School of Software Engineering
University of Science and Technology of China

2012 年参加中国科大软件学院复试前，收集了一些前几届的面试题目。

和大家分享一下，希望对大家有所帮助。

独上高楼，望尽天涯路。

衣带渐宽终不悔，为伊消的人憔悴。

众里寻她千百度，蓦然回首，伊人却在灯火阑珊处。

turinglife
2013-03-09

1. 好多人会问到时间复杂度

按数量级递增排列，常见的时间复杂度有：

常数阶 O(1)，对数阶 O(log2n)，线性阶 O(n)，
线性对数阶 O(nlog2n)，平方阶 O(n^2)，立方阶 O(n^3)，
k 次方阶 O(n^k),，指数阶 O(2^n) 。

随着**问题规模 n** 的不断增大，上述时间复杂度不断增大，算法的执行效率越低。

2. 各种排序的时间复杂度和性能比较

排序类别	基本思路	详细	时 间 复 杂	空 间 复 杂	特点	注意

		分类	度	度		
插入排序	每一趟将一个待排序的元素，按其关键字值的大小插入到已经排序的有序区中的适当位置上，直到全部插入完成。	直接插入排序	最好（正序）：O(n) 最坏（逆序）：O(n^2) 平均：O(n^2)	O(1)，需要一个监视哨。	稳定排序	直接插入排序所产生的有序区**不一定是全局有序**的，有序区中的关键字并不一定大于或小于无序区中所有元素的关键字，这样每一趟排序并不一定将一个元素放置在最终的位置上。插入排序与待排序数据的顺序有关，当**正序**时**效率最高**，当**反序**时**效率最低**。
		折半插入	平均：O(n^2)	O(1)		折半插入排序仅减少了关键字间的比较次数，而元素的移动次数不变。
		希尔排序	平均：O(n^1.3)	O(1)	不稳定排序	希尔排序每一趟并不产生有序区，也不一定将一个元素放置在最终的位置上，希尔排序和待排序数据的顺序有关，**正序**时**最高**，**逆序**时效率**最低**。
交换排序	基本思想：两两比较待排序元素的关键字，发现两个元素的次序相反时即进行交换，直到没有反序的元素为止。	起泡排序	最坏：O(n2) 平均：O(n2)	O(1)	稳定排序	起泡排序中所产生的有序区一定是全局有序的，有序区中的所有元素的关键字一定大于或小于无序区中所有元素的关键字，每一趟排序都将一个元素放置到最终位置上，起泡排序与待排序数据的顺序有关，**正序**效率**最高**，**逆序**效率**最低**。
		快速排序：在待排序的 n 个元素中任取一个元素（通常取第一个元素）作为基准，把该元素放入最终的位置上（归为一个元素），数据序列被此元素划分成两部分：所有关键字比该元素关键字小的元素放置在前一部分，所有比他大的元素放置在后一部分，这个过程称为一趟快速排序，以后对所有的两部分分别重复上述过程，直至每部分内只有一个元素或空为止。			不稳定排序	在快速排序算法中，并不产生有序区，但每一趟归位一个元素，快速排序与待排序数据的顺序有关，当**正序**和**逆序**时**效率都低**，只有当数据随机分布，每次划分的子区间长度大致相等时效率最高。
选择排序	基本思想：每一趟从	简单选择排序：在无序区中选择关键字最小的元素	O(1)	$O(n^2)$	不稳定排	有序区全局有序，有序区中的所有元素关键字要么全部大于

	待排序的元素序列中选出关键字最大（或最小）的元素，按顺序放在已排序的元素序列的最后面（或最前面），直到全部拍完为止。	放在有序区的最后，形成新的有序区和新的无序区。			序	无序区，要么全部小于无序区，每趟排序归为一个元素，时间复杂度与待排序数据序列的**顺序无关**。
		堆排序： 一种树形选择排序。在排序过程中，将R[1…n]看成是完全二叉树的顺序存储结构，利用完全二叉树中的双亲和孩子结点之间的关系来找到当前序列中关键最大/小的元素。	O(1)	$O(n\log_2^n)$	**不稳定**排序	建初始堆所需要的比较次数较多，所以堆排序不适宜于元素较少的表。 所产生的**有序区一定全局有序**，有序区要么全部大于或小于无序区中的关键字，每趟排序归为一个元素，时间复杂度与**待排序数据顺序无关**。
归并排序						
基数排序						

各种排序方法的综合比较

一.时间性能

1.按平均的时间性能来分，有三类排序方法：

时间复杂度为 O(nlogn)的方法有：快速排序、堆排序和归并排序，其中以快速排序为最好；

直接插入算法

```
00064: /*************************************************************
00065: Function Name: InsertSort
00066: Description: 直接插入排序，按照非递减排列。
00067: Parameter: SqList *pSqList
00068: Creater: turinglife@gmail.com
00069: Time: 18:24 2010-7-14
00070: modify:
00071:
00072: 18:12 2012-04-08 添加注释，pElem[0]不存储有效数据，仅仅被当做监视哨，
00073:                  这样做，程序的效率比较高。需要进行排序的数据从
00074:                  pElem[1]开始存储数据。
00075:
00076: *************************************************************/
00077: void InsertSort(SqList *pSqList)
00078: {
00079:     int i, j;
00080:
00081:     for(i = 2; i <= pSqList->iLength; i++)
00082:     {
00083:         // 待插入的元素值小于已经有序的序列的最后一个值
00084:         if(pSqList->pElem[i] < pSqList->pElem[i - 1])
00085:         {
00086:             // 将i元素放入监视哨位置
00087:             pSqList->pElem[0] = pSqList->pElem[i];
00088:
00089:             // 从i-1开始，寻找插入点，若大于i-1的值，则直接将i值插入到i-1的位置，不需要移动元素。
00090:             // 若小于i-1的值，就需要将大于i的值，依次往后移动，直到找到插入位置。
00091:             for(j = i - 1; pSqList->pElem[0] < pSqList->pElem[j]; --j)
00092:             {
00093:                 pSqList->pElem[j + 1] = pSqList->pElem[j];
00094:             }
00095:
00096:             pSqList->pElem[j + 1] = pSqList->pElem[0];
00097:         }
00098:     }
00099: } ? end InsertSort ?
```

折半插入算法

```
00136: /*****************************************************************
00137: Function Name: BInsertSort
00138: Description: 折半插入排序，按照非递减排列。Binary Insert Sort
00139: Parameter: SqList *pSqList
00140: Creater: turinglife@gmail.com
00141: Time 18:48 2010-7-14
00142: Modify:
00143:
00144: 19:22 2012-04-08 pElem[0]被用作监视哨，并不放置待排序数据。
00145:
00146: *****************************************************************/
00147: void BInsertSort(SqList *pSqList)
00148: {
00149:     int i, j, low, high, mid;
00150:
00151:     for(i = 2; i <= pSqList->iLength; i++)
00152:     {
00153:         pSqList->pElem[0] = pSqList->pElem[i];
00154:
00155:         low = 1;
00156:         high = i - 1;
00157:
00158:         // 利用折半查找，查找插入点。
00159:         while(low <= high)
00160:         {
00161:             mid = (low + high) / 2;
00162:
00163:             if(pSqList->pElem[mid] > pSqList->pElem[0])
00164:                 high = mid - 1;
00165:             else
00166:                 low = mid + 1;
00167:         }
00168:
00169:         // 记录后移
00170:         for(j = i - 1; j >= (high + 1); --j)
00171:             pSqList->pElem[j+1] = pSqList->pElem[j];
00172:
00173:         // 插入正确位置
00174:         pSqList->pElem[high + 1] = pSqList->pElem[0];
00175:
00176:     } ? end for i=2;i<=pSqList->iLeng... ?
00177:
00178:     return;
00179: } ? end BInsertSort ?
```

3. 什么叫堆排序？与快速排序有神马不同？

答案请参加上面

4. 循环队列的顺序表示中，为什么要空一个位置。。。

队列那个为了辨别对空和队满

5. 什么是二叉查找树，原理

二叉排序树（Binary Sort Tree）又称二叉查找树。它或者是一棵空树；或者是具有下列性质的二叉树：（1）若左子树不空，则左子树上所有结点的值均小于它的根结点的值；（2）若右子树不空，则右子树上所有结点的值均大于它的根结点的值；（3）左、右子树也分别为二叉排序树；

步骤：若根结点的关键字值等于查找的关键字，成功。
否则，若小于根结点的关键字值，递归查左子树。
若大于根结点的关键字值，递归查右子树。
若子树为空，查找不成功。

6. 排序算法最优的时间复杂度

O(logn)

7. 哈夫曼树

给定 n 个权值作为 n 个叶子结点，构造一棵二叉树，若**带权路径长度**达到**最小**，称这样的二叉树为**最优二叉树**，也称为**哈夫曼树(Huffman tree)**。

构造方法：

假设有 n 个权值，则构造出的哈夫曼树有 n 个叶子结点。 n 个权值分别设为 w1、w2、…、wn，则哈夫曼树的构造规则为：

(1) 将 w1、w2、…，wn 看成是有 n 棵树的森林(每棵树仅有一个结点)；

(2) 在森林中选出两个根结点的权值最小的树合并，作为一棵新树的左、右子树，且新树的根结点权值为其左、右子树根结点权值之和；

(3)从森林中删除选取的两棵树，并将新树加入森林；

(4)重复(2)、(3)步，直到森林中只剩一棵树为止，该树即为所求得的哈夫曼树。[2]

8. 什么是哈希冲突，及如何解决

哈希表：散列表（Hash table，也叫哈希表），是根据关键码值(Key value)而直接进行访问的数据结构。也就是说，它通过把关键码值映射到表中一个位置来访问记录，以加快查找的速度。这个映射函数叫做**散列函数**，存放记录的数组叫做散列表。

常用散列函数：

1、 直接寻址法
2、 数字分析法
3、 平方取中法
4、 折叠法
5、 随机数法

6、 除留余数法

处理冲突的方法：

1、 开放寻址法
2、 再散列法
3、 链地址法
4、 建立一个公共溢出区

散列因子：

散列表的装填因子定义为：α= 填入表中的元素个数 / 散列表的长度

α是散列表装满程度的标志因子。由于表长是定值，α与“填入表中的元素个数”成正比，所以，α越大，填入表中的元素较多，产生冲突的可能性就越大；α越小，填入表中的元素较少，产生冲突的可能性就越小。

9. 深度、广度搜索的过程

图的深度优先遍历：

假设给定图 G 的初态是所有顶点均未曾访问过。在 G 中任选一顶点 v 为初始出发点(源点)，则深度优先遍历可定义如下：首先访问出发点 v，并将其标记为已访问过；然后依次从 v 出发搜索 v 的每个邻接点 w。若 w 未曾访问过，则以 w 为新的出发点继续进行深度优先遍历，直至图中所有和源点 v 有路径相通的顶点(亦称为从源点可达的顶点)均已被访问为止。若此时图中仍有未访问的顶点，则另选一个尚未访问的顶点作为新的源点重复上述过程，直至图中所有顶点均已被访问为止。

