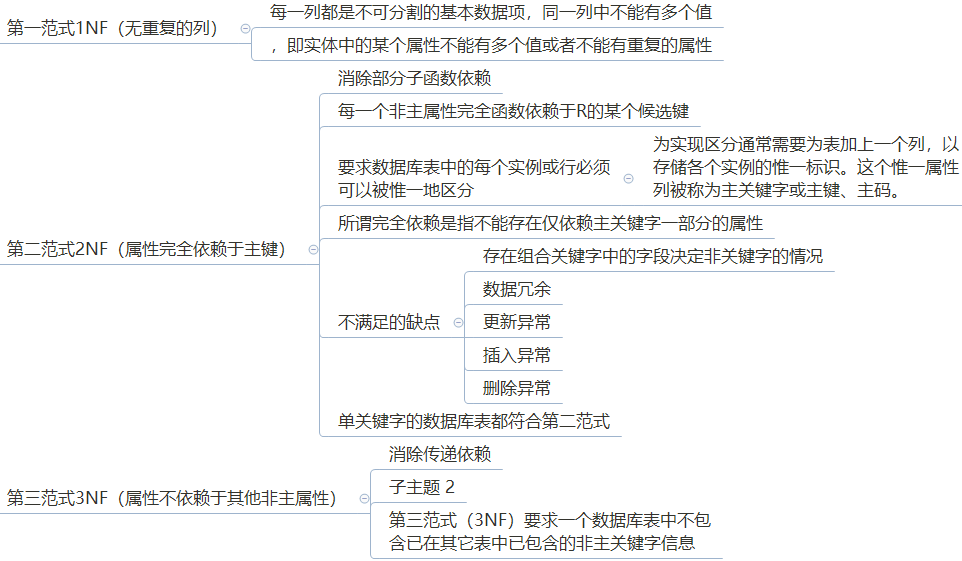
在证明这两个性质的时候，比较好证的是证明最优子结构。

最优子结构一般通过反证法证明，即，如果该问题的最优解不是子问题的最优解，那么假设子问题有一个最优解，然后由该最优解构造出原问题的一个最优解，产生了矛盾。需要注意的是，子问题是指作出贪心选择之后的子问题。

贪心选择性的证明一般先假设一个问题的最优解，据此构造另一个最优解，使其第一步是贪心选择。

数据库：

1. 第一范式的优点是什么，缺点是什么？



73.如果有科生，硕士生，博士生，你将会怎样建表？

JAVA:

1. JSP和JSF的区别是什么?

JSF是UI组件状态，处理事件等的标准化方式。JSF是一种技术，同时笔者认为更是一种框架。与许多的Web框架一样，都会有自已的config.xml即faces-config.xml。而JSP则是表现层，借助于JSP即可表现Web应用的界面效果，而JSF则是表现层框架，与此类似Struts也是表现层框架，虽然性质不同但也殊途同归。

JSF是一个标准的服务器端架构，用于简化网络应用软件表示层的构建。开发人员可以组装这些可复用的用户界面组件来创建网页，将这些组件绑定到应用

程序的数据源，并利用服务器端的事件处理器来处理客户端事件。依据JSF规范，组件开发者所制作的组件可以简洁地整合到JSF运行时架构，而且在应用软件

设计过程中，还可以整合到与JSF兼容的集成开发环境中。JSR

127规范定义了这样的JSF架构，还同时提供了比如输入栏和按钮等基本用户界面组件的可供参考的实现。JSF组件中的绝大部分都符合HTML

2.0标准中的HTML组件和标签的规范。这些相对简单的组件对于很多网络应用程序来讲已经足够了。

JSF框架的主要优势之一就是它既是   Java Web 用户界面标准又是严格遵循模型-视图－控制器  (MVC)设计模式的框架。用户界面代码（视图）与应用程序数据和逻辑（模型）的清晰分离使  JSF应用程序更易于管理。为了准备提供页面对应用程序数据访问的  JSF上下文和防止对页面未授权或不正确的访问，所有与应用程序的用户交互均由一个前端“Faces”servlet（控制器）来处理。

1. 面向对象相关的问题。

1.面向对象的特征： 封装、继承、多态（抽象）。

2.什么是面向对象编程：简称OOP，应用程序是许多对象在计算机中相继表现自己，而对象就是一个个程序实体，每个对象都是独立但又能相互作用，当程序功能需要更改时，只需要修改相应的对象就行了。

3.什么是类：类是面向对象编程的基本概念，是具有相同性质和功能的事物的集合，也是对一组具有共性的对象的一种抽象描述。

4.类和对象的区别：类是抽象的概念，用于描述和创建对象；对象是类的实例，是可以单独存在的实体。

5.什么是抽象类：抽象类主要用来提供多个派生类可共享的基类的公共定义。

设置类成员访问权限：在java中可以在声明类、成员变量和方法时，使用private、prtected、public和默认不加（default)对其进行修饰，以限制类及其成员的访问权限。

6.什么是构造方法：构造方法是在创建指定类型的对象时执行的方法，名称与类相同，且无返回值，通常用于创建类的实例以及必要的初始化操作。

7.什么是方法重载：方法重载是在一个类或接口中可以有多个相同名称的方法，但是却具有数量不同或者类型不同的形参。

8.什么是接口：接口是对类的抽象描述，只有方法的特征而无方法的实现，因为这的方法可以在不同的地方被不同的类实现，而这些类可以具有不同的行为。

9.变量的作用域：局部变量在方法中声明，并且无法被其它方法所使用，方法开始时被创建，结束后被销毁；全局变量在类中作为成员变量声明，并为其它方法所共享，在实例化时被创建（不包括静态变量）。

10.类的成员变量和方法的命名规范：首字母小写，如果有两个或两个以上单词拼接，从第二个单词开始，每个单词首字母大写。

11.什么是java应用程序的主类：java应用程序一般由若干类和接口组成，若想运行java程序，至少有个类含有main方法，这个方法就是java应用程序的执行的入口，所以我们把含有main方法的类成为java应用程序的主类。

12.什么是静态变量和静态方法：如果希望某个变量的值和某个功能可以别所有对象共享而不需要对拥有这个变量的类进行实例化，可以用static关键字修饰，这就是静态变量和静态方法。

13.java中的常量：在java中使用final关键字来声明常量，如果加上static关键字修饰，那么它就是静态常量，它的值在确定后不允许被修改。

14.什么是静态语句块：在java中可以在static关键字后跟上一对大括号来申明静态语句块，它在类被加载后就被分配内存，并执行静态语句块中的代码，通常用于必要的初始化操作。

15.如何访问类的成员变量和方法：通过使用类实例化后的对象名跟上“.”运算符再跟上要访问的成员变量名或方法名来访问（如果方法有形参，则必须传递相同类型和数量的实参，且顺序一致）。

16.如何在一个类中访问另一个类的方法：通过在一个类中定义另一个类的实例，再通过该实例访问那个类的方法。

17.面向对象的6大原则：开闭原则(OCP)，李氏替换原则(LSP)，依赖注入原则(DIP)，接口隔离(ISP)，单一职责(SRP)，迪米特法则(LKP)。

微机原理：

76.说一下存储器的分类。

77给你一个数组找出其中最大的值，怎么找

78给你一个线性方程组怎么解

79什么是多态

80把一些具有相同性质的类型，根据相同性质抽象出一个基类，然后具体用到的类型都继承自这个基类，在实现函数的时候，参数使用基类的指针或者引用，这样就可以用不同类型的对象调用函数。成员函数定义为虚函数，每个类会有一个虚表，该类的对象包含虚指针，运行时会通过虚指针找到虚表从而调用对应的成员函数

老师那就是用不同的对象调用同一个函数得到不同的结果？

81给你一堆无序的数据，怎么把他组织起来，可以让我们在查找的时候迅速的确定某个值在不在这堆数据里面

用hash表

A：hash表怎么弄

我：就是把所有的数据组成一个hash表，查找的时候，计算出来要查找的数据的hash值，然后看表中对应的位置有没有，没有就是没有...

A：hash表的查找时间复杂度是多少

我： O（1）

82kmp算法

83面向对象的优点

## **面向过程与面向对象编程的区别**

面向过程就是分析出解决问题所需要的步骤，然后用函数把这些步骤一步一步实现，使用的时候一个一个依次调用就可以了；

面向对象是把构成问题事务分解成各个对象，建立对象的目的不是为了完成一个步骤，而是为了描叙某个事物在整个解决问题的步骤中的行为。

可以拿生活中的实例来理解面向过程与面向对象，例如五子棋，面向过程的设计思路就是首先分析问题的步骤：1、开始游戏，2、黑子先走，3、绘制画面，4、判断输赢，5、轮到白子，6、绘制画面，7、判断输赢，8、返回步骤2，9、输出最后结果。把上面每个步骤用不同的方法来实现。

如果是面向对象的设计思想来解决问题。面向对象的设计则是从另外的思路来解决问题。整个五子棋可以分为1、黑白双方，这两方的行为是一模一样的，2、棋盘系统，负责绘制画面，3、规则系统，负责判定诸如犯规、输赢等。第一类对象（玩家对象）负责接受用户输入，并告知第二类对象（棋盘对象）棋子布局的变化，棋盘对象接收到了棋子的变化就要负责在屏幕上面显示出这种变化，同时利用第三类对象（规则系统）来对棋局进行判定。

可以明显地看出，面向对象是以功能来划分问题，而不是步骤。同样是绘制棋局，这样的行为在面向过程的设计中分散在了多个步骤中，很可能出现不同的绘制版本，因为通常设计人员会考虑到实际情况进行各种各样的简化。而面向对象的设计中，绘图只可能在棋盘对象中出现，从而保证了绘图的统一。

## **■面向过程与面向对象的优缺点**

**面向过程**  
　　优点：性能比面向对象高，因为类调用时需要实例化，开销比较大，比较消耗资源，比如单片机、嵌入式开发、Linux/Unix等一般采用面向过程开发，性能是最重要的因素。   
　　缺点：没有面向对象易维护、易复用、易扩展

**面向对象**  
　　优点：易维护、易复用、易扩展，由于面向对象有封装、继承、多态性的特性，可以设计出低耦合的系统，使系统更加灵活、更加易于维护   
　　缺点：性能比面向过程低

84什么是文件系统？文件系统干吗用的？ 我说用来 管理文件、磁盘等，老师说那肯定的啊，我问你干嘛用的！  
文件是具有文件名的一组相关信息的集合。由若干个记录组成。文件系统是操作系统中与文件管理有关的软件和数据的集合。

功能

1实现按文件名存取文件信息

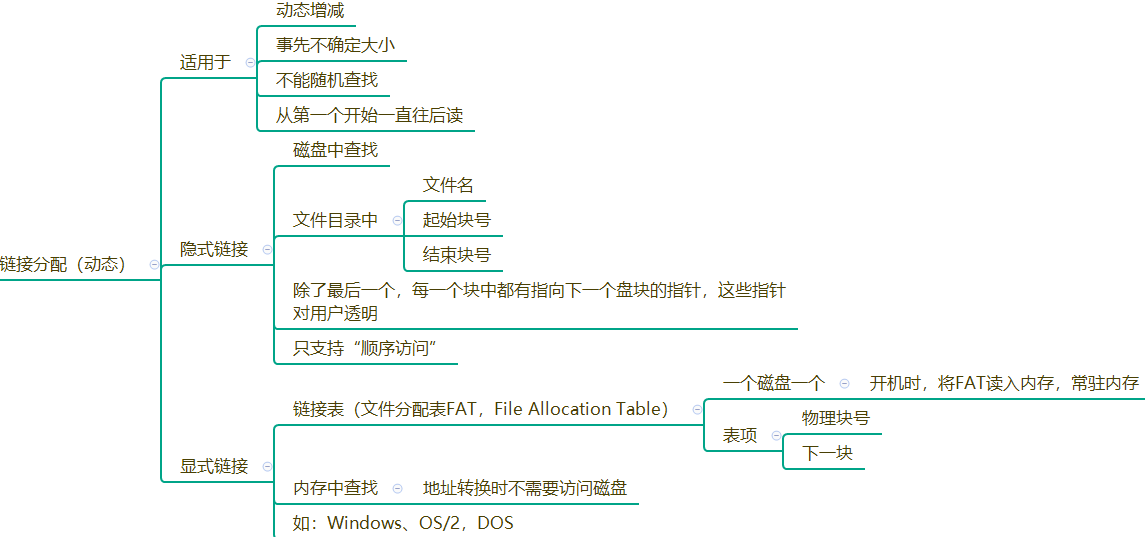
2为用户提供统一友好的接口

3实施对文件和文件目录的管理

4文件存储器空间的分配与回收

5提供有关文件的共享和保护。

85为什么要有FAT？



**86简述一下 csma**

87最后问了一个临界区的问题

**成绩单，就叫我从个人情况，科研能力，实践等几个方面介绍自己，**

88第一个**问我什么是满二叉树？**

除最后一层无任何子节点外，每一层上的所有结点都有两个子结点二叉树。

89快速排序      后又问我在有序的情况下进行快排时间复杂度，是0还是多少，我说不是0，我说这是快排一个缺点，说了时间复杂度应该是o(n平方)

90RAID

RAID 0（不含校验与冗余的条带存储），功能：多块磁盘组合为RAID0后，数据被分割病分别存储在每块硬盘中，所以能最大地提升存储性能与存储空间，但无法容错，RAID 0至少需要两块磁盘。缺点：因为没有校验与备份，两块硬盘中如果有一块磁盘损坏，即磁盘中的任何一个数据块损坏将导致整个文件无法读取。

RAID 1（不含校验的镜像存储），功能：多块磁盘组合为RAID 1后，数据将被同时复制到每块硬盘中，制作这种磁盘阵列至少需要两块硬盘，该级别的RAID只需要一块磁盘可用，即可正常工作，该级别的阵列安全性是最好的，磁盘的利用率也是最低的。缺点：写入的效率低，且磁盘的利用率极低。

RAID 2（位级别的校验式条带存储），功能：多块磁盘组合为RAID2后，数据将以位（bit）位单位同步式分别存储在不同的硬盘上，并采用海明码（一种具有纠错功能的校验码）对数据进行校验和恢复，对大数据量的读写具有很高的性能，但对少量数据的读写性能反而不好，该磁盘阵列至少需要三块硬盘。

RAID 3（字节级别的校验式条带存储），功能：参考RAID2，数据分割级别为字节。

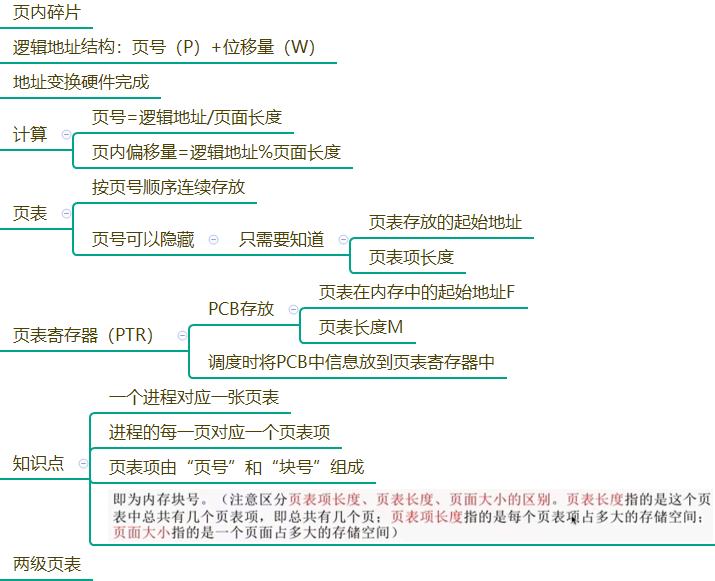
RAID 4（数据块级别的校验式条带存储），功能：该级别的磁盘阵列与RAID2类似，仅数据分割单位不同。