图的深度优先遍历类似于树的前序遍历。采用的搜索方法的**特点**是尽可能先对纵深方向进行搜索。这种搜索方法称为深度优先搜索(Depth-First Search)。相应地，用此方法遍历图就很自然地称之为图的深度优先遍历。

使用数据结构：堆栈

图的广度优先遍历：

设 x 是当前被访问顶点，在对 x 做过访问标记后，选择一条从 x 出发的未检测过的边(x，y)。若发现顶点 y 已访问过，则重新选择另一条从 x 出发的未检测过的边，否则沿边(x，y)到达未曾访问过的 y，对 y 访问并将其标记为已访问过；然后从 y 开始搜索，直到搜索完从 y 出发的所有路径，即访问完所有从 y 出发可达的顶点之后，才回溯到顶点 x，并且再选择一条从 x 出发的未检测过的边。上述过程直至从 x 出发的所有边都已检测过为止。此时，若 x 不是源点，则回溯到在 x 之前被访问过的顶点；否则图中所有和源点有路径相通的顶点(即从源点可达的所有顶点)都已被访问过，若图 G 是连通图，则遍历过程结束，否则继续选择一个尚未被访问的顶点作为新源点，进行新的搜索过程。

使用数据结构：队列

10. 迪杰斯克拉算法的过程

Dijkstra(迪杰斯特拉)算法是典型的**单源最短路径算法**，用于计算一个节点到其他所有节点的最短路径。**主要特点**是以起始点为中心向外层层扩展，直到扩展到终点为止。Dijkstra 算法是很有代表性的最短路径算法，在很多专业课程中都作为基本内容有详细的介绍，如数据结构，图论，运筹学等等。Dijkstra 一般的表述通常有两种方式，一种用永久和临时标号方式，一种是用 OPEN, CLOSE 表的方式，这里均采用永久和临时标号的方式。注意该算法要求图中不存在负权边。

算法步骤如下：

1、初使时令 S={V0},T={其余顶点}，T 中顶点对应的距离值若存在<V0,Vi>，d(V0,Vi)为<V0,Vi>弧上的权值若不存在<V0,Vi>，d(V0,Vi)为∝

2、从 T 中选取一个其距离值为最小的顶点 W 且不在 S 中，加入 S

3、对 T 中顶点的距离值进行修改：若加进 W 作中间顶点，从 V0 到 Vi 的距离值比不加 W 的路径要短，则修改此距离值重复上述步骤 2、3，直到 S 中包含所有顶点，即 S=T 为止

11. 链表查询某个元素，平均时间复杂度是多少？

O(n)

12. 图的存储方式

图**没有**顺序映像的存储结构

1、 **邻接矩阵**：用两个数组分别存储**数据元素（顶点）的信息**和数据元素之间的**关系**（边或弧）的信息。图的邻接矩阵表示是**唯一**的，且无向图的邻接矩阵一定是一个对称矩阵。
2、 **邻接表**：是图的链式存储结构，
3、 **十字链表**：有向图的另一种链式存储结构
4、 **邻接多重表**：无向图的链式存储结构

13. 图的深度遍历是否唯一

不唯一

14. 图相关概念

有向图	无向图
弧，弧头，弧尾	**边**
有向完全图：n*(n-1)条弧的有向图	**完全图**：1/2*n*(n-1)条边的无向图
稀疏图：很少条边或弧（如 e<nlogn）的图 **稠密图**：e>nlogn **权**：有时图的边或弧具有与它相关的数，这种与图的边或弧相关的数叫做**权** **网**：带权图称为网 **子图**：假设有两个图 G=(V, {E})和 G'=(V', {E'})，如果 V'是 V 的子集，E'是 E 的子集，则称 G'为 G 的子图。 **路径**：顶点序列，路径的长度就是路径上的边或弧的数目。 **回路/环**：第一个顶点和最后一个顶点相同的路径称为回路/环 **简单路径**：序列中顶点不重复出现的路径称为简单路径 **简单回路/简单环**：除了第一个顶点和最后一个顶点之外，其余顶点不重复出现的回路，称为简单回路/简单环。	
出度： **入度**：	**度**：
	连通：无向图中从顶点 A 到顶点 B 有路径，则称两个顶点连通。
强连通图：有向图中任意两个不同顶点 A、B，从顶点 A 到顶点 B 和从顶点 B 到顶点 A 都存在路径，则称为强连通图。	**连同图**：图中任意两个顶点都是连通的，则称为连通图。
强连通分量：有向图中的极大强连通子图	**连通分量**：无向图中的**极大**连通子图
生成森林：一个有向图的生成森林由若干棵有向树组成，含有图中全部顶点，但只有足以构成若干棵不相交的有向树的弧。	**生成树**：极小连通子图，含有图中全部顶点，但只有足以构成一棵树的（n-1）条边。如果在一棵生成树上添加一条边，必定构成一个环。
有向树：如果一个有向图恰好有一个顶点的入度为 0，其余顶点的入度均为 1，则是一棵有向树。	

15. 连通图的概念

在图论中，连通图基于连通的概念。在一个无向图 G 中，若从顶点 vi 到顶点 vj 有路径相连(当然从 vj 到 vi 也一定有路径)，则称 vi 和 vj 是连通的。如果 G 是有向图，那么连接 vi 和 vj 的路径中所有的边都必须同向。**如果图中任意两点都是连通的，那么图被称作连通图。**图的连通性是图的基本性质。

16. 解释下最小生成树

极小连通子图，含有图中全部顶点，但只有足以构成一棵树的（n-1）条边。如果在一棵生成树上添加一条边，必定构成一个环。

17. n 个节点的图的最小生成树有几个节点，几条边

n 个顶点，n-1 条边。

18. 平衡二叉树

平衡二叉树（Balanced Binary Tree）又被称为 AVL 树（有别于 AVL 算法），且具有以下性质：它是一 棵空树或它的左右两个子树的高度差的绝对值不超过 1(-1,0,1)，并且左右两个子树都是一棵平衡二叉树。

AVL 树是最先发明的自平衡二叉查找树。AVL 树得名于它的发明者 G.M. Adelson-Velsky 和 E.M. Landis，他们在 1962 年的论文 "An algorithm for the organization of information" 中发表了它。

查找、插入和删除在平均和最坏情况下都是 O（logn）。增加和删除可能需要通过一次或多次树旋转来重新平衡这个树。

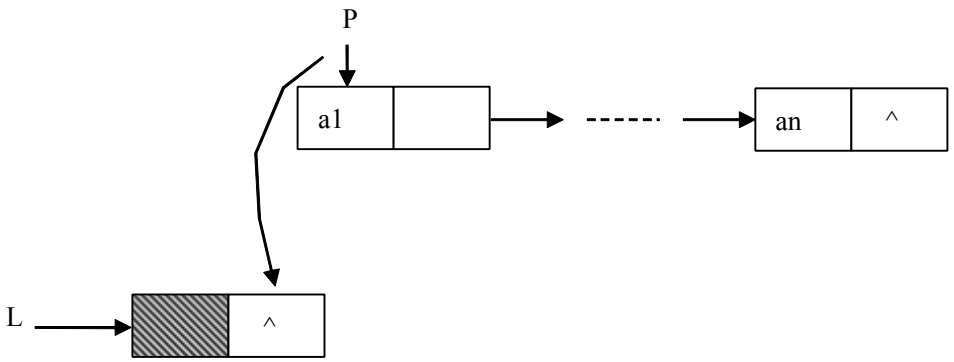
19. 二叉树怎么存储

1、**顺序存储结构**：用一组地址连续的存储单元依次自上而下、自左向右存储完全二叉树上的结点信息。这种顺序存储结构仅适用于完全二叉树。否则其他形式的二叉树用顺序存储，会浪费不少存储空间。
2、**链式存储结构**：二叉链表、三叉链表。含有 n 个结点的二叉链表中有 n+1 个空链域。

20. 单链表就地逆置

所谓 “**就地**”是指算法的辅助空间为 O(1)

思路：用指针 p 扫描原单链表，先将头节点 L 的 next 域设置为 NULL 而变成一个空链表，然后将*p 节点采用头插法插入到 L 中。



算法：

```
00281: /*****************************************************************
00282: Function Name: reverse
00283: Description: 单链表就地逆置算法，采用头插法重建单链表。
00284:              所谓“就地”是指算法的辅助空间为O(1)
00285: Parameter:
00286: Creater: turinglife@gmail.com
00287: Time 14:36 2010-4-14
00288: Modify:
00289: *****************************************************************/
00290: void reverse(LinkList L)
00291: {
00292:     LinkList *p = L->next, *q;
00293:
00294:     L->next = NULL;
00295:
00296:     while(p != NULL)
00297:     {
00298:         q = p->next;
00299:         p->next = L->next;
00300:         L->next = p;
00301:         p = q;
00302:     }
00303: }
```

21. 各种查找总结

动态查找表：若在查找的同时还对表做修改运算（如插入或删除），则相应的表称之为动态查找表。
静态查找表：反之称为静态查找表

查找类型	平均查找长度	时间复杂度	适用存储结构	原理
顺序查找（静态查找）	成功：(n+1)/2 失败：n	O(n)	线性表的**顺序存储**和**链式存储**都可以	从表的一端开始，顺序扫描线性表，依次将扫描到的关

				键字和给定值 k 相比较，若当前扫描到记录的关键字与 k 相等，则查找成功；若扫描结束后，仍未找到关键字等于 k 的记录，则查找失败。
折半查找（静态查找）	成功：查找过程恰好走了一条从判定树的根到被查记录的路径，经历比较的关键字次数恰为该记录在树中的层次。 失败：比较过程经理了一条从判定树根到某个外部节点的路径，所需关键字比较次数不超过判定树的高度。	在最坏的情况下查找成功和查找不成功的比较次数均不超过判定树的高度，即 ⌈log2(n+1)⌉ O(log2n)	**顺序存储结构**且要求元素按关键字**有序排列**	首先用要查找的关键字 k 与中间位置的节点的关键字相比较，这个中间记录把线性表分成了两个子表，若比较结果相等则查找成功；若不相等，再根据 k 与该中间记录关键字的比较大小确定下一步查找哪个子表，这样递归进行下去。 **注意**：折半查找的过程可用二叉树来描述，把当前查找区的中间位置上得记录作为根，左子表和右子表中的记录作为根的左子树和右子树，由此得到的二叉树称为描述折半查找的**判定树**或**比较树**
B-树==多路平衡查找树（动态查找） 特点： 1、 所有叶子节点都在同一层，且不带任何信息。 2、 若根节点不是终端节点，则		应该同二叉查找树 O(log2n)		**查找过程**：B-树查找过程类似于二叉查找树上的查找过程，不同的是在每个记录确定向下查找的路径不一定是二路的。因为一个节点内的关键字有序，故节点内可以采用顺序查找或折半查找。

根节点至少有两棵子树。 3、 树中每个节点至多有 m 棵树 4、 所有内部每个节点至少 ⌈m/2⌉棵子树。				**插入过程**：插入过程分两步，**定位**和**插入**。定位过程利用上面的查找算法，插入过程分为**不分裂**和**分裂**两种情况。分裂插入有可能导致 B-树增高一层。 **删除过程**：两大类。第一类在非最底层的非叶子节点上删除关键字。 第二类在最底层非叶子节点删除关键字，这类型又分三种情况。情况一，直接删除关键字，情况二，兄弟够借，情况三，兄弟不够借（合并），情况三有可能导致 B-树减少一层。
B+树		如果是从根节点开始进行随机查找的话，应该同二叉查找树。 O(log2n)		**查找过程**：两种查找方法，第一种，直接从最小关键字开始进行顺序查找。第二种，从 B+树的根节点开始进行随机查找。 **插入过程**：仅在叶子节点中插入关键字 **删除过程**：仅在叶子节点中删除节点

22. m 阶的 B-树和 m 阶的 B+树主要区别

m 阶的 B-树和 m 阶的 B+树主要区别
1、 B+树所有有效数据全在叶子节点，而 B-树所有节点分散在树中，B-树中的关键字不重复。 2、 B+树种有几个关键字就有几个子树，B-树中具有 n 个关键字的节点含有(n+1)棵子树。 3、 B+树有两个指针，根指针和只想最小节点的指针，叶子节点连接成一个不定长的线性链表 4、 B+树中，每个节点（除根节点外）中的关键字个数 n 的取值范围是⌈m/2⌉<=n<=m，根节点 n 的取值范围是 2<=n<=m。B-树中，每个节点（除根节点外的所有最底层非叶子节点）中的关键字取值范围是⌈m/2⌉-1<=n<=m-1，根节点 n 的取值范围是 1<=n<=m-1。 5、 B+树中的所有非叶子节点仅仅起到索引的作用，节点中的每个索引项只包含对应子树的最大关键字和指向该子树的指针，不含有该关键字对应记录的存储地址。而在 B-树中，每个关键字对应记录的存储地址。

23. 折半查找，适用范围和时间复杂度

适用范围：顺序存储结构且要求元素按关键字**有序排列**

时间复杂度：o($\log_2^n$)

24. 计算机和计算器的区别

计算器：具有简单计算功能，有些具有简单存储功能，不能自动工作，而计算机可以通过编程实现程序自动运行。价位便宜。

计算机：高速计算的电子计算机，可进行数值计算，逻辑计算，还具有存储记忆功能。自动化程度高于计算器。

实际上二者还有另一个本质性的区别。计算器使用的是固化的处理程序，只能完成特定的计算任务；而计算机借助操作系统平台和各类应用软硬件，可以无限扩展其应用领域。也就是说，是否具有扩展性是二者的本质区别。

25. 线程/进程空间是什么

线程运行所需要的内存空间，比如线程程序的存储空间，数据空间，运行空间等。

逻辑地址和物理地址之间的切换，是由 CPU 的内存管理单元 MMU 完成。Linux 内核维护每个进程的逻辑地址到物理地址的对照表。当用户空间进程切换时，内核会更新对照表，用户空间也跟随变。

每个进程的用户空间都是相互独立的，把同一个程序同时运行 10 次，会看到 10 个进程使用的线性地址一模一样但物理内存不一样。

26. 硬实时和软实时

在实时操作系统中，系统必须在特定的时间内完成指定的应用，具有较强的“刚性”，而分时操作系统则注重将系统资源平均地分配给各个应用，不太在意各个应用的进度如何，什么时间能够完成。不过，就算是实时操作系统，其“刚性”和“柔性”的程度也有所不同，就好像是系统的“硬度”有所不同，因而有了所谓的“硬实时(hard real-time)”和“软实时 (soft real-time)”。

硬实时系统有一个刚性的、不可改变的时间限制，它不允许任何超出时限的错误。超时错误会带来损害甚至导致系统失败、或者导致系统不能实现它的预期目标。

软实时系统的时限是一个柔性灵活的，它可以容忍偶然的超时错误。失败造成的后果并不严重，例如在网络中仅仅是轻微地降低了系统的吞吐量。

27. 进程和程序的区别

进程：程序的一次执行过程，一个动态的过程。存在生命周期，包括进程的创建、进程运行、进程挂起、进程结束。

程序：代码+数据的一个集合，一个静态的表示。

28. 进程和线程的区别

进程：资源分配和调度的基本单位，进程切换耗费资源比线程切换大。进程有独立的进程地址空间。

线程：线程是进程中的一个实体，是 CPU 调度的基本单位，是比进程更小的能独立运行的基本单位，线程自己不拥有系统资源，只拥有运行中必不可少的资源（如程序计数器、寄存器、栈），在同一个线程内，可以有多个线程，多个线程共享同一个进程的资源，每个线程具有自己的线程栈。线程没有独立的线程地址空间，多个线程共享同一个进程地址空间。

29. 什么是微内核

微内核是一种能够提供必要服务的操作系统内核；其中这些必要的服务包括任务，线程，交互进程通信（IPC，Inter－Process Communication）以及内存管理等等。所有服务（包括设备驱动）在用户模式下运行，而处理这些服务同处理其他的任何一个程序一样。因为每个服务只是在自己的地址空间运行。所以这些服务之间彼此之间都受到了保护。

微内核是提供操作系统核心功能的内核的精简版本，如 Windows

Linux 是一个单内核，也就是说 Linux 内核运行在单独的内核地址空间。不过，Linux 汲取了微内核的精华，其引以为豪的是模块化设计、抢占式内核、支持内核线程已经动态装载内核模块的能力。

30. call 和 return 具体做了哪些工作

call：参数压栈、返回地址压栈、保护现场

return：返回地址、恢复现场

31. 什么是 DMA，什么是中断。DMA 和中断有什么区别？

DMA：Direct Memory Access 直接内存访问，为了实现外设<->CPU<->存储器之间的不足，从而实现<->存储器。CPU 释放总线，由 DMA 控制器管理。

中断：是指在 CPU 正常运行程序时，由于内部/外部事件引起 CPU 暂时停止正在运行的程序，转而去执行请求 CPU 服务的内部事件或外部事件的服务子程序，待该服务子程序处理完毕后又返回到被中止的程序继续运行，这一过程叫做中断。

区别：首先概念功能就不一样，另外在 DMA 的过程中需要用到中断系统。

32. 硬中断和软中断是什么，区别是什么？

软中断：

1、编程异常通常叫做软中断

2、软中断是通讯进程之间用来模拟硬中断的 一种信号通讯方式。

3、中断源发中断请求或软中断信号后,CPU 或接收进程在适当的时机自动进行中断处理或完成软中断信号对应的功能

4、软中断是软件实现的中断,也就是程序运行时其他程序对它的中断;而硬中断是硬件实现的中断,是程序运行时设备对它的中断。

硬中断：

1、硬中断是由外部事件引起的因此具有随机性和突发性；软中断是执行中断指令产生的，无外部施加中断请求信号，因此中断的发生不是随机的而是由程序安排好的。

2、硬中断的中断响应周期，CPU 需要发中断回合信号（NMI 不需要），软中断的中断响应周期，CPU 不需发中断回合信号。

3、硬中断的中断号是由中断控制器提供的（NMI 硬中断中断号系统指定为 02H）；软中断的中断号由指令直接给出，无需使用中断控制器。

4、硬中断是可屏蔽的（NMI 硬中断不可屏蔽），软中断不可屏蔽。

区别：

1、软中断发生的时间是由程序控制的,而硬中断发生的时间是随机的

2、软中断是由程序调用发生的,而硬中断是由外设引发的

3、硬件中断处理程序要确保它能快速地完成它的任务,这样程序执行时才不会等待较长时间

33. 页面置换算法有哪些？什么是 LRU

在地址映射过程中，若在页面中发现所要访问的页面不再内存中，则产生**缺页中断**。当发生缺页中断时操作系统必须在内存选择一个页面将其移出内存，以便为即将调入的页面让出空间。而用来选择淘汰哪一页的规则叫做**页面置换算法**

常见的置换算法有：

01. 最佳置换算法（OPT）（理想置换算法）
02. 先进先出置换算法（FIFO）：
03. 最近最久未使用（LRU）算法
04. Clock 置换算法（LRU 算法的近似实现）
05. 最少使用（LFU）置换算法
06. 工作集算法
07. 工作集时钟算法
08. 老化算法（非常类似 LRU 的有效算法）
09. NRU(最近未使用）算法
10. 第二次机会算法

LRU 是 Least Recently Used 最近最少使用算法。

为了尽量减少与理想算法的差距，产生了各种精妙的算法，最近最少使用页面置换算法便是其中一个。LRU 算法的提出，是基于这样一个事实：在前面几条指令中使用频繁的页面很可能在后面的几条指令中频繁使用。反过来说，已经很久没有使用的页面很可能在未来较长的一段时间内不会被用到。这个，就是著名的**局部性原理** ——比内存速度还要快的 cache，也是基于同样的原理运行的。因此，我们只需要在每次调换时，找到最近最少使用的那个页面调出内存。这就是 LRU 算法的全部内容。

34. 操作系统中磁盘调度算法

磁盘调度的目标是，使磁盘的平均寻道时间最少。

调度算法名称	基本原理	特点
先来先服务（FCFS：First Come First Serve） **仅适用于请求磁盘 IO 的进程数目较少的场合**	最简单的一种磁盘调度算法，根据进程请求访问磁盘的先后次序进行调度。	**优点**：简单，公平，每一个请求的进程都能够得到处理，不会出现长期得不到处理的进程。
		缺点：未对寻道进行优化，致使平均寻道时间可能较长。
最短寻道时间优先(SSTF：Shorted Seek Time First)	该算法选择这样的进程，其要求访问的磁道，与当前磁头所在的磁道距离最近，以使每次的寻道时间最短，但这种算法不能保证平均寻道时间最短	**优点**：较 FCFS 有更好的寻道性能
		缺点：可能导致老进程**饥饿**，可能会出现**磁臂黏着**
扫描算法(SCAN)==**电梯调度算法** **在 SSTF 算法基础上优化而得到，可以防止老进程饥饿。**	该算法不仅考虑到被访问磁道和当前磁头之间的距离，更优先考虑的是磁头当前的移动方向。(应该选取同方向且距离最近的磁道访问。)	**优点**：磁头双向移动，不会产生饥饿，平均寻道时间短。
		缺点：可能会出现**磁臂黏着**
循环扫描算法(CSCAN)	为了减少延迟，CSCAN 规定磁头单向移动，当从里到最外后，磁头立即返回到最里的欲访问磁道，然后继续向外。	**优点**：磁头单向移动，不会产生饥饿，平均寻道时间短。
N-Step-SCAN 算法，对 SCAN 算法的优化。	将磁盘请求队列分成若干个长度为 N 的子队列，磁盘调度将按照 FCFS 依次处理这些子队列，而每处理一个队列时又是按照 SCAN 算法，对一个队	**优点**：无**磁臂黏着**。