RAID 5（数据块级别的分布式校验条带存储），功能：多块磁盘组合为RAID5后，数据将以块（block）为单位同步式分别存储在不同的硬盘上，并对数据进行海明码运算，与其他级别不同的式，该级别的海明码会被写入不同的磁盘。

RAID 10（镜像与条带存储），功能：它由RAID0余RAID1结合而成，RAID10继承了RAID0的快速与高效，同时也继承了RAID1的数据安全，RAID10至少需要四块硬盘。RAID10通常有两种结构，一种是RAID0+1结构，一种是RAID1+0结构。

91抽到页表

用固定大小的**页(Page)**来描述[逻辑地址空间](https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E5%9C%B0%E5%9D%80%E7%A9%BA%E9%97%B4" \t "https://baike.baidu.com/item/%E9%A1%B5%E8%A1%A8/_blank)，用相同大小的**[页框](https://baike.baidu.com/item/%E9%A1%B5%E6%A1%86" \t "https://baike.baidu.com/item/%E9%A1%B5%E8%A1%A8/_blank)(Frame)**来描述[物理内存](https://baike.baidu.com/item/%E7%89%A9%E7%90%86%E5%86%85%E5%AD%98" \t "https://baike.baidu.com/item/%E9%A1%B5%E8%A1%A8/_blank)空间，由操作系统实现从逻辑页到物理页框的**页面映射**，同时负责对所有页的管理和进程运行的控制。



92栈和队列

1. 冯诺依曼结构中分支指令转移的条件（我只记得条件转移、无条件转移、程序调用，然后就被怼了）

转移指令，相对于流水指令而说的。转移指令是指不按程序的语句流程执行的指令，如跳转指令，call指令、jmp指令等。

转移指令分为以下几类：

无条件转移指令（如：jmp）

（1）绝对跳转指令

（2）相对跳转指令

（3）长跳转指令

（4）间接跳转指令

条件转移指令（如：jcxz）

功能：以标志位的状态或者以标志位的逻辑运算结果作为转移依据，如果满足转移条件，则转到目标地址所指示的指令执行否则继续执行下一条指令。必须指出，条件转移指令转移地址的偏移量限制在-128～+127字节范围内，采用相对转移方式（相对转移指令是指跳转时以当前地址为基准加上相对偏移量进行跳转，一般是在本地址段内跳转。如果需要跳转到较远的地方如跳到另一个地址段，就需要加上跳转的目标段地址）

循环指令（如loop）

可以控制程序的循环.

过程

例如子程序调用和返回指令.

中断

使计算机暂时挂起正在执行的进程而转去处理某事件.处理完后再恢复执行原进程的过程.

94.关系是什么（这个没答出来）

95.一些本科做过的项目的情况（软件工程中你用过的需求分析的方法，你的项目使用的开发模式，逻辑设计的作用和逻辑分析生成的结果）

96满树中有n层 那总共有几个节点

97什么是TLB

98除了时分复用还有什么

**99.时分双工与时分复用有什么区别？**

**100.描述TCP/IP参考模型各层的作用。**

**101.阐述ROM/RAM/SRAM/DRAM的区别与联系**

 102解释一下广播风暴

   103，说一下你对数据结构的理解

   104，说一下数据结构里面的几种排序算法及其时间复杂度

   105，说一下软件工程的过程

   106，说一下你以前用过的算法

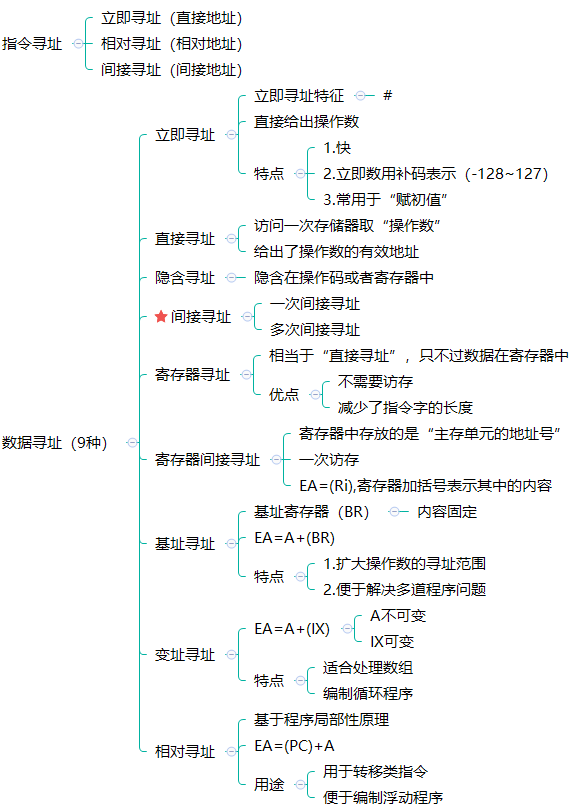
   107，说一下字符串匹配

   108，说一下tlb

109CDMA的工作原理，现在的CDMA有哪些标准，中国自主研发的是哪个标准

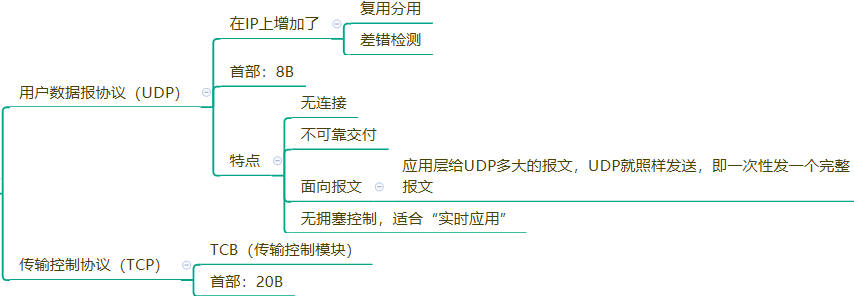
110微指令控制器

111寻址方式



112.c++的特征（封装，继承，多态）

113.tcp与udp的区别



114.osi各层有啥协议

[物理层](https://baike.baidu.com/item/%E7%89%A9%E7%90%86%E5%B1%82" \t "https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/_blank)：[以太网](https://baike.baidu.com/item/%E4%BB%A5%E5%A4%AA%E7%BD%91" \t "https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/_blank) · [调制解调器](https://baike.baidu.com/item/%E8%B0%83%E5%88%B6%E8%A7%A3%E8%B0%83%E5%99%A8" \t "https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/_blank) · 电力线通信(PLC) · [SONET/SDH](https://baike.baidu.com/item/SONET/SDH" \t "https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/_blank) · G.709 · [光导纤维](https://baike.baidu.com/item/%E5%85%89%E5%AF%BC%E7%BA%A4%E7%BB%B4" \t "https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/_blank) · 同轴电缆 · 双绞线等

**数据链路层：**Wi-Fi([IEEE 802.11](https://baike.baidu.com/item/IEEE 802.11" \t "https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/_blank)) · WiMAX([IEEE 802.16](https://baike.baidu.com/item/IEEE 802.16" \t "https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/_blank)) ·ATM · DTM · [令牌环](https://baike.baidu.com/item/%E4%BB%A4%E7%89%8C%E7%8E%AF" \t "https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/_blank) · [以太网](https://baike.baidu.com/item/%E4%BB%A5%E5%A4%AA%E7%BD%91" \t "https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/_blank) ·FDDI · [帧中继](https://baike.baidu.com/item/%E5%B8%A7%E4%B8%AD%E7%BB%A7" \t "https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/_blank) · GPRS · EVDO ·HSPA · HDLC · [PPP](https://baike.baidu.com/item/PPP/6660214" \t "https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/_blank) · L2TP ·PPTP · ISDN·STP · CSMA/CD等

[网络层](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%B1%82" \t "https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/_blank)协议：IP (IPv4 · IPv6) · ICMP· ICMPv6·IGMP ·IS-IS · IPsec · ARP · RARP · RIP等

[传输层](https://baike.baidu.com/item/%E4%BC%A0%E8%BE%93%E5%B1%82" \t "https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/_blank)协议：TCP · UDP · TLS · [DCCP](https://baike.baidu.com/item/DCCP" \t "https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/_blank) · SCTP · RSVP · OSPF 等

[应用层](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E5%B1%82" \t "https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/_blank)协议：DHCP ·[DNS](https://baike.baidu.com/item/DNS" \t "https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/_blank) · FTP · Gopher · HTTP· IMAP4 · IRC · NNTP · XMPP ·POP3 · SIP · SMTP ·SNMP · SSH ·TELNET · RPC · RTCP · RTP ·RTSP· SDP · SOAP · GTP · STUN · NTP· SSDP · BGP 等

1. 描述二分查找算法及复杂

二分查找（binary search），也称作折半查找（half-interval search），每次划分一半进行下一步搜索，所以时间复杂度无非就是while循环的次数！

//二分查找 Java 实现

public static int binarySearch(Integer[] srcArray, int des) {

int low = 0;

int high = srcArray.length - 1;

while ((low <= high) && (low <= srcArray.length - 1)

&& (high <= srcArray.length - 1)) {

int middle = (high + low) >> 1;

if (des == srcArray[middle]) {

return middle;

} else if (des < srcArray[middle]) {

high = middle - 1;

} else {

low = middle + 1;

}

}

return -1;

}

2、时间复杂度

比如：总共有n个元素，每次查找的区间大小就是n，n/2，n/4，…，n/2^k（接下来操作元素的剩余个数），其中k就是循环的次数。

由于n/2^k取整后>=1，即令n/2^k=1，

可得k=log2n,（是以2为底，n的对数），所以时间复杂度可以表示O()=O(logn)

1. 描述哲学家就餐问题

**1.设置互斥信号量**，每次至多4个哲学家进餐

**2.分奇偶数编号进餐**

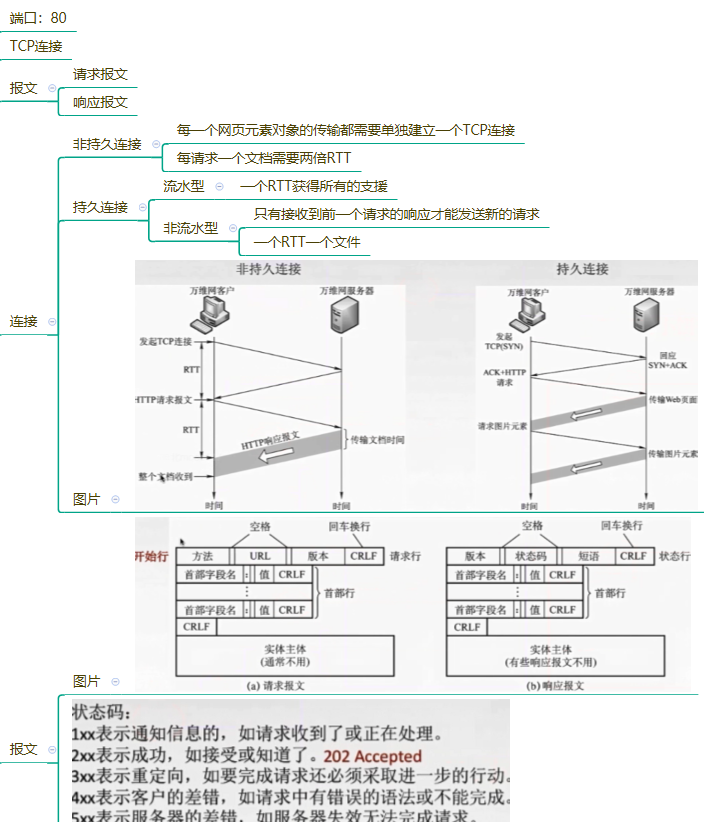
奇数号哲学家先取左手的筷子，然后取右手的筷子，而偶数哲学家先取右手的筷子，然后再取左手的筷

**3.破坏死锁（请求）条件**

**(1)破坏请求保持条件**

利用原子思想完成。即只有拿起两支筷子的哲学家才可以进餐，否则，一支筷子也不拿。

117.http持续与非持续的区别



1. 数据结构在计算机网络和操作系统中的应用
2. 状态机描述数据结构
3. c++里面的模板是什么

　模板是C++支持**参数化**多态的工具，使用模板可以使用户为类或者函数声明一种一般模式，使得类中的某些数据成员或者成员函数的参数、返回值取得任意类型。

　　模板是一种对**类型**进行**参数化**的工具；

　　通常有两种形式：**函数模板**和**类模板**；

　　函数模板针对仅**参数类型**不同的**函数**；

　　类模板针对仅**数据成员**和**成员函数类型**不同的类。

**使用模板的目的就是能够让程序员编写与类型无关的代码。**比如编写了一个交换两个整型int 类型的swap函数，这个函数就只能实现**int** 型，对**double**，字符这些类型无法实现，要实现这些类型的交换就要重新编写另一个**swap**函数。使用模板的目的就是要让这程序的实现与类型无关，比如一个**swap**模板函数，即可以实现**int** 型，又可以实现double型的交换。模板可以应用于函数和类。下面分别介绍。

**注意：模板的声明或定义只能在全局，命名空间或类范围内进行。即不能在局部范围，函数内进行，比如不能在main函数中声明或定义一个模板。**

1. java和c++的本质区别在哪

通常，我们聊到Java，第一印象“面向对象”，“没有指针，编写效率高，执行效率较低”。更深入、专业一点就谈论 “java内存自动回收(GC垃圾回收机制)，多线程编程”。\*\*   
java的三大特性是封装、继承和多态。\*\*

总结如下：

1、 JAVA的应用在高层，C++在中间件和底层

2、 JAVA离不开业务逻辑，而C++可以离开业务为JAVA们服务

3、 java语言给开发人员提供了更为简洁的语法；取消了指针带来更高的代码质量；完全面向对象，独特的运行机制是其具有天然的可移植性。

4、 java 是运行在JVM上的，之所以说它的可移植性强，是因为jvm可以安装到任何的系统

5、 c++不是不能在其他系统运行，而是c++在不同的系统上运行，需要不同的编码（这一点不如java，只编写一次代码，到处运行）。java程序一般都是生成字节码，在JVM里面运行得到结果。

6、 java 在web 应用上具有c++ 无可比拟的优势

7、 java在桌面程序上不如c++实用，C++可以直接编译成exe文件，指针是c++的优势，可以直接对内存的操作，但同时具有危险性 。（操作内存的确是一项非常危险的事情，一旦指针指向的位置发生错误，或者误删除了内存中某个地址单元存放的重要数据，后果是可想而知的）。

8、 垃圾回收机制的区别。c++用析构函数回收垃圾，java自动回收（GC算法）,写C和C++程序时一定要注意内存的申请和释放。

9、 java 丰富的插件是java 发展如此迅速的原因   
10、 java 很大的沿袭了c++的一些实用结构   
11、 对于底层程序的编程以及控制方面的编程，c++很灵活，因为有句柄的存在。