	列处理后再处理其他队列，将新请求队列放入新队列。	
FSCAN 算法，对 SCAN 算法的优化。	将请求队列分成两个子队列，将新出现请求磁盘 IO 的进程放入另一个子队列。	**优点**：无**磁臂黏着**。

35. 操作系统中的信号量

信号量是一种实现进程**同步**和**互斥**的工具。

1、 **整型信号量**：所谓整型信号量就是一个用于表示资源个数的整型量
2、 **记录性信号量**：就是用一个结构体实现，里面包含了表示资源个数的整型量和一个等待队列。

Linux 内核代码对 semaphore 的实现

```
struct semaphore {
    raw_spinlock_t lock;
    unsigned int count;
    struct list_head wait_list;
};
```

36. 什么是 Pv 操作

P、V 操作以原语形式实现，信号量的值仅能由这两条原语加以改变。
P 操作相当于**申请资源**，V 操作相当于**释放资源**。

```
00012: struct semaphore
00013: {
00014:     int count;
00015:     struct list_head wait_queue;
00016: };
00017:
00018: void P(semaphore s)
00019: {
00020:     s.count--;
00021:     if(s.count < 0)
00022:     {
00023:         // 阻塞该进程
00024:         // 将该进程放入此信号量的等待队列s.wait_queue中
00025:     }
00026: }
00027:
00028: void V(semaphore s)
00029: {
00030:     s.count++
00031:     if(s.count <= 0)
00032:     {
00033:         // 将等待队列s.wait_queue中第一个进程取出
00034:         // 放入进程就绪队列
00035:     }
00036: }
```

37. 什么是操作系统

操作系统是管理电脑硬件与软件资源的程序，同时也是计算机系统的内核与基石。操作系统是控制其他程序运行，管理系统资源并为用户提供操作界面的系统软件的集合。操作系统身负诸如管理与配置内存、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。操作系统的型态非常多样，不同机器安装的 OS 可从简单到复杂，可从手机的嵌入式系统到超级电脑的大型操作系统。目前微机上常见的操作系统有 DOS、OS/2、UNIX、XENIX、LINUX、Windows、Netware 等。

38. 简述操作系统中系统调用过程

系统调用把应用程序的请求传给内核，调用相应的的内核函数完成所需的处理，将处理结果返回给应用程序，如果没有系统调用和内核函数，用户将不能编写大型应用程序。

Linux 系统调用，包含了大部分常用系统调用和由系统调用派生出的的函数。

39. 虚拟存储器，虚存，问有啥相关算法

虚拟存储器：由于常规内存的**一次性**（要求将作业全部装入内存后才能运行）和**驻留性**（作业装入内存后，就一直驻留在内存中，直到作业运行结束。）特点，难以满足作业很大和有大量作业要求运行的情况。虚拟存储器是一种借助于外存空间，从而允许一个进程在其运行过程中部分地装入内存的技术。

之所以引入虚拟存储管理方式，是因为程序执行时呈现局部性规律。

局部性原理：

1、 时间局部性：一条指令的一次执行和下次执行，一个数据的一次执行和下次执行，都集中在一个较短时间内。
2、 空间局部性：指当前指令和邻近的几条指令，当前访问的数据和邻近的数据，都集中在一个较小区域。

实现虚拟存储器的硬件支持：

1、 相当数量的外存
2、 相当数量的内存
3、 地址变换机构：以动态实现虚拟地址到实地址的地址变换。

替换算法：替换规则用来确定替换主存中哪一部分，以便腾空部分主存，存放来自辅存要调入的那部分内容。

1、 随机算法：用软件或硬件随机数产生器确定替换的页面。
2、 先进先出：先调入主存的页面先替换。
3、 LRU 算法：替换最长时间不用的页面。
4、 最优算法：替换最长时间以后才使用的页面。这是理想化的算法，只能作为衡量其他各种算法优劣的标准。
5、 CLOCK 置换算法

40. 存储器管理应具有的功能？（+）

1、**内存的分配和回收**
2、**地址变换**：在多道程序环境下，程序的逻辑地址与内存的物理地址不可能一致，因此存储管理器必须提供地址变换机构，将逻辑地址装换为物理地址。
3、**扩充内存容量**：从逻辑上扩充内存容量
4、**存储保护**：保护进入内存的各道作业在自己的存储空间运行，互补干扰。

地址重定位：将目标程序中的逻辑地址转换为可执行代码的实际内存物理地址

41. 什么是 TLB

cache 是一种高速缓存存储器，用于保存 CPU 频繁使用的数据。在使用 Cache 技术的处理器上，当一条指令要访问内存的数据时，首先查询 cache 缓存中是否有数据以及数据是否过期，如果数据未过期则从 cache 读出数据。处理器会定期回写 cache 中的数据到内存。根据程序的局部性原理，使用 cache 后可以大大加快处理器访问内存数据的速度。

TLB 的作用是在处理器访问内存数据的时候做地址转换。TLB 的全称是 Translation Lookaside Buffer，可以翻译做旁路缓冲。**TLB 中存放了一些页表文件，文件中记录了虚拟地址和物理地址的映射关系**。当应用程序访问一个虚拟地址的时候，会从 TLB 中查询出对应的物理地址，然后访问物理地址。TLB 通常是一个分层结构，使用与 Cache 类似的原理。处理器使用一定的算法把最常用的页表放在最先访问的层次。

TLB 加速了查表速度，TLB（块表）放在 cache 中，用于存放慢表中常用的部分内容，专用、快速的硬件缓冲。

42. 程序连接方式有哪些？（+）

1、**静态链接**：
2、**装入时动态链接**：边装入边链接
3、**运行时动态链接**：方便对目标模块的共享

43. 程序的装入方式有哪些？（+）

1、**绝对装入**：
2、**可重定位装入**：也叫静态重定位，地址变换通常是装入时一次性完成的。
3、**动态运行装入**：程序运行时可在内存中移动位置，只有到程序需要真正执行时才把相对地址转换为绝对地址。

44. 什么是交换技术？什么是覆盖技术？及其区别（+）

覆盖技术：把一个大的程序划分为一系列的覆盖，每个覆盖就是一个相对独立的程序单元，把程序执行时并不要求同时装入内存的覆盖组成一组，称为覆盖段。将一个覆盖段分配到同一个存储区域，这个存储区域就称为覆盖区。

交换技术：就是把暂时不用的某个程序及数据部分或全部从内存移到外存中去，以便腾出更多的内存空间。

主要区别：

1、交换主要是在进程和作业之间进行
2、覆盖主要在同一个进程或作业中进行
3、打破了一个程序一旦进入主存便一直运行到结束的限制
4、打破了必须将一个进程的全部信息装入主存后才能运行的限制

45. 内存连续分配管理方式有哪几种？（+）

1、单一连续：适合单道程序
2、固定分区分配：适合多道程序，简单。
3、动态分区分配：根据内存实际需要，动态地为进程分配空间。

46. 动态分区分配算法有哪些？（+）

动态分区分配将内存的空闲分区单独构成一个**空间分区表**或**空闲分区链**

算法名称	原理	优点	缺点	要求
首次适应算法	从链首开始循序查找，找到第一个满足空间要求的空闲分区为止。	优先利用低地址部分，无内部碎片	低地址内存被无限划分，有外部碎片。	空闲分区链以地址递增的次序连接
循环首次适应算法	从上次找到空闲分区的下一个空间分区开始查找。	空闲分区分布均匀，无内部碎片。	缺乏大的空闲分区，有外部碎片。	空闲分区链以地址递增的次序连接
最佳适应算法	把满足要求又是最小的空闲分区分配给作业	产生的外部碎片很小，无内部碎片。	产生需要无法利用的外部小碎片。	空闲分区链以容量从小到大的次序连接
最坏适应算法	把满足要求又是	留下来的空闲空间	大空间不易被保	空闲分区链以容量

	最大的空闲分区分配给作业	比较大，适合下次使用，无内部碎片。	留	从大到小的次序连接

47. 什么是拼接技术？（+）

拼接技术/紧凑技术：移动存储中所有已分配区到内存的一端，将其余空闲分区合并为一个大的空闲分区。

48. 什么是原子操作

"原子操作(atomic operation)是不需要 synchronized"，这是 Java 多线程编程的老生常谈了。所谓原子操作是指不会被线程调度机制打断的操作；这种操作一旦开始，就一直运行倒结束，中间不会有任何 context switch (切换到另一个线程)。

49. 什么内部碎片？什么是外部碎片？（+）

内部碎片：分配给作业的存储空间中未被利用的部分
外部碎片：系统中无法利用的小存储块，比如通过动态内存分配技术从空闲内存区上分配内存后剩下的那部分内存块。

50. 常用存储保护方法有哪些？（+）

1、**界限寄存器**

- 上下界寄存器方法
- 基址、限长寄存器方法

2、**存储保护键**：给每个存储块分配一个单独的存储键，它相当于一把锁。

51. 连续分区分配 vs 非连续分区分配（+）

连续内存分配方式	**非连续内存分配方式：**允许将程序分散装载到很多个不相邻的小分区中。
	基本分页存储管理方式
	请求分页存储管理方式

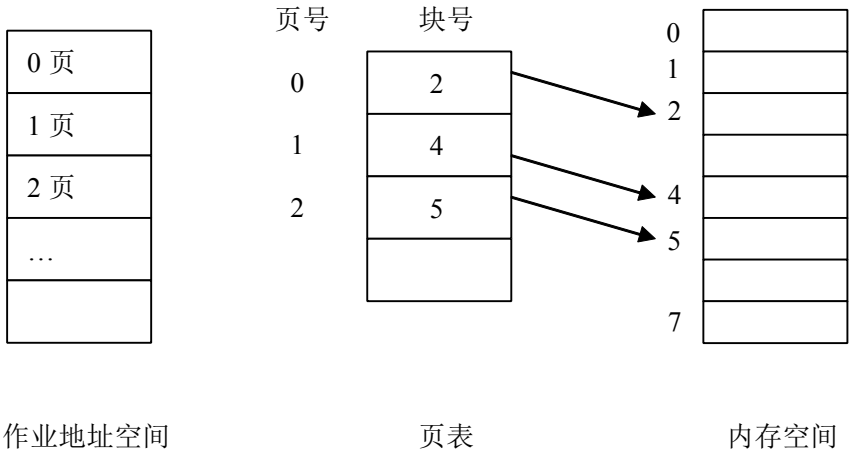
52. 什么是页面？什么是块或物理块？（+）

页面：作业地址空间被划分为若干个大小相等的区域，页面大小应该是 2 的整数幂，通常为 4KB。

块/物理块：将内存存储空间也分为和页大小相等的区域，这些区域被称为块。

53. 什么是页表？及页表的作用？

页表：为了便于在内存中找到进程的每个页面所对应的物理块，系统为每个进程建立一张页面映射表。



54. 段寄存器

段寄存器：在 8086 系统中，访问存储器的地址码由段地址和段内偏移地址两部分组成。段寄存器用来存放各分段的逻辑基值，并指示当前正在使用的 4 个逻辑段，包括代码段寄存器 CS、堆栈段寄存器 SS、数据段寄存器 DS 和附加段数据寄存器 ES。

1、**代码段寄存器 CS**：存放当前正在运行的程序代码所在段的段基值，表示当前使用的指令代码可以从该段寄存器指定的存储器段中取得，相应的偏移值则由 IP 提供。

2、**数据段寄存器 DS**：指出当前程序使用的数据所存放段的最低地址，即存放数据段的段基值。

3、**堆栈段寄存器 SS**：指出当前堆栈的底部地址，即存放堆栈段的段基值。

4、**附加段寄存器 ES**：指出当前程序使用附加数据段的段基址，该段是串操作指令中目的串所在的段。

55. 进程线程树图

进程树是一个形象化的比喻，比如一个进程启动了一个程序，而启动的这个进程就是原来那个进程的子进程，依此形成的一种树形的结构，我们可以在进程管理器选择结束进程树，就可以结束其子进程和派生的子进程。

56. 作业与进程的区别

一个进程是一个程序对某个数据集的执行过程，是分配资源的基本单位。作业是用户需要计算机完成的某项任务，是要求计算机所做工作的集合。一个作业的完成要经过作业提交、作业收容、作业执行和作业完成 4 个阶段。而进程是对已提交完毕的程序所执行过程的描述，是资源分配的基本单位。

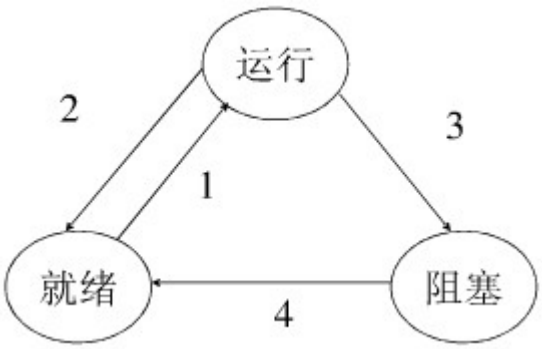
其主要区别如下。

(1)作业是用户向计算机提交任务的任务实体。在用户向计算机提交作业后，系统将它放入外存中的作业等待队列中等待执行。而进程则是完成用户任务的执行实体，是向系统申请分配资源的基本单位。任一进程，只要它被创建，总有相应的部分存在于内存中。

(2) 一个作业可由多个进程组成，且必须至少由一个进程组成，反过来则不成立。

(3) 作业的概念主要用在**批处理系统**中，像 UNIX 这样的分时系统中就没有作业的概念。而进程的概念则用在几乎所有的多道程序系统中进程是操作系统进行资源分配的单位。 在 Windows 下,进程又被细化为线程,也就是一个进程下有多个能独立运行的更小的单位。

57. 进程的三种状态，以及之间转换的过程



58. 进程调度算法

先来先服务，短作业，高优先权，时间片，高响应比，多级反馈队列，这些是进程调度算法吧。

59. 死锁

死锁： 是指两个或两个以上的进程在执行过程中，因争夺资源而造成的一种互相等待的现象，若无外力作用，它们都将无法推进下去。此时称系统处于死锁状态或系统产生了死锁，这些永远在互相等待的进程称为死锁进程。

产生死锁的必要条件：

1、 **互斥条件**：指进程对所分配到的资源进行排它性使用，即在一段时间内某资源只由一个进程占用。
2、 **请求和保持条件**：指进程已经保持至少一个资源，但又提出了新的资源请求，而该资源已被其它进程占有，此时请求进程阻塞，但又对自己已获得的其它资源保持不放。
3、 **不剥夺条件**：指进程已获得的资源，在未使用完之前，不能被剥夺，只能在使用完时由自己释放。
4、 **环路等待条件**：指在发生死锁时，必然存在一个进程——资源的环形链，即进程集合{P0，P1，P2，…，Pn}中的 P0 正在等待一个 P1 占用的资源；P1 正在等待 P2 占用的资源，……，Pn 正在等待已被 P0 占用的资源。

处理死锁的基本方法：

1、 **预防死锁**：这是一种较简单和直观的事先预防的方法。方法是通过设置某些限制条件，去破坏产生死锁的四个必要条件中的一个或者几个，来预防发生死锁。预防死锁是一种较易实现的方法，已被广泛使用。但是由于所施加的限制条件往往太严格，可能会导致系统资源利用率和系统吞吐量降低。
2、 **避免死锁**：该方法同样是属于事先预防的策略，但它并不须事先采取各种限制措施去破坏产生死锁的的四个必要条件，而是在资源的动态分配过程中，用某种方法去防止系统进入不安全状态，从而避免发生死锁。
3、 **检测死锁**：这种方法并不须事先采取任何限制性措施，也不必检查系统是否已经进入不安全区，此方法允许系统在运行过程中发生死锁。但可通过系统所设置的检测机构，及时地检测出死锁的发生，并精确地确定与死锁有关的进程和资源，然后采取适当措施，从系统中将已发生的死锁清除掉。
4、 **解除死锁**：这是与检测死锁相配套的一种措施。当检测到系统中已发生死锁时，须将进程从死锁状态中解脱出来。常用的实施方法是撤销或挂起一些进程，以便回收一些资源，再将这些资源分配给已处于阻塞状态的进程，使之转为就绪状态，以继续运行。

60. 分页和分段的区别

61. 死锁及死锁原因

产生死锁的原因：

1、 **竞争资源引起的进程死锁**：当系统中供多个进程共享的资源如打印机、公用队列的等，其数目不足以满足诸进程的需要时，会引起诸进程对资源的竞争而产生死锁。

2、 **进程推进顺序不当引起的死锁**：由于进程在运行中具有异步性特征，这可能使 P1 和 P2 两个进程推进顺序出现问题。

62. 介绍下银行家算法

避免死锁算法中最有代表性的算法是 Dijkstra E.W 于 1968 年提出的银行家算法：

该算法需要检查申请者对资源的最大需求量，如果系统现存的各类资源可以满足申请者的请求，就满足申请者的请求。

这样申请者就可很快完成其计算，然后释放它占用的资源，从而保证了系统中的所有进程都能完成，所以可避免死锁的发生。

63. RAID

磁盘阵列（Redundant Arrays of Inexpensive Disks，RAID）， 原理是利用数组方式来作磁盘组，配合数据分散排列的设计，提升数据的安全性。

RAID 0：RAID 0 连续以位或字节为单位分割数据，并行读/写于多个磁盘上，因此具有很高的数据传输率，但它没有数据冗余，因此并不能算是真正的 RAID 结构。RAID 0 只是单纯地提高性能，并没有为数据的可靠性提供保证，而且其中的一个磁盘失效将影响到所有数据。因此，RAID 0 不能应用于数据安全性要求高的场合。

RAID 1：它是通过磁盘数据镜像实现数据冗余，在成对的独立磁盘上产生互为备份的数据。当原始数据繁忙时，可直接从镜像拷贝中读取数据，因此 RAID 1 可以提高读取性能。RAID 1 是磁盘阵列中单位成本最高的，但提供了很高的数据安全性和可用性。当一个磁盘失效时，系统可以自动切换到镜像磁盘上读写，而不需要重组失效的数据。

RAID 0+1: 也被称为 RAID 10 标准，实际是将 RAID 0 和 RAID 1 标准结合的产物，在连续地以位或字节为单位分割数据并且并行读/写多个磁盘的同时，为每一块磁盘作磁盘镜像进行冗余。它的优点是同时拥有 RAID 0 的超凡速度和 RAID 1 的数据高可靠性，但是 CPU 占用率同样也更高，而且磁盘的利用率比较低。

RAID 2 将数据条块化地分布于不同的硬盘上，条块单位为位或字节，并使用称为"加重平均纠错码（海明码）"的编码技术来提供错误检查及恢复。这种编码技术需要多个磁盘存放检查及恢复信息，使得 RAID 2 技术实施更复杂，因此在商业环境中很少使用。

RAID 3：它同 RAID 2 非常类似，都是将数据条块化分布于不同的硬盘上，区别在于 RAID 3 使用简单的奇偶校验，并用单块磁盘存放奇偶校验信息。如果一块磁盘失效，奇偶盘及其他数据盘可以重新产生数据；如果奇偶盘失效则不影响数据使用。RAID 3 对于大量的连续数据可提供很好的传输率，但对于随机数据来说，奇偶盘会成为写操作的瓶颈。

RAID 4：RAID 4 同样也将数据条块化并分布于不同的磁盘上，但条块单位为块或记录。RAID 4 使用一块磁盘作为奇偶校验盘，每次写操作都需要访问奇偶盘，这时奇偶校验盘会成为写操作的瓶颈，因此 RAID 4 在商业环境中也很少使用。

RAID 5：RAID 5 不单独指定的奇偶盘，而是在所有磁盘上交叉地存取数据及奇偶校验信息。在 RAID 5 上，读/写指针可同时对阵列设备进行操作，提供了更高的数据流量。RAID 5 更适合于小数据块和随机读写的数据。RAID 3 与 RAID 5 相比，最主要的区别在于 RAID 3 每进行一次数据传输就需涉及到所有的阵列盘；而对于 RAID 5 来说，大部分数据传输只对一块磁盘操作，并可进行并行操作。在 RAID 5 中有"写损失"，即每一次写操作将产生四个实际的读/写操作，其中两次读旧的数据及奇偶信息，两次写新的数据及奇偶信息。

RAID 6：与 RAID 5 相比，RAID 6 增加了第二个独立的奇偶校验信息块。两个独立的奇偶系统使用不同的算法，数据的可靠性非常高，即使两块磁盘同时失效也不会影响数据的使用。但 RAID 6 需要分配给奇偶校验信息更大的磁盘空间，相对于 RAID 5 有更大的"写损失"，因此"写性能"非常差。较差的性能和复杂的实施方式使得 RAID 6 很少得到实际应用。

RAID 7：这是一种新的 RAID 标准，其自身带有智能化实时操作系统和用于存储管理的软件工具，可完全独立于主机运行，不占用主机 CPU 资源。RAID 7 可以看作是一种存储计算机（Storage Computer），它与其他 RAID 标准有明显区别。除了以上的各种标准 如表 1），我们可以如 RAID 0+1 那样结合多种 RAID 规范来构筑所需的 RAID 阵列，例如 RAID 5+3（RAID 53）就是一种应用较为广泛的阵列形式。用户一般可以通过灵活配置磁盘阵列来获得更加符合其要求的磁盘存储系统。