Java并不仅仅是C++语言的一个变种，它们在某些本质问题上有根本的不同：

(1)Java比C++程序可靠性更高。有人曾估计每50行C++程序中至少有一个BUG。姑且不去讨论这个数字是否夸张，但是任何一个C++程序员都不得不承认C++语言在提供强大的功能的同时也提高了程序含BUG的可能性。Java语言通过改变语言的特性大大提高了程序的可靠性。

(2)Java语言不需要程序对内存进行分配和回收。Java丢弃了C++ 中很少使用的、很难理解的、令人迷惑的那些特性，如操作符重载、多继承、自动的强制类型转换。特别地，Java语言不使用指针，并提供了自动的废料收集，在Java语言中，内存的分配和回收都是自动进行的，程序员无须考虑内存碎片的问题。

(3)Java语言中没有指针的概念，引入了真正的数组。不同于C++中利用指针实现的“伪数组”，Java引入了真正的数组，同时将容易造成麻烦的指针从语言中去掉，这将有利于防止在c++程序中常见的因为数组操作越界等指针操作而对系统数据进行非法读写带来的不安全问题。

(4)Java用接口(Interface)技术取代C++程序中的多继承性。接口与多继承有同样的功能，但是省却了多继承在实现和维护上的复杂性。

122对称加密和非对称加密？AES是对称还是非对称

123指针是什么？    指针怎么表示数组？指针表示和数组下标表示的区别是什么？

指针就是一个存放地址的变量，当指针指向某个变量，这时这个指针里就存放了那个变量的地址，同时可以利用指针直接取变量中的值用 只要在指针前加 \* 就是取其真值。

1.数组名代表了一个指向数组首元素的常量指针，一经定义，不可更改，数组名作为常量指针，其类型与数组元素类型相同。指针是变量指针，定义之后仍可更改，其类型在定义时确定。

2.当出现sizeof,和&操作符时，数组名不再当成指向一个元素的常量指针来使用，而指针仍当成指向一个元素的变量指针来使用。

对于使用指针和数组下标的选择：

系统在使用数组下标对数组成员变量进行访问时，开销比较大，指针的访问效率是远远大于数组名的访问效率的。

但是只有在指针正确访问时，才成比下标法更有效率。

下标法更加容易理解，在可读性方面，也更加的具有优势

124什么是数据结构?

数据结构：数据结构是指数据以及数据相互之间的联系，可以看成是相互之间具有某种特定关系的数据元素的集合，因此，可以把数据结构看成是带结构的数据元素的集合。

数据结构包含以下几个方面：

数据元素之间的逻辑关系，即数据的逻辑结构。

数据元素及其关系在计算机存储器中的存储方式，即数据的存储结构，也称为数据的物理结构。

施加在该数据上的操作，即数据的运算。

所以数据结构由三个部分组成：逻辑结构、物理结构、运算。

 125什么是指针？为什么要有指针？

指针是内存的标签

在实际编程中，既然使用变量就可以访问数据，为什么还用指针呢？使用指针来操纵自动变量的唯一原因是，为了说明指针的工作原理。指针真正的用途是被用于完成下列3种任务：

管理自由存储区中的数据

访问类的成员数据和函数

按引用传递参数

    在“函数的 工作原理 ”中，提到了5个内存区域：

全局名称空间

自由存储区

寄存器

代码空间

堆栈

     局部变量和函数参数位于堆栈中 ：当然，代码位于代码空间中；而全局变量位于全局名称空间中；寄存器用于内部管理工作，如记录栈顶指针和指令指针。余下的所有内存都被作为自由存储区，通常被称为堆。

     局部变量不是永久性的，函数返回时，局部变量就被删除。这很好，因为这意味着根本不用为管理这种内存空间而劳神；也不好，因为这使得函数在不将堆中的对象复制到调用函数中的目标对象的情况下，将难以创建供其他对象或函数使用的对象。全局变量解决了这种问题，其代价是整个程序中都可以访问它们。这导致创建了难以理解和维护的代码。如果管理得当，将数据存储在自由存储区可以解决这两种问题。

可以将自由存储区视为一块很大的内存，其中有数以千计的依次被编号的内存单元，可用于存储数据。与堆栈不同，你不能对这些单元进行标记，而必须先申请内存单元的地址，然后将它存储到指针中。

     可以使用这样的类比：朋友给了你交通银行的800客服电话号码。你回到家中，将该电话号码与某个按钮绑定，然后扔掉记录电话号码的纸张。如果按下按钮，被拨打的电话将响铃，交通银行的职员进行接听。你不需要记下这个电话号码，，也不知道被拨打的电话在哪里，但只要按下绑定的按钮就能致电交通银行。自由存储区中的数据就像交通银行一样，你不知道它在什么地方，但知道如何找到它。你使用地址（在这个例子中，地址为电话号码）访问它。你不必知道地址，只需将其放在一个指针（按钮）中。指针让你能够访问数据，而不必知道细节。

     函数返回时，堆栈被自动清空。所有局部变量都不在作用域内，他们被从堆栈中删除。程序结束前，自由存储器不会自动清空，程序员使用完自己分配的内存后，必须负责将其释放。

自由存储区的优点是，你从中分配的内存将一直可用，知道你明确的指出不再需要——将其释放为止。如果在函数中分配自由存储区中的内存，在函数返回后该内存仍可用。

这也是自由存储区的缺点，如果你忘记释放内存，被占据而没有使用的内存将随着时间的推移愈来愈多，导致系统崩溃。

     采用这种内存访问方式而不是全局变量的优点是，只有能够访问指针的函数才能访问它指向的数据。这样只有将包含指针的对象或指针本身传递给函数，函数才能修改指针指向的数据，从而减少了函数能够改变数据，而又无法跟踪变更的情况发生。

 126链表和数组区别

链表是一种上一个元素的引用指向下一个元素的存储结构，链表通过指针来连接元素与元素；

链表是线性表的一种，所谓的线性表包含顺序线性表和链表，顺序线性表是用数组实现的，在内存中有顺序排列，通过改变数组大小实现。而链表不是用顺序实现的，用指针实现，在内存中不连续。意思就是说，链表就是将一系列不连续的内存联系起来，将那种碎片内存进行合理的利用，解决空间的问题。

所以，链表允许插入和删除表上任意位置上的节点，但是不允许随即存取。链表有很多种不同的类型：单向链表、双向链表及循环链表。

不同：链表是链式的[存储结构](https://www.baidu.com/s?wd=%E5%AD%98%E5%82%A8%E7%BB%93%E6%9E%84&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://blog.csdn.net/m0_37631322/article/details/_blank)；数组是顺序的[存储结构](https://www.baidu.com/s?wd=%E5%AD%98%E5%82%A8%E7%BB%93%E6%9E%84&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://blog.csdn.net/m0_37631322/article/details/_blank)。

链表通过指针来连接元素与元素，数组则是把所有元素按次序依次存储。

链表的插入删除元素相对数组较为简单，不需要移动元素，且较为容易实现长度扩充，但是寻找某个元素较为困难；

数组寻找某个元素较为简单，但插入与删除比较复杂，由于最大长度需要再编程一开始时指定，故当达到最大长度时，扩充长度不如链表方便。  
相同：两种结构均可实现数据的顺序存储，构造出来的模型呈[线性结构](https://www.baidu.com/s?wd=%E7%BA%BF%E6%80%A7%E7%BB%93%E6%9E%84&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://blog.csdn.net/m0_37631322/article/details/_blank)。

 127链表怎么插入删除？

单链第1个数据删除结点的算法思路:

 1.声明一指针p指向链表头指针，初始化j 从1开始;

 2.当j<i 时，就遍历链表，让P 的指针向后移动，不断指向下一个结点，累加1;

 3.若到链表末尾p为空,说明第个结点不存在;

 4.否则查找成功，将欲删除的结点p->next 赋值给9;

 5.单链表的删除标准语句p->next=q->next;

 6.将q 结点中的数据赋值给e，作为返回;

 7.释放q 结点;

 128用C语言怎么输入一张成绩单并查询？

 129什么是结构体？

其实结构体也是一种数据类型，只不过在这种数据类型中又包含了几个基本的数据类型；

但为什么要有结构体这种数据类型呢？试想一下，假如我们要描述一个学生的基本信息，那一个学生有年龄，有性别，有爱好等等，我们要描述这些，我们的基本类型可以吗？显然是不行的，所以我们必须用一个结构体将描述学生的基本信息组合起来，构成一种我们想要的类型；但结构体变量在内存中的存放和我们的基本数据类型变量在内存中的存放是不同的，基本数据类型的存放系统是会给他们分配一块连续的空间用来存放，而结构体变量确是不一定的

130关系是什么

关系是指人与人之间，人与事物之间，事物与事物之间的相互联系。关系数据库，是建立在关系模型基础上的[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728" \t "https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93/_blank)，借助于集合代数等数学概念和方法来处理数据库中的[数据](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE/33305" \t "https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93/_blank)。现实世界中的各种实体以及实体之间的各种联系均用关系模型来表示。关系模型是由埃德加·科德于1970年首先提出的，并配合“科德十二定律”。现如今虽然对此模型有一些批评意见，但它还是数据存储的传统标准。标准数据查询语言SQL就是一种基于关系数据库的语言，这种语言执行对关系数据库中数据的检索和操作。 关系模型由关系数据结构、[关系操作](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E6%93%8D%E4%BD%9C/9797727" \t "https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93/_blank)集合、关系完整性约束三部分组成。

简单说，关系型数据库是由多张能互相联接的二维行列表格组成的数据库。

131mvc

### [MVC](https://baike.baidu.com/item/MVC" \t "https://baike.baidu.com/item/MVC%E6%A1%86%E6%9E%B6/_blank)全名是Model View Controller，是模型(model)－视图(view)－控制器(controller)的缩写，一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。MVC被独特的发展起来用于映射传统的输入、处理和输出功能在一个逻辑的图形化用户界面的结构中。

### 视图

视图是用户看到并与之交互的界面。对老式的Web应用程序来说，视图就是由[HTML](https://baike.baidu.com/item/HTML" \t "https://baike.baidu.com/item/MVC%E6%A1%86%E6%9E%B6/_blank)元素组成的界面，在新式的Web应用程序中，[HTML](https://baike.baidu.com/item/HTML" \t "https://baike.baidu.com/item/MVC%E6%A1%86%E6%9E%B6/_blank)依旧在视图中扮演着重要的角色，但一些新的技术已层出不穷，它们包括[Adobe Flash](https://baike.baidu.com/item/Adobe Flash" \t "https://baike.baidu.com/item/MVC%E6%A1%86%E6%9E%B6/_blank)和像[XHTML](https://baike.baidu.com/item/XHTML" \t "https://baike.baidu.com/item/MVC%E6%A1%86%E6%9E%B6/_blank)，[XML](https://baike.baidu.com/item/XML" \t "https://baike.baidu.com/item/MVC%E6%A1%86%E6%9E%B6/_blank)/[XSL](https://baike.baidu.com/item/XSL" \t "https://baike.baidu.com/item/MVC%E6%A1%86%E6%9E%B6/_blank),[WML](https://baike.baidu.com/item/WML" \t "https://baike.baidu.com/item/MVC%E6%A1%86%E6%9E%B6/_blank)等一些标识语言和[Web services](https://baike.baidu.com/item/Web services" \t "https://baike.baidu.com/item/MVC%E6%A1%86%E6%9E%B6/_blank).

MVC好处是它能为应用程序处理很多不同的[视图](https://baike.baidu.com/item/%E8%A7%86%E5%9B%BE" \t "https://baike.baidu.com/item/MVC%E6%A1%86%E6%9E%B6/_blank)。在视图中其实没有真正的处理发生，不管这些数据是联机存储的还是一个雇员列表，作为视图来讲，它只是作为一种输出数据并允许用户操纵的方式。 [6]

### 模型

模型表示企业数据和业务规则。在MVC的三个部件中，模型拥有最多的处理任务。例如它可能用像[EJB](https://baike.baidu.com/item/EJB" \t "https://baike.baidu.com/item/MVC%E6%A1%86%E6%9E%B6/_blank)s和ColdFusion Components这样的构件对象来处理数据库，被模型返回的数据是中立的，就是说模型与数据格式无关，这样一个模型能为多个视图提供数据，由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。 [6]

### 控制器

控制器接受用户的输入并调用模型和视图去完成用户的需求，所以当单击Web页面中的超链接和发送[HTML表单](https://baike.baidu.com/item/HTML%E8%A1%A8%E5%8D%95" \t "https://baike.baidu.com/item/MVC%E6%A1%86%E6%9E%B6/_blank)时，控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后再确定用哪个视图来显示返回的数据。 [6] 

132数组和链表的区别及优缺点

133解释一下全局变量局部变量动态变量

**1、变量的分类**

l         根据作用域：可分为全局变量和局部变量。

l           根据生存周期：可分为静态存储方式和动态存储方式，具体地又分为自动的（auto）、静态的（static）、寄存器的（register）、外部的（extern）。

**静态存储方式**是指在**程序运行期间分配固定的存储空间的方式，动态存储方式**是在**程序运行期间根据需要进行动态的分配存储空间的方式**。

**2、变量的作用域和存储类别的关系**

每一个变量均有**作用域**和**存储类别**两个**属性**，**这些属性共同用于描述一个变量**，这些不同类型的变量与存储位置的关系如下：

l          **外部变量（全局变量）、静态外部变量、静态局部变量存储在静态存储区。**

l          自动局部变量（**局部变量默认为自动局部变量**）、函数形参**存储在动态存储区（即栈区）**。

l          **不论是静态存储区还是动态存储区均属于内存中的用户区。**

l          **而寄存器变量是存储在CPU寄存器中的而不是内存中。**

**3、与作用域相关的几个属性：**

l          **局部变量**：在一个函数内部定义的变量是内部变量，它只在本函数范围内有效，也就是说只有在本函数内才能使用它们，在此函数以外是不能使用这些变量的，这种类型的变量就称为“局部变量”。

l          **全局变量**：在函数外定义的变量，可以为本源文件中其它函数所公用，有效范围为从定义变量的位置开始到本源文件结束，这种类型的变量就称为“全局变量”。

**4、与存储类型相关的几个属性：**

l          **atuo：**在声明局部变量时，若不指定 static，默认均是 auto，这类变量都是动态分配存储空间的，数据存储在动态存储区中。

l         **static：**在声明局部变量时，使用关键字 static 将局部变量指定为“静态局部变量”，这样在函数调用结束后不消失而保留原值，即占用的存储单元不释放，在下一次函数调用时，该变量已有值就是上次函数调用结束时的值。

l          register：**在声明动态局部变量或者函数形参时，可将变量声明为register**，**这样编译系统就会为变量分配一个寄存器而不是内存空间，通过这种方式可提升对某些局部变量频繁调用的程序的性能。**（**寄存器运算速度远高于内存**）

l          extern：用于**扩展**全局变量的作用域。

1)         比如如果函数想引用一个外部变量，但该外部变量在该函数后定义，那么这个函数需要使用 extern来声明变量，这样才能使用在该函数后面定义的全局变量。