64. 数据库里面的： 什么分级？什么是 ER 图？

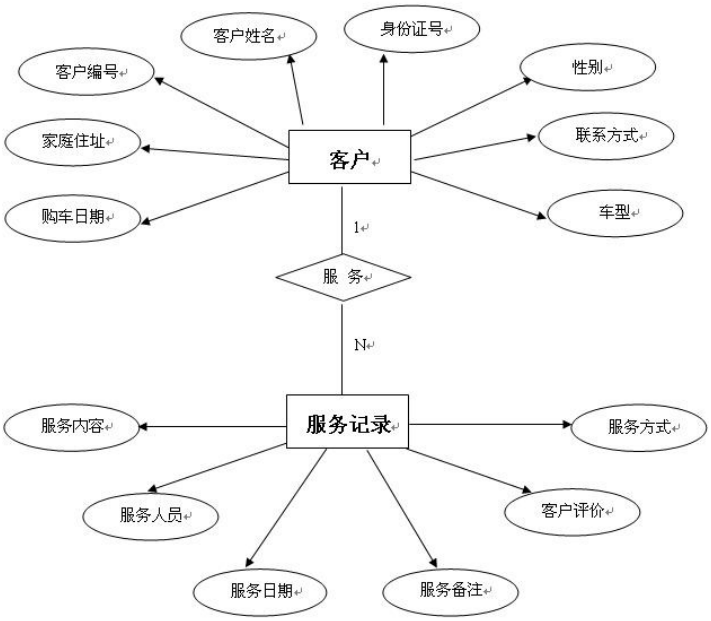
实体-联系图(Entity-Relation Diagram)用来建立数据模型,在数据库系统概论中属于概念设计阶段，形成一个独立于机器，独立于 DBMS 的 ER 图模型。 通常将它简称为 ER 图，相应地可把用 ER 图描绘的数据模型称为 ER 模型。ER 图提供了表示实体（即数据对象）、属性和联系的方法，用来描述现实世界的概念模型。

构成 E-R 图的基本要素是实体、属性和联系，其表示方法为：

- 实体型：用**矩形**表示，矩形框内写明实体名；
- 属性：用**椭圆形或圆角矩形**表示，并用无向边将其与相应的实体连接起来；多值属性由双线连接；主属性名称下加下划线；
- 联系：用**菱形**表示，菱形框内写明联系名，并用无向边分别与有关实体连接起来，同时在无向边旁标上联系的类型

在 E-R 图中要明确表明 1 对多关系，1 对 1 关系和多对多关系。

- 1 对 1 关系在两个实体连线方向写 1；
- 1 对多关系在 1 的一方写 1，多的一方写 N；
- 多对多关系则是在两个实体连线方向各写 N,M



65. 数据库的三级模式结构

http://baike.baidu.com/view/1186644.html?fromTaglist

数据库领域公认的标准结构是三级模式结构，它包括**外模式**、**模式**和**内模式**，有效地组织、管理数据，提高了数据库的**逻辑独立性**和**物理独立性**。

用户级对应**外模式**
概念级对应**模式**
物理级对应**内模式**

视图，就是指观察、认识和理解数据的范围、角度和方法，是数据库在用户“眼中”的反映，很显然，不同层次(级别)用户所“看到”的数据库是不相同的。

使不同级别的用户对数据库形成不同的视图。

数据库的**三级模式**分别为**外模式**、**模式**、**内模式**。

1、 模式

模式又称概念模式或逻辑模式，对应于概念级。它是由数据库设计者综合所有用户的数据，按照统一

的观点构造的全局逻辑结构，是对数据库中全部数据的逻辑结构和特征的总体描述，是所有用户的公共数据视图(全局视图)。它是由数据库管理系统提供的数据模式描述语言(Data Description Language，DDL)来描述、定义的，体现、反映了数据库系统的整体观。

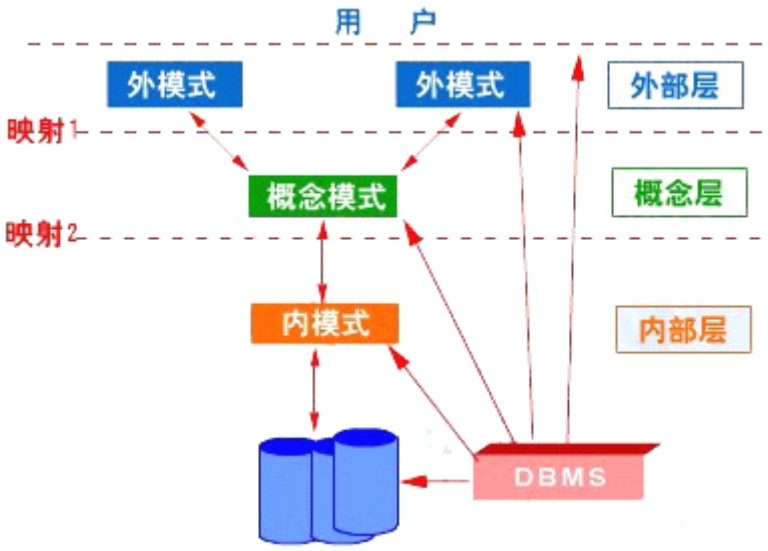
2、 外模式

外模式又称子模式或用户模式，对应于用户级。它是某个或某几个用户所看到的数据库的数据视图，是与某一应用有关的数据的逻辑表示。外模式是从模式导出的一个子集，包含模式中允许特定用户使用的那部分数据。用户可以通过外模式描述语言来描述、定义对应于用户的数据记录(外模式)，也可以利用数据操纵语言(Data Manipulation Language，DML)对这些数据记录进行。外模式反映了数据库的用户观。

3、 内模式

内模式又称存储模式，对应于物理级，它是数据库中全体数据的内部表示或底层描述，是数据库最低一级的逻辑描述，它描述了数据在存储介质上的存储方式和物理结构，对应着实际存储在外存储介质上的数据库。内模式由内模式描述语言来描述、定义，它是数据库的存储观。

在一个数据库系统中，只有唯一的数据库， 因而作为定义 、描述数据库存储结构的**内模式**和定义、描述数据库逻辑结构的**模式**，也是**唯一**的，但建立在数据库系统之上的应用则是非常广泛、多样的，所以对应的**外模式不是唯一**的，也不可能是唯一的。



用户应用程序根据外模式进行数据操作，通过外模式一模式映射，定义和建立某个外模式与模式间的对应关系，将外模式与模式联系起来，当模式发生改变时，只要改变其映射，就可以使外模式保持不变，对应的应用程序也可保持不变；另一方面，通过模式一内模式映射，定义建立数据的逻辑结构(模式)与存储结构(内模式)间的对应关系，当数据的存储结构发生变化时，只需改变模式一内模式映射，就能保持模式不变，因此应用程序也可以保持不变。

66. 视图在数据库的第几层

数据库有三层模式：外模式（用户模式或子模式）、模式（逻辑模式）内模式（物理模式）。视图属于外模式范畴。

67. 数据库的两级映像是什么，作用

数据库的二级映像功能与数据独立性 为了能够在内部实现数据库的三个抽象层次的联系和转换，数据库管理系统在这三级模式之间提供了两层映像。

（1）外模式 / 模式映像 对应于同一个模式可以有任意多个外模式。对于每一个外模式，数据库系统都有一个外模式 / 模式映像，它定义了该外模式与模式之间的对应关系。 当模式改变时，由数据库管理员对各个外模式 / 模式映像作相应的改变，可以使外模式保持不变。应用程序是依据数据的外模式编写的，从而应用程序可以不必修改，保证了数据与程序的逻辑独立性。

（2）模式 / 内模式映像 数据库中只有一个模式，也只有一个内模式，所以模式 / 内模式映像是惟一的，它定义了数据库的全局逻辑结构与存储结构之间的对应关系。当数据库的存储结构改变时，由数据库管理员对模式/内模式映像做相应改变，可以使模式保持不变，从而应用程序也不必修改。保证了数据与程序的物理独立性。

68. 数据库，ER 模型转换成关系模型是数据库设计的第几个阶段。

第四步骤

第一步，规划。规划阶段的主要任务是进行建立数据库的必要性及可行性分析。如系统调查（即对企业全面调查，画出组织层次图，以了企业组织结构），可行性分析，确定 DBS（数据库系统）的总目标和制定项目开发计划。

第二步，需求分析。需求分析阶段应该对系统的整个应用情况作全面的、详细的调查，确定企业组织的目标，收集支持系统总的设计目标的基础数据和对这些数据的要求，确定用户的需求，并把这些要求写成用户和数据库设计者都能够接受的需求分析报告。这一阶段的工作只要有，分析用户活动，产生业务流程图; 确定系统范围，产生体统范围图；分析用户活动涉及的数据，产生数据流程图；分析系统数据，产生数据字典。

第三步，概念设计。概念设计的目标是产生反应企业组织信息需求的数据库概念结构,即设计出独立与计算机硬件和 DBMS（数据库管理系统）的概念模式。E-R 模型是主要设计工具。

第四步，逻辑结构设计。其目的是把概念设计阶段设计好的全局 E-R 模式转换成与选用的具体机器上的 DBMS 所支持的数据模型相符合的逻辑结构（包括数据库模式和外模式）。

第五步，数据库的物理设计。对于给定的数据模型选取一个最适合应用应用环境的物理结构的过程。数据库的物理结构主要指数据库的存储记录格式、存储记录安排和存取方法，完全依赖于给定的硬件环境赫尔数据库产品。

第六步，数据库的实现。该阶段主要有 3 项工作：1 建立实际数据库结构 2 装入试验数据对应用程序进行调试 3 装入实际数据，进入试运行状态。

第七步，数据库的运行与维护。数据库系统的正式运行，标志着数据库设计与应用开发工作的结束和维护阶段的开始，该阶段有 4 项任务：1 维护数据库的安全性与完整性 2 监测并改善数据库运行性能 3 根据用户要求对数据库现有功能进行扩充 4 及时改正运行中发现的系统错误。

69. 数据库，数据模型有哪几种，说出至少两种的特征

1、非关系模型

- 层次模型：记录之间的联系通过指针实现，查找效率高。
- 网状模型：一个结点可以有多于一个的双亲，允许一个以上的结点无双亲。

2、关系模型：概念简单，结构清晰，用户易学易用

3、面向对象模型

4、对象关系模型

70. 数据库中什么叫主码

主关键字(primary key)是表中的一个或多个字段，它的值用于惟一地标识表中的某一条记录。

71. 数据库的表怎样分级

72. 事务

数据库事务(Database Transaction) ，是指作为单个逻辑工作单元执行的一系列操作。 事务处理可以确保除非事务性单元内的所有操作都成功完成，否则不会永久更新面向数据的资源。通过将一组相关操作组合为一个要么全部成功要么全部失败的单元，可以简化错误恢复并使应用程序更加可靠。一个逻辑工作单元要成为事务，必须满足所谓的 ACID(原子性、一致性、隔离性和持久性)属性。