2)         此外，extern 还可以在多文件的程序中声明外部变量。

134什么是数据模型

135关系模型用什么表示   关系不是二维表嘛

136奈奎斯特抽样

137.建模语言，算法描述（不会，说了好多不明不白，也没有解释好）

138折半查找简介，折半查找数组描还是很重要的，

1. **mac地址与ip地址的区别，数据链路层和网络层的接口**

1. 对于网络上的某一设备，如一台计算机或一台路由器，其IP地址可变（但必须唯一），而MAC地址不可变。我们可以根据需要给一台主机指定任意的IP地址，如我们可以给局域网上的某台计算机分配IP地址为192.168.0.112 ，也可以将它改成192.168.0.200。而任一网络设备（如网卡，路由器）一旦生产出来以后，其MAC地址永远唯一且不能由用户改变。

　　2. 长度不同。IP地址为32位，MAC地址为48位。

　　3. 分配依据不同。IP地址的分配是基于网络拓朴，MAC地址的分配是基于制造商。

4. 寻址协议层不同。IP地址应用于OSI第三层，即网络层，而MAC地址应用在OSI第二层，即数据链路层。 数据链路层协议可以使数据从一个节点传递到相同链路的另一个节点上（通过MAC地址），而网络层协议使数据可以从一个网络传递到另一个网络上（ARP根据目的IP地址，找到中间节点的MAC地址，通过中间节点传送，从而最终到达目的网络）。

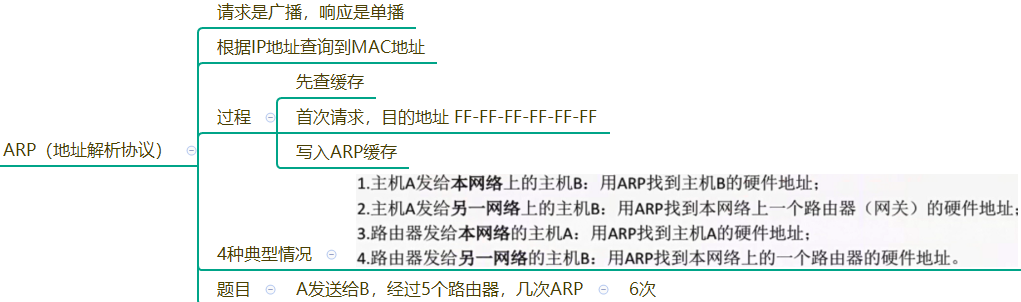
mac地址和ip地址有什么区别

IP地址是指Internet协议使用的地址，而MAC地址是Ethernet协议使用的地址。IP地址与MAC地址之间并没有什么必然的联系，MAC地址是Ethernet网卡上带的地址，长度为48位。

每个Ethernet网卡生产厂家必须向IEEE组织申请一组MAC地址，在生产网卡时在网卡的串行EEPROM中写入一个唯一的MAC地址。任何两个Ethernet网卡的MAC地址，不管是哪一个厂家生产的都不应相同。Ethernet芯片厂家不必负责MAC地址的申请，MAC地址存在于每一个Ethernet包中，是Ethernet包头的组成部分，Ethernet交换机根据Ethernet包头中的MAC源地址和MAC目的地址实现包的交换和传递。

IP地址是Internet协议地址，每个Internet包必须带有IP地址，每个Internet服务提供商（ISP）必须向有关组织申请一组IP地址，然后一般是动态分配给其用户。IP地址现是32位长，正在扩充到128位。IP地址与MAC地址无关，因为Ethernet的用户，仍然可通过Modem连接Internet，取得一个动态的IP地址，这个地址每次可以不一致。IP地址通常工作于广域网，路由器处理的就是IP地址。 MAC地址工作于局域网，局域网之间的互连一般通过现有的公用网或专用线路，需要进行网间协议转换。可以在Ethernet上传送IP信息，此时IP地址只是Ethernet信息包数据域的一部分，Ethernet交换机或处理器看不见IP地址，只是将其作为普通数据处理。

1. **arp协议**



141软件开发面向底层多还是高层的多，十万行代码与一万行代码的安全性倍数关系

142你认为软件工程平时codeing时间和代码量

（本来还想问我自控，我实在不会了，学习时间8小时，主考官说要18小时）

143静态变量是什么

**静态变量**（*Static Variable*）在计算机编程领域指在[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F/13831935" \t "https://baike.baidu.com/item/%E9%9D%99%E6%80%81%E5%8F%98%E9%87%8F/_blank)执行前系统就为之静态分配（也即在运行时中不再改变分配情况）存储空间的一类[变量](https://baike.baidu.com/item/%E5%8F%98%E9%87%8F" \t "https://baike.baidu.com/item/%E9%9D%99%E6%80%81%E5%8F%98%E9%87%8F/_blank)。与之相对应的是在运行时只暂时存在的[自动变量](https://baike.baidu.com/item/%E8%87%AA%E5%8A%A8%E5%8F%98%E9%87%8F/2646909" \t "https://baike.baidu.com/item/%E9%9D%99%E6%80%81%E5%8F%98%E9%87%8F/_blank)（即局部变量）与以动态分配方式获取存储空间的一些对象，其中自动变量的存储空间在[调用栈](https://baike.baidu.com/item/%E8%B0%83%E7%94%A8%E6%A0%88" \t "https://baike.baidu.com/item/%E9%9D%99%E6%80%81%E5%8F%98%E9%87%8F/_blank)上分配与释放。

144编程实现输入字符串再把它输出

145你说一下字符串和字符数组的区别

146进程和线程

147进程的资源可以共享吗，那线程的呢

148比如说服务器怎么做到负载均衡

负载均衡（Load Balance）  
由于目前现有网络的各个核心部分随着业务量的提高，访问量和数据流量的快速增长，其处理能力和计算强度也相应地增大，使得单一的服务器设备根本无法承担。在此情况下，如果扔掉现有设备去做大量的硬件升级，这样将造成现有资源的浪费，而且如果再面临下一次业务量的提升时，这又将导致再一次硬件升级的高额成本投入，甚至性能再卓越的设备也不能满足当前业务量增长的需求。  
针对此情况而衍生出来的一种廉价有效透明的方法以扩展现有网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力、提高网络的灵活性和可用性的技术就是负载均衡（Load Balance）。  
负载均衡技术主要应用  
1、DNS负载均衡 最早的负载均衡技术是通过DNS来实现的，在DNS中为多个地址配置同一个名字，因而查询这个名字的客户机将得到其中一个地址，从而使得不同的客户访问不同的服务器，达到负载均衡的目的。DNS负载均衡是一种简单而有效的方法，但是它不能区分服务器的差异，也不能反映服务器的当前运行状态。  
2、代理服务器负载均衡 使用代理服务器，可以将请求转发给内部的服务器，使用这种加速模式显然可以提升静态网页的访问速度。然而，也可以考虑这样一种技术，使用代理服务器将请求均匀转发给多台服务器，从而达到负载均衡的目的。  
3、地址转换网关负载均衡 支持负载均衡的地址转换网关，可以将一个外部IP地址映射为多个内部IP地址，对每次TCP连接请求动态使用其中一个内部地址，达到负载均衡的目的。  
4、协议内部支持负载均衡 除了这三种负载均衡方式之外，有的协议内部支持与负载均衡相关的功能，例如HTTP协议中的重定向能力等，HTTP运行于TCP连接的最高层。  
5、NAT负载均衡 NAT（Network Address Translation 网络地址转换）简单地说就是将一个[IP地址转换](https://www.baidu.com/s?wd=IP%E5%9C%B0%E5%9D%80%E8%BD%AC%E6%8D%A2&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)为另一个IP地址，一般用于未经注册的内部地址与合法的、已获注册的Internet IP地址间进行转换。适用于解决Internet IP地址紧张、不想让网络外部知道内部网络结构等的场合下。  
6、[反向代理](https://www.baidu.com/s?wd=%E5%8F%8D%E5%90%91%E4%BB%A3%E7%90%86&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)负载均衡 普通代理方式是代理内部网络用户访问internet上服务器的连接请求，客户端必须指定代理服务器,并将本来要直接发送到internet上服务器的连接请求发送给代理服务器处理。[反向代理](https://www.baidu.com/s?wd=%E5%8F%8D%E5%90%91%E4%BB%A3%E7%90%86&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)（Reverse Proxy）方式是指以代理服务器来接受internet上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给internet上请求连接的客户端，此时代理服务器对外就表现为一个服务器。[反向代理](https://www.baidu.com/s?wd=%E5%8F%8D%E5%90%91%E4%BB%A3%E7%90%86&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)负载均衡技术是把将来自internet上的连接请求以反向代理的方式动态地转发给内部网络上的多台服务器进行处理，从而达到负载均衡的目的。  
7、混合型负载均衡 在有些大型网络，由于多个服务器群内硬件设备、各自的规模、提供的服务等的差异，我们可以考虑给每个服务器群采用最合适的负载均衡方式，然后又在这多个服务器群间再一次负载均衡或群集起来以一个整体向外界提供服务（即把这多个服务器群当做一个新的服务器群），从而达到最佳的性能。我们将这种方式称之为混合型负载均衡。此种方式有时也用于单台均衡设备的性能不能满足大量连接请求的情况下。

149怎么解决12306同时订票

150怎么解决服务器主动访问客户端

151如果电信服务器崩溃了使用他的相关网站怎么办

152还有怎么解决大家访问网站上传下载资源慢，除了增加带宽

153还问了一些本科软工的名词

154.dsp程序设计和51单片机程序设计的最大区别是啥

155.cache的换入换出和存储器的换入换出有什么区别

替换策略：

一、最不经常使用算法（Least Frequently Used，LFU）：

LFU算法认为：应该将一段时间内被访问次数最少的那行数据替换出。

实现方法：每行设置一个计数器。新行建立后从0开始计数，每被访问一次，被访问行的计数器增1。当需要替换时，对这些特定行的计数值进行比较，将计数值最小的行换出，同时将这些特定行的计数器清零。

该算法不能严格反映近期访问情况。

二、近期最少使用算法（Least Recently Used，LRU）：

LFU算法将近期内长久未被访问过的行换出。

实现方法：每行设置一个计数器，但它们是cache每命中一次，命中行计数器清零，其他各行计数器增1。当需要替换时，将计数值最大的行换出。

该算法保护了刚拷贝到cache中的新数据行，较为符合cache的工作原理，有效地提高了命中率。

三、随机替换：

不需要算法，从特定的行位置中随机地选取一行换出即可。

该策略在硬件上容易实现，速度较前两种策略快。但是会在一定程度上降低cache的效率。这些不足会随着cache容量增大而减小。

写操作策略：

一、写回法：

要求：当CPU写cache命中时，只修改cache中的内容，不立即写入主存，只有当此行被换出时才写回主存。当CPU写未命中时，为了包含欲写字的主存块在cache中分配一行，将此块拷贝到cache后再进行修改。主存的写修改操作统一地留到换出时再进行。

实现：每个cache行配置修改位，以反映CPU是否对该行修改过。该行被换出时，根据修改位的内容决定改行内容是写回主存还是简单弃去。

特点：写cache与写主存异步进行，可以减少访存次数，但是存在cache与主存不一致性的隐患。同时，还要设置修改位，相应的逻辑较为复杂。

二、全写法：

要求：当写cache命中时，cache与主存同步发生修改。当写cache未命中时，直接向主存进行写入，但是修改过的主存块是否拷贝到cache，有WTWA法（将修改后的主存块立即拷贝到cache），WTNWA法（不会将修改后的主存块立即拷贝到cache）。

特点：实现写cache与写主存同步，有效地保证了cache与主存内容的一致性。无需设置修改位，逻辑较为容易。但由于写cache未命中时，即cache对CPU向主存的写操作无高速缓冲功能，降低了cache的功效。

三、写一次法：

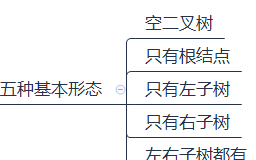
基于上述两种写策略的综合，第一次写命中时，采用全写法；后续采用写回法。

1. CDN是什么

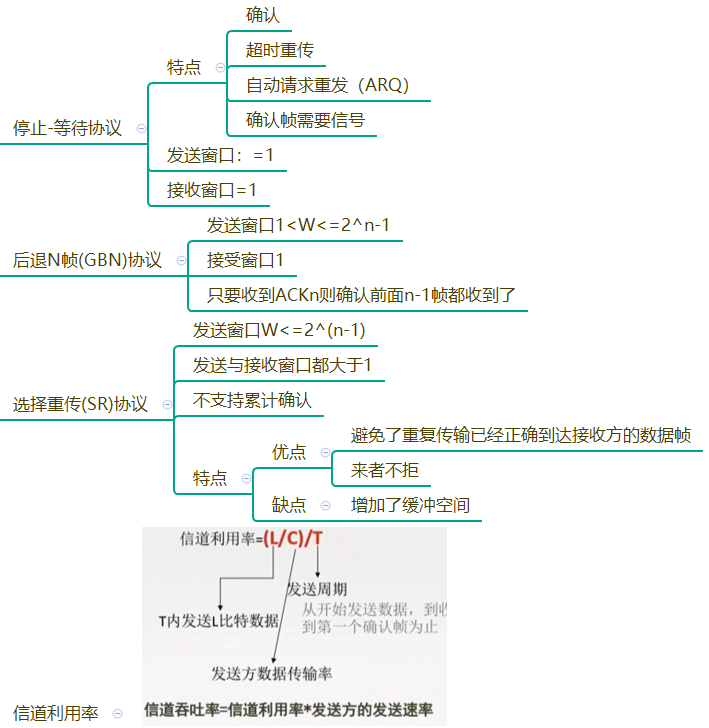
CDN的全称是Content Delivery Network，即[内容分发网络](https://baike.baidu.com/item/%E5%86%85%E5%AE%B9%E5%88%86%E5%8F%91%E7%BD%91%E7%BB%9C/4034265" \t "https://baike.baidu.com/item/CDN/_blank)。CDN是构建在网络之上的内容分发网络，依靠部署在各地的边缘服务器，通过中心平台的负载均衡、内容分发、调度等功能模块，使用户就近获取所需内容，降低网络拥塞，提高用户访问响应速度和命中率。CDN的关键技术主要有内容存储和分发技术。

157.问了我java和c++的本质区别

158.二叉树和节点只有两个孩子的树有什么区别



1. 滑动窗口



160软件设计过程

161.uml

162.再问网络运用哪些数据结构。

1.OSPF使用使用Dijkstra计算最优路径

2.路由器主要：路由选择与分组转发。分组转发中要排队用到队列

3.网页之间类似于有向图

4.DNS解析树状结构

5.缓存，一般用队列

163.问的我计算机组成，

冯诺依曼模型：以运算器为核心，包括：运算器，控制器，存储器，输入输出设备

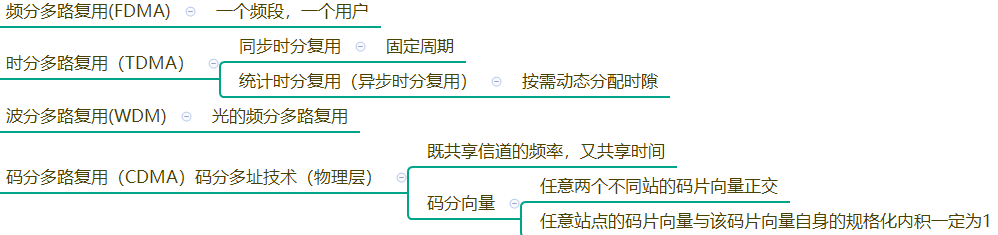
164快速排序

165测试的目的是确认范围，保证质量

166.pid 是操作系统标识符

167.信道复用

信道划分介质访问控制（静态）：



随机访问介质访问控制（动态）：

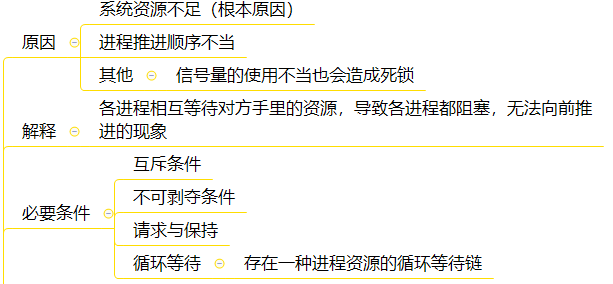
ALOHA协议，CSMA协议（载波侦听多路访问），CSMA/CD协议（带冲突检测的载波侦听多路访问协议），CSMA/CA协议，带碰撞避免的载波侦听多路访问协议