数据库事务的 ACID 性

原子性：事务必须是原子工作单元

一致性：事务在完成时，必须使所有的数据都保持一致状态。
隔离性：由并发事务所作的修改必须与任何其它并发事务所作的修改隔离。
持久性：事务完成之后，它对于系统的影响是永久性的。

73. 数据库操纵语言，定义语言，定义、操作、查询、控制

数据操纵语言 DML（Data Manipulation Language），用户通过它可以实现对数据库的基本操作。例如，对表中数据的插入、删除和修改。

插操作：数据操纵语言 DML（Data Manipulation Language），用户通过它可以实现对数据库的基本操作。例如，对表中数据的插入、删除和修改。
删操作：删除数据库中不必再继续保留的一组记录，如 DELETE 对数据库中记录作删除标志。PACK 是将标有删除标志的记录彻底清除掉。ZAP 是去掉数据库文件的所有记录。
改操作：
排序操作
检索操作

74. 数据库中怎样预防死锁

死锁发生在当多个进程访问同一数据库时，其中每个进程拥有的锁都是其他进程所需的，由此造成每个进程都无法继续下去。

理解了死锁的原因，尤其是产生死锁的四个必要条件，就可以最大可能地避免、预防和解除死锁。

下列方法有助于最大限度地降低死锁：
（1）按同一顺序访问对象。
（2）避免事务中的用户交互。
（3）保持事务简短并在一个批处理中。
（4）使用低隔离级别。
（5）使用绑定连接。

75. 并发控制是为了保证事务的？

当多个用户并发地存取数据库时可能出现多个事务同时存取同一数据的情况，并发控制机制将对这些并发操作加以控制以保证每个事务的 ACID 性，确保数据库的一致性。

76. 在数据库中什么是关系，它和普通二维表啥区别

二维表可以看作是一种关系，关系是二维表的超集

77. 视图、索引

视图概念：计算机数据库中的视图是一个虚拟表，其内容由查询定义。同真实的表一样，视图包含一系列带有名称的列和行数据。但是，视图并不在数据库中以存储的数据值集形式存在。行和列数据来自由定义视图的查询所引用的表，并且在引用视图时动态生成。

视图作用：

1、 为用户集中数据，简化用户的数据查询和处理。
2、 屏蔽数据库的复杂性
3、 便于数据共享

索引的概念：数据库中的索引和书籍中的索引类似。在数据库中，索引使数据库程序无需对整个表进行扫描。就可以在其中找到所需要数据。

索引的作用：

1、 通过创建唯一索引，可以保证数据记录的唯一性。
2、 可以大大加快数据检索速度。
3、 加速表与表之间的连接。

78. 还有个根本没见过的说是什么标准。。。在数据库中的那一层？

79. 数据库里的读锁、写锁

读写锁实际是一种特殊的自旋锁，它把对共享资源的访问者划分成读者和写者，读者只对共享资源进行读访问，写者则需要对共享资源进行写操作。这种锁相对于自旋锁而言，能提高并发性，因为在多处理器系统中，它允许同时有多个读者来访问共享资源，最大可能的读者数为实际的逻辑 CPU 数。写者是排他性的，一个读写锁同时只能有一个写者或多个读者（与 CPU 数相关），但不能同时既有读者又有写者。

80. 数据库里，如何解决数据冗余问题？

81. 关系范式

第一范式：满足最低程度要求的范式，简称 1NF。
定义 设 R 是一个关系模式，如果 R 中的每一个属性 A 的值域中的每个值都是不可分解的，则称 R 是属于第一范式的，记作 R ∈ 1NF。

例如：在关系 SA（姓名，工资）中，属性“工资”还可再分为基本工资，奖金还有补贴 3 个数据项，这违背了第一范式中元组的每个属性不可再分的原则，所以它不满足第一范式。

第二范式：在第一范式中进一步满足一些要求的关系，简称 2NF。

定义如果关系 R ∈ 1NF，并且 R 中每一个非主属性完全函数依赖于任一个候选码，则 R ∈ 2NF。

从定义可以看出，若某个 1NF 的关系的主码只由一个列组成，那么这个关系就是 2NF 关系。但是，如果主码是由多个属性列共同组成的复合主码，并且存在非主属性对属性的部分函数依赖，则这个关系不是 2NF 关系。

例如：在关系 SB（学号，姓名，系名，系主任，课号，成绩）中，、

非主属性“姓名”仅函数依赖于“学号”，也就是“姓名”部分函数依赖于主码（学号，课号）而不是完全依赖；

非主属性“系名”仅函数依赖于“学号”，也就是“系名”部分函数依赖于主码（学号，课号）而不是完全依赖；

非主属性“系主任”仅函数依赖于“学号”，也就是“系主任”部分函数依赖于主码（学号，课号）而不是完全依赖

第三范式：

对关系模式的属性间的函数依赖加以不同的限制就形成了不同的范式。

定义 如果关系 R ∈ 2NF，并且 R 中每一个非主属性对任何候选码都不存在传递函数依赖，则 R ∈ 3NF 。

从定义中可以看出，如果存在非主属性对主码的传递依赖，则相应的关系模式就不是 3NF。

接着上面的例子，关系模式 SC 和 SD 均是 2NF 的，但在关系 SD（学号，姓名，系名，系主任）中，存在如下函数依赖：

学号 → 系名

系名 → 系主任

系名 −\→ 学号

那么，存在着一个传递函数依赖“学号 → 系主任”成立。

82. 断点之类的问题

所谓断点就是程序被中断的地方，这个词对于解密者来说是再熟悉不过了。那么什么又是中断呢？中断就

是由于有特殊事件（中断事件）发生，计算机暂停当前的任务（即程序），转而去执行另外的任务（中断服务程序），然后再返回原先的任务继续执行。

83. 关于显卡的，显卡作用，原理。

显卡：显卡全称显示接口卡（Video card，Graphics card），又称为显示适配器（Video adapter），显示器配置卡简称为显卡，是个人电脑最基本组成部分之一。

显卡作用：是将计算机系统所需要的显示信息进行转换驱动，并向显示器提供**行扫描信号**，控制显示器的正确显示，是连接显示器和个人电脑主板的重要元件，是“人机对话”的重要设备之一。

工作原理：

1、 将 CPU 送来的数据送到北桥（主桥）再送到 GPU（图形处理器）里面进行处理。
2、 将芯片处理完的数据送到显存。
3、 从显存读取出数据再送到 RAM DAC 进行数据转换的工作（数字信号转模拟信号）。
4、 将转换完的模拟信号送到显示屏。

84. VGA

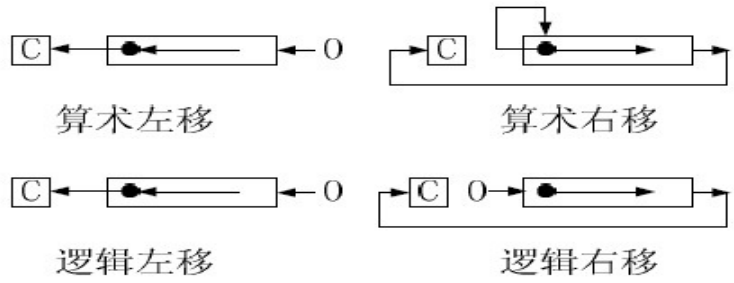
VGA(Video Graphics Array)是 IBM 在 1987 年随 PS/2 机一起推出的一种视频传输标准，具有分辨率高、显示速率快、颜色丰富等优点，在彩色显示器领域得到了广泛的应用。

85. FPGA

FPGA（Field－Programmable Gate Array），即现场可编程门阵列，它是在 PAL、GAL、CPLD 等可编程器件的基础上进一步发展的产物。它是作为专用集成电路（ASIC）领域中的一种半定制电路而出现的，既解决了定制电路的不足，又克服了原有可编程器件门电路数有限的缺点。

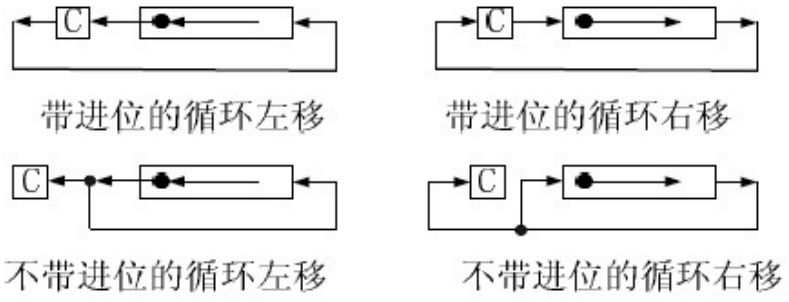
86. 算数移位、逻辑移位、循环移位

算术移位：算术移位指令对带符号数进行移位。
逻辑移位：逻辑移位指令对无符号数进行移位。



这里有一个进位位 C，它就是标志寄存器（即状态寄存器，也称程序状态寄存器 PSW）中的那个进位位，指示是否有进位或借位，若有则该位为 1，否则为 0。逻辑左移跟算术左移完全一样。而逻辑右移跟算术右移则不一样，逻辑右移的最高位在移出后补 0，而在算术右移中，最高位不变（这里的最高位指整个编码的最高位。即有符号数的符号位。），其他跟逻辑右移一样。

循环移位：分为带进位 C 和不带进位 C 两种



注意，在循环移位中没有算法移位、逻辑移位之分，只有是否带进位位之分，不要搞混淆了。在循环移位中，只有“带进位的循环移位”这种方式中，进位位 C 才对移位后的结果产生影响，其他饿进位位都是受影响（被新移入的二进制位覆盖）

87. 386 的保护模式是什么？

保护模式：寻址采用 32 位段和偏移量，最大寻址空间 4GB，最大分段 4GB (Pentium Pre 及以后为 64GB)。在保护模式下 CPU 可以进入虚拟 8086 方式，这是在保护模式下的实模式程序运行环境。

保护模式下程序的运行：

保护模式---从程序运行说起无论实模式还是保护模式，根本的问题还是程序如何在其中运行。

因此我们在学习保护模式时应该时刻围绕这个问题来思考。和实模式下一样，保护模式下程序运行的实质仍是“CPU 执行指令，操作相关数据”，因此实模式下的各种代码段、数据段、堆栈段、中

断服务程序仍然存在，且功能、作用不变。那么保护模式下最大的变化是什么呢？答案可能因人而异，我的答案是“地址转换方式”变化最大。

88. 什么是实模式？（+）

实模式：实模式是指寻址采用和 8086 相同的 16 位段和偏移量，最大寻址空间 1MB，最大分段 64KB。可以使用 32 位指令。32 位的 x86 CPU 用做高速的 8086。

实模式下程序的运行：

程序运行的实质是什么?其实很简单,就是指令的执行,显然 CPU 是指令得以执行的硬件保障，那么 CPU 如何知道指令在什么地方呢?