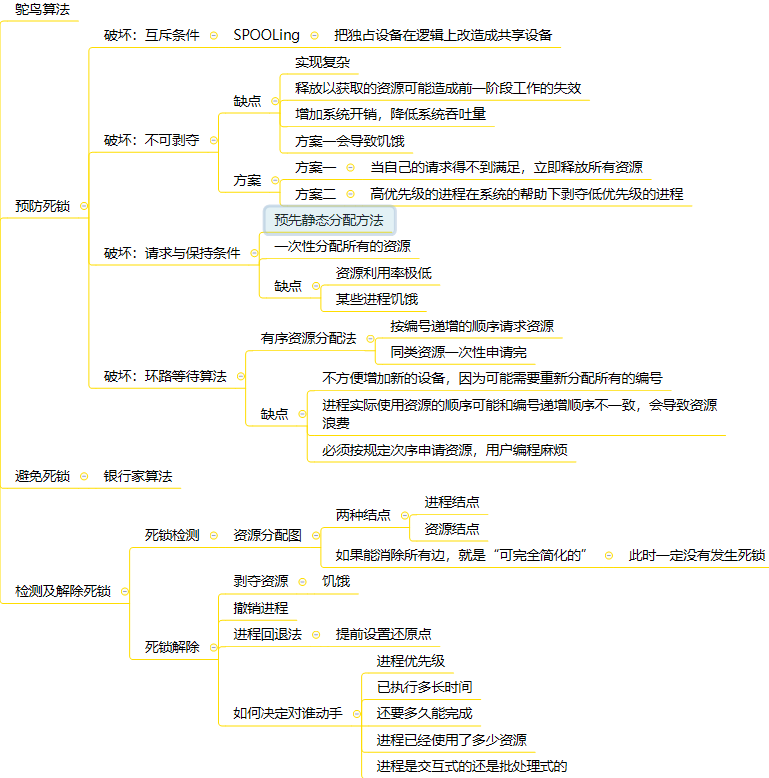
论询访问介质访问控制（动态）：令牌传递协议（受控接入）

168.网络：拥塞控制

169.数据库读写锁

170.操作系统死锁





1. osi的英文和中文全称

OSI模型，即**[开放系统互连](https://www.baidu.com/s?wd=%E5%BC%80%E6%94%BE%E7%B3%BB%E7%BB%9F%E4%BA%92%E8%BF%9E&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)基本参考模型**(OSI/RM)，是[国际标准组织](https://www.baidu.com/s?wd=%E5%9B%BD%E9%99%85%E6%A0%87%E5%87%86%E7%BB%84%E7%BB%87&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)(ISO)提出的一个试图使各种计算机在世界范围内互连为网络的标准框架，简称OSI。在1983年形成了OSI/RM的正式文件，即ISO 7498国际标准。OSI将计算机网络体系结构(architecture)划分为以下七层：  
物理层：Physical Layer  
数据链路层：Data Link Layer  
网络层：Network Layer  
传输层：Transport Layer  
会话层：Session Layer  
表示层：Presentation Layer  
应用层：Application Layer  
OSI/RM”来自英文“**Open System Interconnection Reference Model”**缩写

1. 递归算法求阶乘

#include<stdio.h>

int fact(int); //声明阶乘fact函数

int main(){

int x;

scanf("%d",&x);

x = fact(x);//调用函数返回int值

printf("%d\n",x);

}

int fact(int n){//定义阶乘函数

if(n==1) return 1;//输入的参数是1，直接返回1

else return n\*fact(n-1);//递归算法

}

1. 简述快排，快排时间复杂度，最好最坏，分别是什么情况下

快速排序的时间性能取决于快速排序递归的深度，可以用递归树来描述递归算法的执行情况。如图 9‐9‐7 所示，它是 {50,10,90,30, 70,40,80,60,20} 在快速排序过程中的递归过程。由于我们的第一个关键字是 50，正好是待排序的序列的中间值，因此递归树是平衡的，此时性能也比较好。

在最坏的情况下，待排序的序列为正序或者逆序，每次划分只得到一个比上一次划分少一个记录的子序列，注意另一个为空。如果递归树画出来，它就是一棵斜树。此时需要执行 n‐1 次递归调用，且第 i 次划分需要经过 n‐i 次关键字的比较才能找到第 i 个记录，也就是枢轴的位置，因此比较次数为最终其时间复杂度为 O(n2)。

就空间复杂度来说，主要是递归造成的栈空间的使用，最好情况，递归树的深度为 log2n，其空间复杂度也就为 O(logn)，最坏情况，需要进行 n‐1 递归调用，其空间复杂度为 O(n)，平均情况，空间复杂度也为 O(logn)。

可惜的是，由于关键字的比较和交换是跳跃进行的，因此，快速排序是一种不稳定的排序方法。

174.离散：群，集群等概念问题   我说我学的不好，能不能换个题目

175.死锁是什么，为什么发生死锁，怎么解决死锁

176.你说说软件工程和计算机技术这两个专业有什么区别？

1、专业方面不同

[计算机科学与技术](https://www.baidu.com/s?wd=%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%A7%91%E5%AD%A6%E4%B8%8E%E6%8A%80%E6%9C%AF&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)就相当于工科里的偏理的一面，是为了研究与解决算法问题而出现的科学。而软件工程是比较典型的工科专业，主要是编程和随之而来的计算机组件交互的问题，是丛计算机科学中分出来的一个分支。

2、技术方面不同

计算机专业涉及到硬件，软件工程只涉及软件

177.传统软件开发过程是什么？

传统的 [1]  是一个文档驱动的流程，它将整个软件开发过程划分为顺序相接的几个阶段，每个阶段都必需完成全部规定的任务（文档）后才能够进入下一个阶段。如必须完成全部的系统需求规格说明书之后才能够进入概要设计阶段，编码必需在系统设计完成之后才能够进行。这就意味着只有当所有的系统模块全部开发完成之 后，我们才进行系统集成，对于一个由上百个模块组的复杂系统来说，这是一个非常艰巨而漫长的工作。 [1]

可行性分析和项目开发计划

↓  
　　需求分析  
　　↓  
　　概要设计  
　　↓  
　　详细设计  
　　↓  
　　编码  
　　↓  
　　测试  
　　↓  
　　维护

178.树的层序遍历

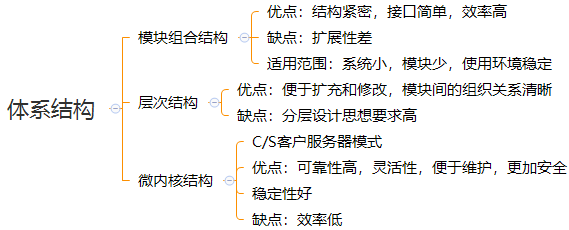
179.c++ Java 在继承上有什么区别

C++的多重继承简要而言来说，就是区别单继承的。也就是说，C++中，类继承不像java那样只能实行单继承，一个类能同时继承多个类。这符合[自然界](https://www.baidu.com/s?wd=%E8%87%AA%E7%84%B6%E7%95%8C&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://blog.csdn.net/yyhhlancelot/article/details/_blank)的规律，比如一个人同时继承自己父亲与母亲的基因。

也就是说，一个子类只能拥有一个父亲，所以extends只能继承一个类。

而java中所谓的多重继承，就是例如A类继承B类，B类继承C类（其实质也是单继承），这里就是与C++的区别。

1. 操作系统，什么是微内核



1. 两个网络靠什么可以将他们链接一起

路由器，异构网络互连

182.OS的三个指标是什么

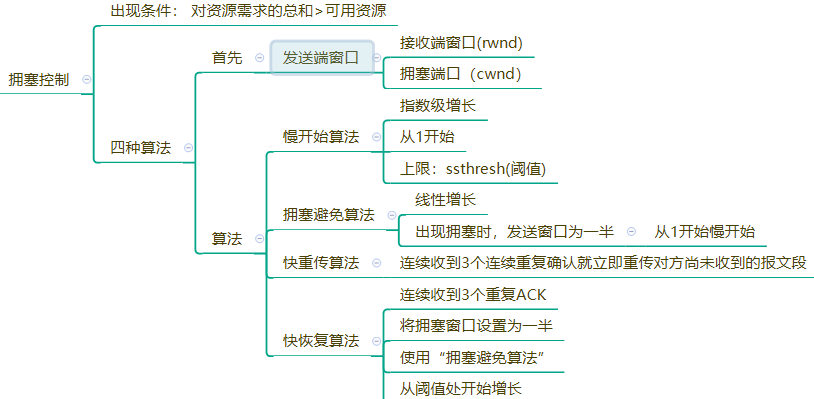
183.一个数字数组排序你会选择什么排序算法？

我回答冒泡排序，然后问其时间复杂度

184.进程三种状态以及什么情况下转换；

185.虚拟内存的作用，好处；

186.拥塞控制在哪个层，具体说明。



1. 全局变量与静态局部变量的定义，以及静态局部变量能否屏蔽全局变量

静态局部变量(static) 静态局部变量定义时前面加static关键字。

总结：

1、静态局部变量和普通局部变量不同。静态局部变量也是定义在函数内部的，静态局部变量定义时前面要加static关键字来标识，静态局部变量所在的函数在多调用多次时，只有第一次才经历变量定义和初始化，以后多次在调用时不再定义和初始化，而是维持之前上一次调用时执行后这个变量的值。本次接着来使用。

2、静态局部变量在第一次函数被调用时创造并初始化，但在函数退出时它不死亡，而是保持其值等待函数下一次被调用。下次调用时不再重新创造和初始化该变量，而是直接用上一次留下的值为基础来进行操作。

3、静态局部变量的这种特性，和全局变量非常类似。它们的相同点是都创造和初始化一次，以后调用时值保持上次的不变。不同点在于作用域不同

全局变量 定义在函数外面的变量，就叫全局变量。

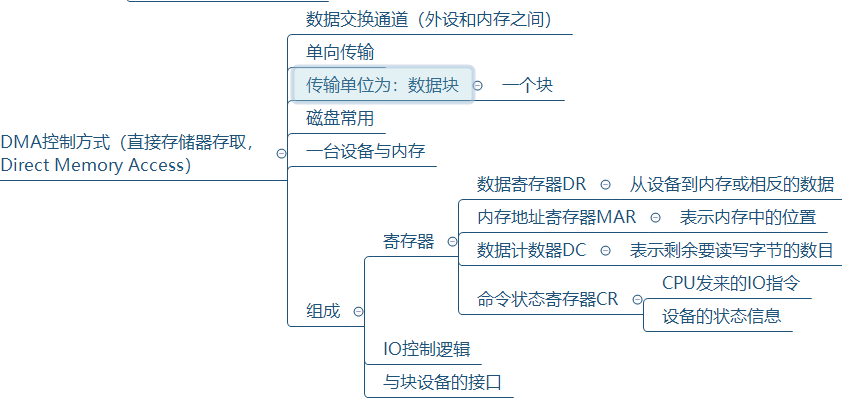
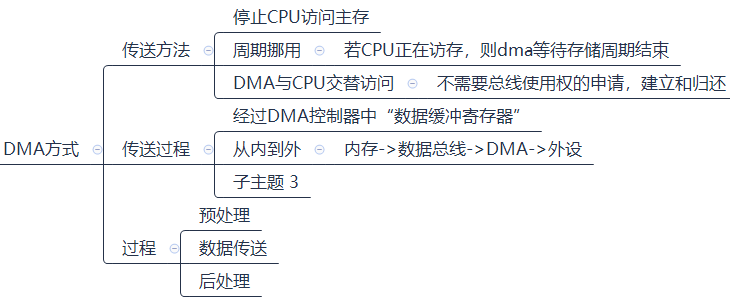
普通全局变量  普通全局变量就是平时使用的，定义前不加任何修饰词。普通全局变量可以在各个文件中使 用，可以在项目内别的.c文件中被看到，所以要确保不能重名。

静态全局变量  静态全局变量就是用来解决重名问题的。静态全局变量定义时在定义前加static关键字， 告诉编译器这个变量只在当前本文件内使用，在别的文件中绝对不会使用。这样就不用担心重名问题。所以静态的全局变量就用在我定义这个全局变量并不是为了给别的文件使用，本来就是给我这个文件自己使用的。

跨文件引用全局变量(extern) 就是说，你在一个程序的多个.c源文件中，可以在一个.c文件中定义全局变量g\_a,并且可以在别的另一个.c文件中引用该变量g\_a（引用前要声明）

函数和全局变量在C语言中可以跨文件引用，也就是说他们的连接范围是全局的，具有文件连接属性，总之意思就是全局变量和函数是可以跨文件看到的（直接影响就是，我在a.c和b.c中各自定义了一个函数func，名字相同但是内容不同，编译报错。）。

1. 什么是DMA



1. java和C语言的区别吗？

我说1、一个面向对象，一个面向过程。

1. java中不支持使用指针，c利用指针，java并不是不用指针而是编程者不使用。

3、java具有垃圾自动回收机制，c没有，c需要用free函数

190.哈夫曼树    说说它的应用

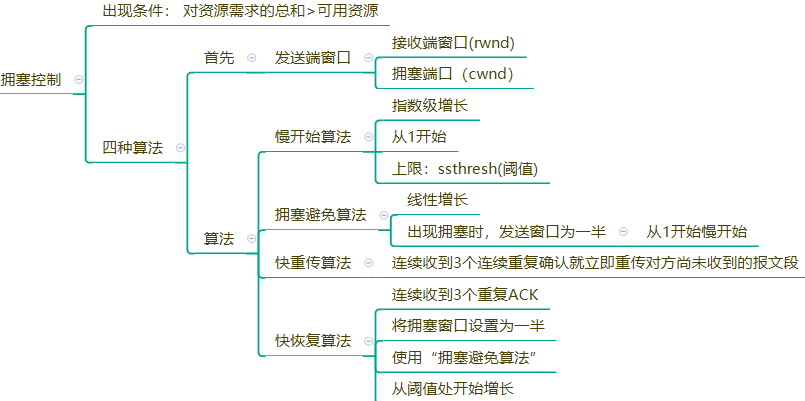
哈夫曼树：由n个带权叶子结点构成的所有二叉树中带权路径长度最短的二叉树

 哈夫曼编码：对一棵具有n个叶子的哈夫曼树，若对树中的每个左分支赋予0，右分支赋予1，则从根到每个叶子的通路上，各分支的赋值分别构成一个二进制串，该二进制串就称为哈夫曼编码。

        哈夫曼编码是最优前缀码。

191：（数据结构）图的存储结构？

192：（网络）拥塞控制在哪一层？



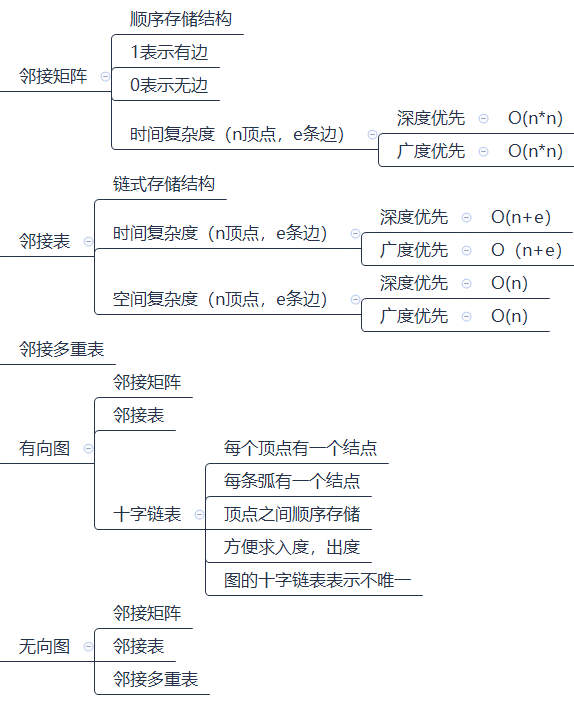
193：（数据库）数据模型？

数据模型是数据库的基础，数据是对客观事物的符号表示，模型是现实世界的抽象。数据模型是对数据特征的抽象。

将客观事物抽象为数据模型，是一个逐步转化的过程，经历了现实世界、信息世界和计算机世界这三个不同的世界，经历了两级抽象和转换，如下图所示：

1. 图的存储结构时，叫我介绍邻接矩阵具体是怎么存储的

邻接矩阵(adjacency matrix)是图ADT最基本的实现方式，使用二维数组A[n][n]来表示由n个顶点构成的图。二维数组中的A[i][j]表示一条从来顶点i为起点到顶点j的弧。



195.进程是什么，进程和线程的区别。

196.两个函数中的变量名一样，然后他们使用的时候是什么样呢？

197.malloc函数分配的内存是在哪里？

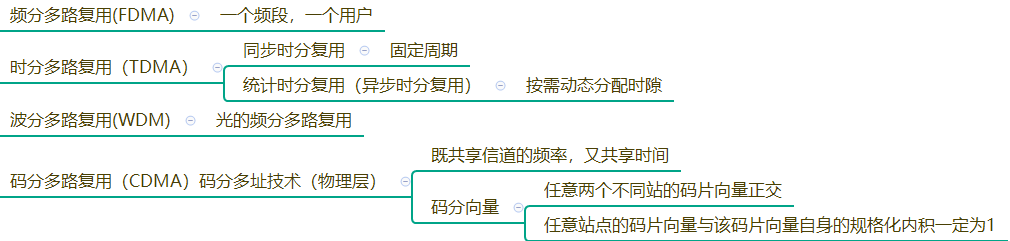
在C语言中，根据数据在内存中存在的时间（生存周期）不同，将内存空间分为三个区：

1）程序区：用于存储程序的代码，即程序的二进制代码。

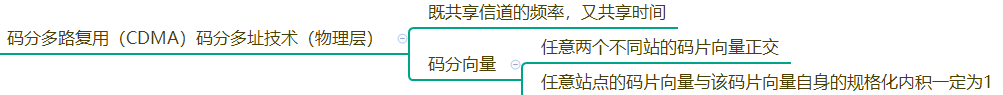
2）静态存储区：用于存储全局变量和静态变量，这些变量的空间在程序编译时就已经分配好了。

3）动态存储区：用于在程序执行时分配的内存，又分为：堆区（heap）和栈区（stack）。堆区:用于动态内存 分配，程序运行时由内存分配函数在堆上分配内存。在C语言中，只能使用指针才能动态的分配内存。栈区:在 函数执行时，函数内部的局部变量和函数参数的存储单元的内存区域，函数运行结束时，这些内存区域会自动释 放。   
　　2.2 C语言动态内存分配   
　　在C语言中用内存分配函数来实现内存的动态分配，这些函数有：malloc()和realloc()等函数。malloc():使用这个 函数时需要包含头文件。使用该函数需要指定要分配的内存字节数作为参数，例如：   
　　int \*pNumber=（int \*） malloc（100）   
　　这条语句分配了100个字节的内存，并把这个内存块的地址赋给pNumber，这个内存块可以保存最大25个int值， 每个int占4个字节。如果不能分配请求的内存，malloc（）会返回一个null指针。   
　　2.3 释放动态分配的内存   
　　堆上分配的内存会在整个应用程序结束之后，由操作系统负责回收，但最好是在使用完这些内存后立即释放。如 果不释放，会引起内存泄漏，极大占用系统资源，可能会产生各种未知的错误。所以，必须使用free（）函数释 放内存，参数是[内存地址](https://www.baidu.com/s?wd=%E5%86%85%E5%AD%98%E5%9C%B0%E5%9D%80&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://blog.csdn.net/xzengwei1313/article/details/_blank)（指针），例如：free（pNumber），依上例。

1. 信道划分介质访问控制（静态）（物理层）



1. 手机3G网WCDMA是用的什么复用方式



200.中断的过程   操作系统里的中断的实现过程吧

201.信号量机制的原理



1. 你刚刚说到网络，那么说说要你自己写一个网络协议你怎么做?

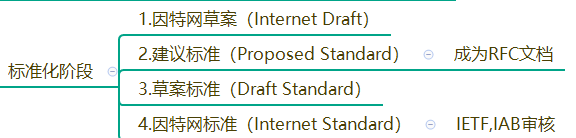
网络协议有基本的范式的：数据包长度+数据包关键字（命令名称、函数名、索引都行）+数据。

**网络协议是由三个要素组成： [2]**

(1) **语义**。语义是解释控制信息每个部分的意义。它规定了需要发出何种控制信息，以及完成的动作与做出什么样的响应。

(2) **语法**。语法是用户数据与控制信息的结构与格式，以及数据出现的顺序。

(3) **时序**。时序是对事件发生顺序的详细说明。（也可称为“同步”）。 [3]



1. 数据库的三级模式是什么

### **1.概念模式**

概念模式也称模式，是数据库中全部数据的逻辑结构和特征的描述，它由若干个概念记录类型组成，只涉及行的描述，不涉及具体的值。概念模式的一个具体值称为模式的一个实例，同一个模式可以有很多实例。

概念模式反映的是数据库的结构及其联系，所以是相对稳定的；而实例反映的是数据库某一时刻的状态，所以是相对变动的。

需要说明的是，概念模式不仅要描述概念记录类型，还要描述记录间的联系、操作、数据的完整性和安全性等要求。但是，概念模式不涉及存储结构、访问技术等细节。只有这样，概念模式才算做到了“物理数据独立性”。

描述概念模式的数据定义语言称为“模式DDL”

### **2.外模式**

外模式也称用户模式或子模式，是用户与数据库系统的接口，是用户用到的那部分数据的描述。它由若干个外部记录类型组成。用户使用数据操纵语言对数据库进行操作，实际上是对外模式的外部记录进行操作。

描述外模式的数据定义语言称为“外模式DDL”。有了外模式后，程序员不必关心概念模式，只与外模式发生联系，按外模式的结构存储和操作数据。

### **3.内模式**

内模式也称为存储模式，是数据物理结构和存储方式的描述，是数据在数据库内部的表示方式。需要定义所以的内部记录类型、索引和文件的组织方式，以及数据控制方面的细节。

例如，记录的存储方式是顺序存储、B树结构存储还是Hash方法存储；索引按照什么方式组织；数据是否压缩存储，是否加密；数据的存储记录结构有何规定。

需要说明的是，内部记录并不涉及物理记录，也不涉及设备的约束。比内模式更接近于物理存储和访问的那些软件机制是操作系统的一部分（即文件系统）。例如，从磁盘上读、写数据。

描述内模式的数据定义语言称为“内模式DDL”。

总之，数据按外模式的描述提供给用户；按内模式的描述存储在磁盘上；而概念模式提供了连接这两级模式的相对稳定的中间层，并使得两级中任意一级的改变都不受另一级的牵制。

## **三级模式两级映像**

数据库系统在三级模式之间提供了两级映像：模式/内模式的映像、外模式/模式的映像。这两级映射保证了数据库中的数据具有较高的物理独立性和逻辑独立性。

•模式/内模式的映像：实现概念模式到内模式之间的相互转换。

•外模式/模式的映像：实现外模式到概念模式之间的相互转换。

数据的独立性是指数据与程序独立，将数据的定义从程序中分离出去，由DBMS负责数据的存储，从而简化应用程序，大大减少应用程序编制的工作量。数据的独立性是由DBMS的二级映像功能来保证的。数据的独立性包括数据的物理独立性和数据的逻辑独立性。

数据的物理独立性是指当数据库的内模式发生改变时，数据的的逻辑结构不变。由于应用程序处理的只是数据的逻辑结构，这样物理独立性可以保证，当数据的物理结构改变了，应用程序不用改变。但是，为了保证应用程序能够正确执行，需要修改概念模式/内模式之间的映像。

数据的逻辑独立性是指用户的应用程序与数据库结构是相互独立的。数据的逻辑结构发生变化后，用户程序也可以不修改。但是，为了保证应用程序能够正确执行，需要修改外模式/概念模式之间的映像。

204.什么是偏序集

205.OR映射

实体对象采用的是面向对象技术，它表达实体的关系是用关联、继承。而RDBMS 是遵循关系的范式理论构建的二维表,主要采用主键和外键的关联方式。因此，对象模型与数据库模型是不一致的，需要在两者之间进行变换即映射。  
映射的完成可以按照规则进行人工映射，可以从O映射到R，反之亦然。可以借助开发工具的正、反向工程完成映射。Hibernate就是管理OR映射的容器。

**206.数据库怎么对共享数据进行保护**

**数据库完整性**： 正确性、有效性、相容性

**完整性约束：**主键约束，外键约束， 属性约束， 域约束， 断言约束， 触发器约束

**数据库安全性：**指数据库的任何部分都不允许受到恶意侵害或未经授权的存取和修改

层次：物理层、操作系统层、网络层、人员层、数据库系统层

   技术：

    身份认证：用户名加口令、生物特征识别技术、基于USB Key的身份认证方法

    存取控制：不同用户可以被授权使用相同或不同的数据库数据集合，通过数据库系统的存取控制机制实现

    建立视图：隐藏用户不需要看见的数据，其构造基于基本表或视图

    数据加密：根据一定的算法将原始数据变换为不可是别的格式，从而使得不知道解密算法的人无法获知数据的内容

    审计跟踪：即监视功能

事物：用户定义的一个数据库操作序列，这些操作要么全做，要么全不做，是一个不可分割的工作单位

特性：

 原子性：事物的所有操作要么都做，要么都不做

 一致性：事物单独执行时，即在没有和其他事物并发执行的情况下，保持数据库的一致性

 隔离性：一个事物的执行不能被其他事物干扰

 持久性：一个事物一旦提交成功后，他对数据库中的数据的改变应该是永久的，即使系统出现故障

**并发控制**：在多个事物对数据库并发操作的情况下，对数据库实行的管理和控制

    并发控制的好处：1.提高吞吐量和资源利用率2.减少等待时间

带来的数据不一致性问题：

丢失更新：事物并发执行时，一个事物的提交的结果破坏了另外一个事物提交的结果，导致一个修改丢失

不可重复读：一事物读取数据后，另一事物执行更新操作，使上一个事物无法在重复前一次的结果

读“脏”数据：事物1修改了某一数据并将其写回磁盘，事物2读取同一数据后，事物1由于某种原因被撤销，这时事物1将已修改过的数据恢复原值，事物2读到的数据就与数据库中的数据不一致，这种读也称为脏读

原因：事物执行时收到其他事物干扰，破坏了事物的隔离性

 解决方法：封锁技术和时标技术

**封锁**：

**排他锁（X锁）**：只用获得了数据的X锁的事物才可以对数据进行读取与修改，也不能再对该数据加其他锁

**共享锁（S锁）**：获得数据共享锁的事物可读但不可修改数据，其他事物只能加S锁不能加X锁

**封锁协议（三级）：**

**一级封锁协议**：事物T在修改数据A前必须先对其加X锁，直到事物结束才释放，可解决更新丢失，不能解决后两个问题

**二级封锁协议**：一级的基础上，事物T在对数据A进行读操作前必须先对A加S锁，读完后马上释放S锁，可防止读“脏”数据

**三级封锁协议**：一级的基础上，事物T在对数据A进行读操作前必须先对A加S锁，事物完成后释放S锁，可防止不可重复读与读“脏”数据

活锁：无限等待，得不到封锁，设置优先级

死锁：事物均等待对方解除封锁

预防：一次加锁法，顺序加锁法

 诊断：事物依赖图

解除：选择一个处理死锁代价最小的事物，将其撤销，恢复对数据库的改变

故障：事物故障、系统故障、介质故障、计算机病毒

恢复：数据转储、日志文件

207.解释一下KMP算法

208.TCP的段头里窗口字段表示什么意思

1. JSP与html有什么区别？

1、HTML能直接打开，jsp只能发布到Tomact等服务器上才能打开 。

2、定义上HTML页面是静态页面可以直接运行，JSP页面是动态页它运行时需要转换成servlet。

3、他们的表头不同，这个是JSP的头“ <%@ page language="java" import="java.util.\*" pageEncoding="gbk"%>”在表头中有编码格式和倒入包等。

4、在jsp中用<%%>就可以写Java代码了，而html没有<%%>

二、在html里面是无法进行java编写的。原因：JSP页面是动态页，HTML是静态的。所以不支持。

定义：

1、JSP：全名为Java Server Pages，中文名叫java服务器页面，其根本是一个简化的Servlet设计，它是由Sun Microsystems公司倡导、许多公司参与一起建立的一种动态网页技术标准。

2、HTML：超级文本标记语言是标准通用标记语言下的一个应用，也是一种规范，一种标准，它通过标记符号来标记要显示的网页中的各个部分。

扩展资料：

jsp与servlet的联系:

JSP是Servlet技术的扩展，本质上就是Servlet的简易方式。JSP编译后是“类servlet”。

Servlet和JSP最主要的不同点在于：Servlet的应用逻辑是在Java文件中，并且完全从表示层中的HTML里分离开来。而JSP的情况是Java和HTML可以组合成一个扩展名为.jsp的文件。

JSP侧重于视图，Servlet主要用于控制逻辑Servlet更多的是类似于一个Controller，用来做控制。

210.快速排序平均复杂度是多少？O(log(n)).      最坏时间复杂度是多少？O(n^2).       什么时候达到最坏？ 这个问题的答案应该是逆序，这是后来才想出来的，但是由于当时紧张没有回答出来。老师又问：“空间复杂度低是多少？请举例说明。” O(1)

211.TCP包有一个窗口字段，作用是什么？

212.滑动窗口是干什么用的   “流量控制

213.二叉树的层序遍历什么

214.分页是什

215.什么事软实时和硬实时

1. 图的两种存储结构
2. 哈希表的冲突及解决方法
3. 矩阵法存储（不论稀疏图还是稠密图）有什么优点 我刚开始还没听懂，后来才改过来，是能直接判断两点之间是否有边存在。就哈希表我当时都蒙了，反正这个问题没说好，解决方法有再散列法跟拉链法，简要说了一下这两种方法的原理。
4. 什么是中断？虽然心里有些小明白，但是不知道怎么表述，就说外部I/O对CPU的请求什么的......专业面老师还好，并没有多严厉！

219.操作系统抽到的是TLB

220.虚拟存储器是什么

221.快排的最快和最慢时间复杂度

222.操作系统问题是什么是cache

223.cache TLB区别

224.网络的题是什么是QOS

**225.计算机网络是数据链路层停等协议**

**226.数据结构是二分查找及其适用范围**

227.图论

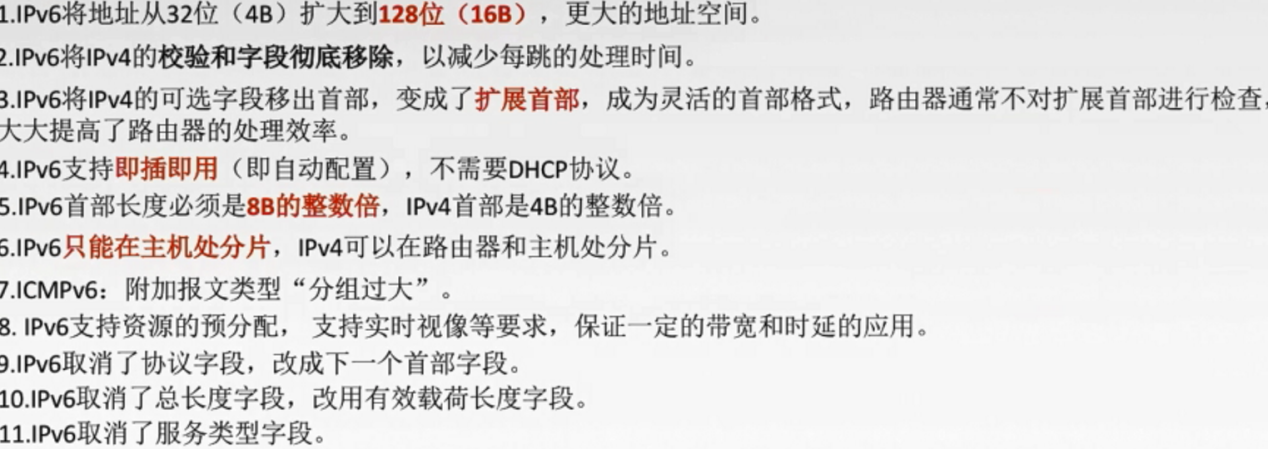
228.生成树   最小生成树

229.什么是形参，什么是实参及其区别（大概说了下）

230.C语言能传地址吗（这个问题我没听清楚，就说能），怎么传？

231.什么是二叉排序树，什么是平衡二叉树，如何调整

1. 单链表逆置如何实现？
2. 测试如何从已知bug，推出新的测试的覆盖率90%？
3. 传值和传引用的区别？
4. ipv4和ipv6？



235.让解释快速排序的过程，对面老教授接着问时间复杂度，我说nlogn；再问最坏时间复杂度，我n^2；又问什么情况下最坏，我，基本有序；旁边一个女老师问为什么基本有序最坏，我凭感觉说了一下，老师说不对，我就蔫了。老教授说让我结合nlogn说明一下一趟快速排序需要比较多少次，我想了一会儿，说n-1次；老师又问，第二次呢，我说(n-1)/2，然后就又错了。这个就是跨考的底子实在不行。老师就让我抽下一题了。

236.网络，抽到的是让我说明物理层、数据链路层、网络层的设备及作用， 集线器的本质是什么   连接完的拓扑结构？答星形

237.让解释Cache

238.解释链表下查找的时间复杂度，答n，原因在于不支持随机访问。

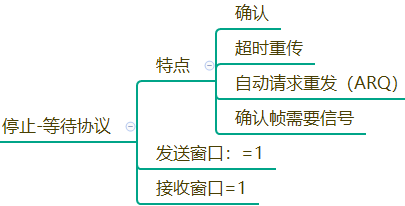
1. 中断的过程？



1. 什么是互斥？

相交进程之间的关系主要有两种，同步与互斥。所谓互斥，是指散步在不同进程之间的若干程序片断，当某个进程运行其中一个程序片段时，其它进程就不能运行它 们之中的任一程序片段，只能等到该进程运行完这个程序片段后才可以运行。所谓同步，是指散步在不同进程之间的若干程序片断，它们的运行必须严格按照规定的 某种先后次序来运行，这种先后次序依赖于要完成的特定的任务。  
　　显然，同步是一种更为复杂的互斥，而互斥是一种特殊的同步。  
　　也就是说互斥是两个线程之间不可以同时运行，他们会相互排斥，必须等待一个线程运行完毕，另一个才能运行，而同步也是不能同时运行，但他是必须要安照某种次序来运行相应的线程（也是一种互斥）！  
　　总结：互斥：是指某一资源同时只允许一个访问者对其进行访问，具有唯一性和排它性。但互斥无法限制访问者对资源的访问顺序，即访问是无序的。  
　　同步：是指在互斥的基础上（大多数情况），通过其它机制实现访问者对资源的有序访问。在大多数情况下，同步已经实现了互斥，特别是所有写入资源的情况必定是互斥的。少数情况是指可以允许多个访问者同时访问资源。

1. 线性表 队列？
2. 停等协议    停等的目的是啥

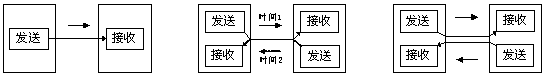


243.介绍折半查找       问时间复杂度，这个我给说错了，应该是log2n。

244.计算机网络    是单工，半双工，双工，

**单工数据传输**只支持数据在一个方向上传输；在同一时间只有一方能接受或发送信息，不能实现双向通信，举例：电视，广播。

**半双工数据传输**允许数据在两个方向上传输,但是,在某一时刻,只允许数据在一个方向上传输,它实际上是一种切换方向的[单工通信](https://www.baidu.com/s?wd=%E5%8D%95%E5%B7%A5%E9%80%9A%E4%BF%A1&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)；在同一时间只可以有一方接受或发送信息，可以实现双向通信。举例：对讲机。

[IMG_256](https://gss0.baidu.com/7Po3dSag_xI4khGko9WTAnF6hhy/zhidao/pic/item/78310a55b319ebc4902c8bb58f26cffc1e171641.jpg)[](https://gss0.baidu.com/7Po3dSag_xI4khGko9WTAnF6hhy/zhidao/pic/item/9e3df8dcd100baa12255ba974a10b912c8fc2ea5.jpg)

3.**全双工数据通信**允许数据同时在两个方向上传输,因此,全双工通信是两个[单工通信](https://www.baidu.com/s?wd=%E5%8D%95%E5%B7%A5%E9%80%9A%E4%BF%A1&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)方式的结合,它要求发送设备和接收设备都有**独立的接收和发送能力**；在同一时间可以同时接受和发送信息，实现双向通信，举例：电话通信。

245.讲一下数据结构的栈和队列？

246.C语言的指针吗？

**247.计算机网络分层的缺点     分层的缺点是不稳定，下层出了问题会影响上层**

**248.http的英文全称和acid是什么**

249.快速排序和C语言的函数调用

250.什么是优先级数组？

251.堆的存储结构是什么？

堆的物理存储结构是一维数组，逻辑存储结构是完全二叉树

 252.rup是什么  rup三个字母全称是啥  用例图里用例之间的关系有哪些

RUP（Rational Unified Process），[统一软件开发过程](https://baike.baidu.com/item/%E7%BB%9F%E4%B8%80%E8%BD%AF%E4%BB%B6%E5%BC%80%E5%8F%91%E8%BF%87%E7%A8%8B/2088774" \t "https://baike.baidu.com/item/RUP/_blank)，[统一软件过程](https://baike.baidu.com/item/%E7%BB%9F%E4%B8%80%E8%BD%AF%E4%BB%B6%E8%BF%87%E7%A8%8B" \t "https://baike.baidu.com/item/RUP/_blank)是一个[面向对象](https://baike.baidu.com/item/%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1/2262089" \t "https://baike.baidu.com/item/RUP/_blank)且基于网络的程序开发方法论。

软件统一过程（RUP）是Rational软件公司（Rational公司被IBM并购）创造的**软件工程方法** [1]  。RUP描述了如何有效地利用商业的可靠的方法开发和部署软件，是一种重量级过程（也被称作厚方法学），因此特别适用于大型软件团队开发大型项目。

RUP最重要的它有三大特点：1）软件开发是一个迭代过程，2）软件开发是由Use Case驱动的，3）软件开发是以架构设计（Architectural Design）为中心的。 [2]

**六大经验编辑**

**迭代式开发**

**在软件开发的早期阶段就想完全、准确的捕获用户的需求几乎是不可能的。实际上，我们经常遇到的问题是需求在整个软件开发工程中经常会改变。迭代式开发允许在每次迭代过程中需求可能有变化，通过不断细化来加深对问题的理解。迭代式开发不仅可以降低项目的风险，而且每个迭代过程都以可执行版本结束，可以鼓舞开发人员。**

**管理需求**

**确定系统的需求是一个连续的过程，开发人员在开发系统之前不可能完全详细的说明一个系统的真正需求。RUP描述了如何提取、组织系统的功能和约束条件并将其文档化，用例和脚本的使用已被证明是捕获功能性需求的有效方法。**

**体系结构**

**组件使重用成为可能，系统可以由组件组成。基于独立的、可替换的、模块化组件的体系结构有助于降低管理复杂性，提高重用率。RUP描述了如何设计一个有弹性的、能适应变化的、易于理解的、有助于重用的软件体系结构。**

**可视化建模**

**RUP往往和UML联系在一起，对软件系统建立可视化模型帮助人们提供管理软件复杂性的能力。RUP告诉我们如何可视化的对软件系统建模，获取有关体系结构于组件的结构和行为信息。**

**验证软件质量**

**在RUP中软件质量评估不再是事后进行或单独小组进行的分离活动，而是内建于过程中的所有活动，这样可以及早发现软件中的缺陷。**

**控制软件变更**

**迭代式开发中如果没有严格的控制和协调，整个软件开发过程很快就陷入混乱之中，RUP描述了如何控制、跟踪、监控、修改以确保成功的迭代开发。RUP通过软件开发过程中的制品，隔离来自其他工作空间的变更，以此为每个开发人员建立安全的工作空间。**

**二维开发模型**

**RUP软件开发生命周期是一个二维的软件开发模型。横轴通过时间组织，是过程展开的生命周期特征，体现开发过程的动态结构，用来描述它的术语主要包括周期(Cycle）、阶段（Phase）、迭代（Iteration）和里程碑(Milestone）；纵轴以内容来组织为自然的逻辑活动，体现开发过程的静态结构，用来描述它的术语主要包括活动（Activity）、产物（Artifact）、工作者（Worker）和工作流（Workflow）。如图1：**

**敏捷软件开发**（英语：Agile software development），又称**敏捷开发**，是一种从1990年代开始逐渐引起广泛关注的新型[软件开发](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%BC%80%E5%8F%91" \t "https://baike.baidu.com/item/_blank)方法，是一种能应对快速变化需求的软件开发能力。它们的具体名称、理念、过程、术语都不尽相同，相对于“非敏捷”，更强调程序员团队与业务专家之间的紧密协作、面对面的沟通（认为比书面的文档更有效）、频繁交付新的软件版本、紧凑而自我组织型的团队、能够很好地适应需求变化的代码编写和团队组织方法，也更注重软件开发过程中人的作用。

敏捷开发是一种从1990年代开始逐渐引起广泛关注的新型[软件开发方法](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%BC%80%E5%8F%91%E6%96%B9%E6%B3%95" \t "https://baike.baidu.com/item/_blank)，是一种能应对快速变化需求的软件开发能力。它们的具体名称、理念、过程、术语都不尽相同，相对于"非敏捷"，更强调程序员团队与业务专家之间的紧密协作、面对面的沟通（认为比书面的文档更有效）、频繁交付新的软件版本、紧凑而自我组织型的团队、能够很好地适应需求变化的代码编写和团队组织方法，也更注重做为软件开发中人的作用。

敏捷软件开发描述了一套[软件开发](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%BC%80%E5%8F%91" \t "https://baike.baidu.com/item/_blank)的价值和原则，在这些开发中，需求和解决方案皆通过自组织跨功能团队达成。敏捷软件开发主张适度的计划、进化开发、提前交付与持续改进，并且鼓励快速与灵活的面对开发与变更。这些原则支援许多[软件开发方法](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%BC%80%E5%8F%91%E6%96%B9%E6%B3%95" \t "https://baike.baidu.com/item/_blank)的定义和持续进化。 [1]

极限编程（ExtremeProgramming，简称XP）是由KentBeck在1996年提出的，是一种[软件工程](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%B7%A5%E7%A8%8B/16352442" \t "https://baike.baidu.com/item/_blank)方法学，是[敏捷软件开发](https://baike.baidu.com/item/%E6%95%8F%E6%8D%B7%E8%BD%AF%E4%BB%B6%E5%BC%80%E5%8F%91/7108658" \t "https://baike.baidu.com/item/_blank)中可能是最富有成效的几种方法学之一。如同其他敏捷方法学，极限编程和传统方法学的本质不同在于它更强调可适应性而不是可预测性。

极限编程是一个轻量级的、灵巧的软件开发方法；同时它也是一个非常严谨和周密的方法。它的基础和价值观是交流、朴素、反馈和勇气；即，任何一个软件项目都可以从四个方面入手进行改善：加强交流；从简单做起；寻求反馈；勇于实事求是。

XP是一种近螺旋式的开发方法，它将复杂的开发过程分解为一个个相对比较简单的小周期；通过积极的交流、反馈以及其它一系列的方法，开发人员和客户可以非常清楚开发进度、变化、待解决的问题和潜在的困难等，并根据实际情况及时地调整开发过程。

极限编程的主要目标在于降低因需求变更而带来的成本。在传统系统开发方法中，[系统需求](https://baike.baidu.com/item/%E7%B3%BB%E7%BB%9F%E9%9C%80%E6%B1%82/5439123" \t "https://baike.baidu.com/item/_blank)是在项目开发的开始阶段就确定下来，并在之后的开发过程中保持不变的。这意味着项目开发进入到之后的阶段时出现的需求变更（而这样的需求变更在一些发展极快的领域中是不可避免的）将导致开发成本急速增加。

极限编程透过引入基本价值、原则、方法等概念来达到降低变更成本的目的。一个应用了极限编程方法的系统开发项目在应对需求变更时将显得更为灵活。

极限编程方法的基本特征是：

增量和反复式的开发----一次小的改进跟着一个小的改进。

反复性，通常是自动重复的单元测试，回归测试。参见JUnit。

结对程序设计

在程序设计团队中的用户交互（在场的客户）

软件重构

共享的代码所有权

简单

反馈

用隐喻来组织系统

可以忍受的速度

--------------------------------------------------------------------------------------------------------------------------------------

用例图（Use Case Diagram）:用例图显示谁是相关的用户，用户希望系统提供什么服务（用例），以及用例之间的关系图。用例图主要的作用是获取需求、指导测试。   
  
用例图的4个基本组件：参与者(Actor)、用例(Use Case)、关系(Relationship)和系统。   
泛化(generalization)：泛化关系是一种继承关系，子用例将继承基用例的所有行为，关系和通信关系，也就是说在任何使用基用例的地方都可以用子用例来代替。泛化关系在用例图中使用空心的箭头表示，箭头方向从子用例指向基用例。   
  
扩展(extend)： extend关系是对基用例的扩展，基用例是一个完整的用例，即使没有子用例的参与，也可以完成一个完整的功能。extend的基用例中将存在一个扩展点，只有当扩展点被激活时，子用例才会被执行。 extend关系在用例图中使用带箭头的虚线表示(在线上标注<<extend>>)，箭头从子用例指向基用例。   
  
包含(include)： include为包含关系，当两个或多个用例中共用一组相同的动作，这时可以将这组相同的动作抽出来作为一个独立的子用例，供多个基用例所共享。因为子用例被抽出，基用例并非一个完整的用例，所以include关系中的基用例必须和子用例一起使用才够完整，子用例也必然被执行。include关系在用例图中使用带箭头的虚线表示(在线上标注<<include>>)，箭头从基用例指向子用例。

1. 数据结构问哈希的题目，问完基础的之后，老师又延伸问了为什么要有哈希（为了压缩范围啊。。。映射不就是这个原理。。）
2. 第二个操作系统的问题是进程的，我刚说完进程的三种状态
3. 堆的特点是什么，作什么用

栈：容量小，存取速度 快，适合存储生命周期短的数据，栈中存储变量（称为局部变量），特点是先进后出，使用完后就会销毁  
堆：容量大，存取速度慢，适合存储生命周期长的数据，堆适合存储对象，可以取出任意位置的数据

1. 栈在函数调用起什么作用

函数调用的局部状态之所以用栈来记录是因为这些数据的存活时间满足“后入先出”（LIFO）顺序，而栈的基本操作正好就是支持这种顺序的访问。可以看到，函数的调用有完美的嵌套关系——调用者的生命期总是长于被调用者的生命期，并且后者在前者的之内。  
这样，被调用者的局部信息所占空间的分配总是后于调用者的（后入），而其释放则总是先于调用者的（先出），所以正好可以满足栈的LIFO顺序，选用栈这种数据结构来实现调用栈是一种很自然的选择。

1. 二分查找是否适用于链表，哪种数据结构查找最快，二叉排序树的最坏和平均复杂度及适用情况
2. 最短路径等等及一些延伸。

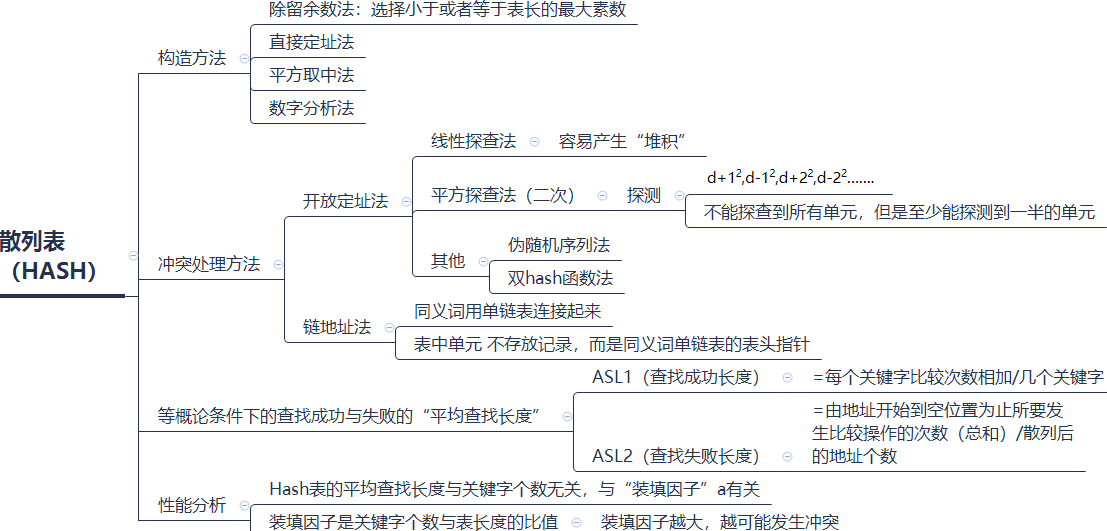
259.数据结构满二叉树高度怎么算

260.计网，滑动窗口协议，我讲了下流量控制背景，开始说停等

261.什么是中断

中断指当出现需要时，CPU暂时停止当前程序的执行转而执行处理新情况的程序和执行过程

1. 什么是哈希冲突？如何解决哈希冲突？



263、（TCP/IP协议簇）传输层有哪些协议？它们有什么区别？

264、软实时和硬实时的区别。

265.二分查找有什么特点

1. 网络服务质量qos.一知半解的讲了一下不理想

QoS（Quality of Service，[服务质量](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E8%B4%A8%E9%87%8F/9401950" \t "https://baike.baidu.com/item/qos/_blank)）指一个网络能够利用各种基础技术，为指定的[网络通信](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E9%80%9A%E4%BF%A1" \t "https://baike.baidu.com/item/qos/_blank)提供更好的服务能力, 是网络的一种安全机制， 是用来解决网络延迟和阻塞等问题的一种技术。 在正常情况下，如果网络只用于特定的无时间限制的应用系统，并不需要QoS，比如[Web](https://baike.baidu.com/item/Web" \t "https://baike.baidu.com/item/qos/_blank)应用，或[E-mail](https://baike.baidu.com/item/E-mail" \t "https://baike.baidu.com/item/qos/_blank)设置等。但是对关键应用和多媒体应用就十分必要。当网络过载或拥塞时，QoS 能确保重要业务量不受[延迟](https://baike.baidu.com/item/%E5%BB%B6%E8%BF%9F/5950826" \t "https://baike.baidu.com/item/qos/_blank)或[丢弃](https://baike.baidu.com/item/%E4%B8%A2%E5%BC%83/10248074" \t "https://baike.baidu.com/item/qos/_blank)，同时保证网络的高效运行。

QoS应运而生。支持QoS功能的设备，能够提供传输品质服务；针对某种类别的数据流，可以为它赋予某个级别的传输优先级，来标识它的相对重要性，并使用设备所提供的各种优先级转发策略、拥塞避免等机制为这些数据流提供特殊的传输服务。配置了QoS的[网络环境](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E7%8E%AF%E5%A2%83" \t "https://baike.baidu.com/item/qos/_blank)，增加了网络性能的可预知性，并能够有效地分配网络[带宽](https://baike.baidu.com/item/%E5%B8%A6%E5%AE%BD" \t "https://baike.baidu.com/item/qos/_blank)，更加合理地利用网络资源。

QoS的关键指标主要包括：可用性、[吞吐量](https://baike.baidu.com/item/%E5%90%9E%E5%90%90%E9%87%8F" \t "https://baike.baidu.com/item/qos/_blank)、时延、时延变化(包括抖动和漂移)和丢失。

用户需要与网络系统进行协商。协商就是用户之间/用户与网络之间就QoS要求进行交互，最后确定QoS的过程。用户可以表达的QoS描述包括：

* 信息流特征。例如，信息流产生的峰值速率和平均速率等，以及五元组等网络数据。体现用户让网络知晓自己的特定的流量特征，需要QoS服务。
* 信息流的性能要求、同步要求。如吞吐量、延迟、抖动、丢包率的各种要求。
* 服务层次。例如想要可控负载型服务、保证性服务、还是尽力而为服务。

当用户与网络系统达成QoS约定之后，网络系统就要提供基于QoS信息的流量实时控制了

1. 哈弗曼树
2. 数据库的三个模式
3. 加问了事务的概念
4. 做过哪些项目
5. 用什么平台
6. java的开发模式。

总体来说设计模式分为三大类：

创建型模式，共五种：工厂方法模式、抽象工厂模式、单例模式、建造者模式、原型模式。

结构型模式，共七种：适配器模式、装饰器模式、代理模式、外观模式、桥接模式、组合模式、享元模式。

行为型模式，共十一种：策略模式、模板方法模式、观察者模式、迭代子模式、责任链模式、命令模式、备忘录模式、状态模式、访问者模式、中介者模式、解释器模式。

其实还有两类：并发型模式和线程池模式。

**二、设计模式的六大原则**

**1、开闭原则（Open Close Principle）**

开闭原则就是说**对扩展开放，对修改关闭**。在程序需要进行拓展的时候，不能去修改原有的代码，实现一个热插拔的效果。所以一句话概括就是：为了使程序的扩展性好，易于维护和升级。想要达到这样的效果，我们需要使用接口和抽象类，后面的具体设计中我们会提到这点。

**2、里氏代换原则（Liskov Substitution Principle）**

里氏代换原则(Liskov Substitution Principle LSP)面向对象设计的基本原则之一。 里氏代换原则中说，任何基类可以出现的地方，子类一定可以出现。 LSP是继承复用的基石，只有当衍生类可以替换掉基类，软件单位的功能不受到影响时，基类才能真正被复用，而衍生类也能够在基类的基础上增加新的行为。里氏代换原则是对“开-闭”原则的补充。实现“开-闭”原则的关键步骤就是抽象化。而基类与子类的继承关系就是抽象化的具体实现，所以里氏代换原则是对实现抽象化的具体步骤的规范。—— From Baidu 百科

**3、依赖倒转原则（Dependence Inversion Principle）**

这个是开闭原则的基础，具体内容：真对接口编程，依赖于抽象而不依赖于具体。

**4、接口隔离原则（Interface Segregation Principle）**

这个原则的意思是：使用多个隔离的接口，比使用单个接口要好。还是一个降低类之间的耦合度的意思，从这儿我们看出，其实设计模式就是一个软件的设计思想，从大型软件[架构](http://lib.csdn.net/base/architecture" \o "大型网站架构知识库" \t "https://www.cnblogs.com/wangzhongqiu/p/_blank)出发，为了升级和维护方便。所以上文中多次出现：降低依赖，降低耦合。

**5、迪米特法则（最少知道原则）（Demeter Principle）**

为什么叫最少知道原则，就是说：一个实体应当尽量少的与其他实体之间发生相互作用，使得系统功能模块相对独立。

**6、合成复用原则（Composite Reuse Principle）**

原则是尽量使用合成/聚合的方式，而不是使用继承。

一．工厂模式【普通工厂模式，】

多个类实现一个抽象接口，一个工厂根据条件返回一个特定的实例。

1. 单例模式

数据库Connection只有一个

1. 适配器模式

 适配器模式将某个类的接口转换成客户端期望的另一个接口表示，目的是消除由于接口不匹配所造成的类的兼容性问题

1. 数据库的锁机制

    数据库的大并发要考虑锁和锁的性能问题，加锁是为了实现并发控制。数据库是一个多用户资源，

    若对并发控制不加控制会读取和存储不正确的数据，破坏数据的一致性（脏读，不可重复读，幻读等）可能会产生死锁。

    锁机制保证在一个事务释放锁之前其他事务不可以进行修改。

    锁：行级锁，表级锁，悲观锁，乐观锁

    悲观锁：事务每次操作之前假设有其他事务会修改需访问的数据，会要求上锁。

    乐观锁：事务每次操作之前假设没有其他事务会修改需访问的数据，不会要求上锁。

    共享锁：对某一资源加共享锁，自身可以读该资源，其他人也可以读该资源

    共享锁（S锁）：如果事务T对数据A加上共享锁后，则其他事务只能对A再加共享锁，不能加排他锁。获准共享锁的

    事务只能读数据，不能修改数据。

    排他锁（X锁）：如果事务T对数据A加上排他锁后，则其他事务不能再对A加任任何类型的封锁。获准排他锁的事务

    既能读数据，又能修改数据。

共享锁下其它用户可以并发读取，查询数据。但不能修改，增加，删除数据。资源共享。

1. 数据库

1、触发器的作用？

触发器是一中特殊的存储过程，主要是通过事件来触发而被执行的。它可以强化约束，来维护数据的完整性和一致性，可以跟踪数据库内的操作从而不允许未经许可的更新和变化。可以联级运算。如，某表上的触发器上包含对另一个表的数据操作，而该操作又会导致该表触发器被触发。

2、什么是存储过程？用什么来调用？

存储过程是一个预编译的SQL语句，优点是允许模块化的设计，就是说只需创建一次，以后在该程序中就可以调用多次。如果某次操作需要执行多次SQL，使用存储过程比单纯SQL语句执行要快。 调用： 1）可以用一个命令对象来调用存储过程。 2）可以供外部程序调用，比如：java程序。

3、存储过程的优缺点？

优点： 1）存储过程是预编译过的，执行效率高。 2）存储过程的代码直接存放于数据库中，通过存储过程名直接调用，减少网络通讯。 3）安全性高，执行存储过程需要有一定权限的用户。 4）存储过程可以重复使用，可减少数据库开发人员的工作量。 缺点：移植性差

5、索引的作用？和它的优点缺点是什么？

索引就一种特殊的查询表，数据库的搜索可以利用它加速对数据的检索。它很类似与现实生活中书的目录，不需要查询整本书内容就可以找到想要的数据。索引可以是唯一的，创建索引允许指定单个列或者是多个列。缺点是它减慢了数据录入的速度，同时也增加了数据库的尺寸大小。

6、什么样的字段适合建索引

唯一、不为空、经常被查询的字段

9、什么叫视图？游标是什么？

视图：是一种虚拟的表，具有和物理表相同的功能。可以对视图进行增，改，查，操作，试图通常是有一个表或者多个表的行或列的子集。对视图的修改会影响基本表。它使得我们获取数据更容易，相比多表查询。 游标：是对查询出来的结果集作为一个单元来有效的处理。游标可以定在该单元中的特定行，从结果集的当前行检索一行或多行。可以对结果集当前行做修改。一般不使用游标，但是需要逐条处理数据的时候，游标显得十分重要。

10、视图的优缺点

优点： 1）对数据库的访问，因为视图可以有选择性的选取数据库里的一部分。 2 ）用户通过简单的查询可以从复杂查询中得到结果。 3 ）维护数据的独立性，试图可从多个表检索数据。 4 ）对于相同的数据可产生不同的视图。 缺点： 性能：查询视图时，必须把视图的查询转化成对基本表的查询，如果这个视图是由一个复杂的多表查询所定义，那么，那么就无法更改数据

11、列举几种表连接方式,有什么区别？

内连接、自连接、外连接（左、右、全）、交叉连接 内连接：只有两个元素表相匹配的才能在结果集中显示。 外连接： 左外连接:左边为驱动表，驱动表的数据全部显示，匹配表的不匹配的不会显示。 右外连接:右边为驱动表，驱动表的数据全部显示，匹配表的不匹配的不会显示。 全外连接：连接的表中不匹配的数据全部会显示出来。 交叉连接： 笛卡尔效应，显示的结果是链接表数的乘积。