对了,80x86 系列是使用 CS 寄存器配合 IP 寄存器来通知 CPU 指令在内存中的位置.程序指令在执行过程中一般还需要有各种数据,80x86 系列有 DS、ES、FS、GS、SS 等用于指示不同用途的数据段在内存中的位置。
程序可能需要调用系统的服务子程序，80x86 系列使用中断机制来实现系统服务。总的来说，这些就是实模式下一个程序运行所需的主要内容（其它如跳转、返回、端口操作等相对来说比较次要。）

89. 芯片组是什么

芯片组（Chipset）是构成主板电路的核心。一定意义上讲，它决定了主板的级别和档次。它就是"南桥"和"北桥"的统称，就是把以前复杂的电路和元件最大限度地集成在几颗芯片内的芯片组。

如果说中央处理器（CPU）是整个电脑系统的大脑，那么芯片组将是整个身体的神经。在电脑界称设计芯片组的厂家为 Core Logic，Core(酷睿)的中文意义是核心或中心，光从字面的意义就足以看出其重要性。对于主板而言，芯片组几乎决定了这块主板的功能，进而影响到整个电脑系统性能的发挥，芯片组是主板的灵魂。芯片组性能的优劣，决定了主板性能的好坏与级别的高低。这是因为目前 CPU 的型号与种类繁多、功能特点不一，如果芯片组不能与 CPU 良好地协同工作，将严重地影响计算机的整体性能甚至不能正常工作。

一般来说，芯片组的名称就是以北桥芯片的名称来命名的，例如英特尔 GM45 芯片组的北桥芯片是 G45、最新的则是支持酷睿 i7 处理器的 X58 系列的北桥芯片。主流的有 P45、P43、X48、790GX、790FX、780G、880G、890GX、890FX 等等。NVIDIA 还有 780i、790/等。原因在于：北桥芯片（NorthBridge）是主板芯片组中起主导作用的最重要的组成部分，也称为主桥（HostBridge）。

90. 介绍下南桥和北桥芯片（+）

一块电脑主板，按照一定的方法拿着，以 CPU 插座为北边的话，那靠近 CPU 插座的一个起连接作用的芯片称为“北桥芯片”。英文名：North Bridge Chipset。北桥芯片（NorthBridge）是主板芯片组中起主导作用的最重要的组成部分，也称为主桥（HostBridge）。

北桥芯片就是主板上离 CPU 最近的芯片，这主要是考虑到北桥芯片与处理器之间的通信最密切，为了提高通信性能而缩短传输距离。因为北桥芯片的数据处理量非常大，发热量也越来越大，所以现在的北桥芯片都覆盖着散热片用来加强北桥芯片的散热，有些主板的北桥芯片还会配合风扇进行散热。

北桥芯片负责与 CPU 的联系并控制内存(仅限于 Intel 除 i7 系列以外的 cpu，AMD 系列 cpu 在 K8 系列以后就在 cpu 中集成了内存控制器，因此 AMD 平台的北桥芯片不控制内存)、AGP 数据在北桥内部传输，提供对 CPU 的类型和主频、系统的前端总线频率、内存的类型（SDRAM，DDR SDRAM 以及 RDRAM 等等）和最大容量、AGP 插槽、ECC 纠错等支持，整合型芯片组的北桥芯片还集成了显示核心。

南桥芯片（South Bridge）是主板芯片组的重要组成部分，一般位于主板上离 CPU 插槽较远的下方，PCI 插槽的附近，这种布局是考虑到它所连接的 I/O 总线较多，离处理器远一点有利于布线。相对于北桥芯片来说，其数据处理量并不算大，所以南桥芯片一般都没有覆盖散热片。南桥芯片不与处理器直接相连，而是通过一定的方式（不同厂商各种芯片组有所不同，例如英特尔的英特尔 Hub Architecture 以及 SIS 的 Multi-Threaded“妙渠”）与北桥芯片相连。

91. cache 和虚拟存储的功能不同

cache 是为了提高常访问的内存块儿的速度，虚拟存储是为了扩展空间。

92. 接口芯片使用 8259A 8251

8259A 是专门为了对 8085A 和 8086/8088 进行**中断控制而设计的芯片**，它是可以用程序控制的中断控制器。单个的 8259A 能管理 8 级向量优先级中断。在不增加其他电路的情况下，最多可以级联成 64 级的向量优级中断系统。8259A 有多种工作方式，能用于各种系统。各种工作方式的设定是在初始化时通过软件进行的。 在总线控制器的控制下，8259A 芯片可以处于**编程状态**和**操作状态**，编程状态是 CPU 使用 IN 或 OUT 指令对 8259A 芯片进行初始化编程的状态。

8295A 主要功能： 就是在有多个中断源的系统中，接受外部的中断请求，并进行判断，选中当前优先级最高的中断请求，再将此请求送到 CPU 的 INTR 端；当 CPU 响应中断并进入中断子程序的处理过程后，中断控制器仍负责对外部中断请求的管理。

8259A 工作原理：

一个外部中断请求信号通过中断请求线 IRQ，传输到 IMR（中断屏蔽寄存器），IMR 根据所设定的中断屏蔽字（OCW1），决定是将其丢弃还是接受。如果可以接受，则 8259A 将 IRR（中断请求暂存寄存器）中代表此 IRQ 的位置位，以表示此 IRQ 有中断请求信号，并同时向 CPU 的 INTR（中断请求）管脚发送一个信号。但 CPU 这时可能正在执行一条指令，因此 CPU 不会立即响应。而当这 CPU 正忙着执行某条指令时，还有可能有其余的 IRQ 线送来中断请求，这些请求都会接受 IMR 的挑选。如果没有被屏蔽，那么这些请求也会被放到 IRR 中，也即 IRR 中代表它们的 IRQ 的相应位会被置 1。

当 CPU 执行完一条指令时后，会检查一下 INTR 管脚是否有信号。如果发现有信号，就会转到中断服务，此时，CPU 会立即向 8259A 芯片的 INTA（中断应答）管脚发送一个信号。当芯片收到此信号后，判优部件开始工作，它在 IRR 中，挑选优先级最高的中断，将中断请求送到 ISR（中断服务寄存器），也即将 ISR 中代表此 IRQ 的位置位，并将 IRR 中相应位置零，表明此中断正在接受 CPU 的处理。同时，将它的编号写入中断向量寄存器 IVR 的低三位（IVR 正是由 ICW2 所指定的，不知你是否还记得 ICW2 的最低三位在指定时都是 0，而在这里，它们被利用了！）这时，CPU 还会送来第二个 INTA 信号，当收到此信号后，芯片将 IVR 中的内容，也就是此中断的中断号送上通向 CPU 的数据线。

93. 组成总线里的异步通信

94. 两个时钟不同步的设备怎么通信？

异步通信

异步通信的应答方式可分为

1、**不互锁**：主模块发出请求信号后，不必等待接到从模块的回答信号，而是经过一段时间，确认从模块已经收到请求信号后，便撤销其请求新信号。

2、**半互锁**：主模块发出请求信号，必须等待接到从模块的回答信号后再撤销其请求信号。

3、**全互锁**：主模块发出请求信号，必须等待从模块回答后再撤销其请求信号；从模块发出回答信号后，必须获知主模块请求信号已经撤销后，再撤销其回答信号。

95. 编译:如何把一个机器的语言拿到另一台机器语言机器上执行

96. 编译原理语法分析句法分析

词法分析（英语：**lexical analysis**）是计算机科学中将字符序列转换为单词（Token）序列的过程。进行语法分析的程序或者函数叫作**词法分析器**（Lexical analyzer，简称 Lexer），也叫**扫描器**（Scanner）。

词法分析器一般以函数的形式存在，供语法分析器调用。

词法分析阶段是编译过程的第一个阶段，是编译的基础。这个阶段的任务是从左到右一个字符一个字符地读入源程序，即对构成源程序的字符流进行扫描然后根据构词规则识别单词(也称单词符号或符号)。词法分析程序实现这个任务。

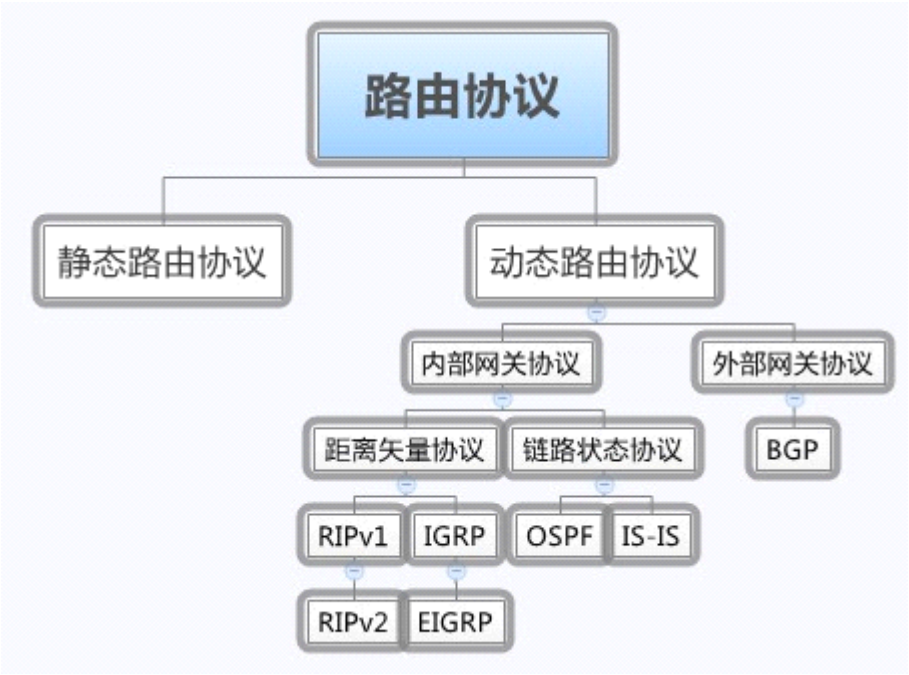
语法分析是编译过程的一个逻辑阶段。语法分析的任务是在词法分析的基础上将单词序列组合成各类语法短语，如“程序”，“语句”，“表达式”等等.语法分析程序判断源程序在结构上是否正确.源程序的结构由上下文无关文法描述.语法分析程序可以用 YACC 等工具自动生成。

句法分析(Parsing)就是指对句子中的词语语法功能进行分析，比如“我来晚了”，这里“我”是主语，“来”是谓语，“晚了”是补语。句法分析现在主要的应用在于中文信息处理中，如机器翻译等。

97. 编译的过程

(1) 词法分析程序 (也称为扫描器);
(2) 语法分析程序 (有时简称为分析器);
(3) 语义分析程序;
(4) 中间代码生成程序;
(5) 代码优化程序;
(6) 目标代码生成程序;
(7) 错误检查和处理程序;
(8) 各种信息表格的管理程序。

98. 路由协议有哪些？



路由分为静态路由和动态路由，其相应的路由表称为静态路由表和动态路由表。静态路由表由网络管理员在系统安装时根据网络的配置情况预先设定，网络结构 发生变化后由网络管理员手工修改路由表。动态路由随网络运行情况的变化而变化，路由器根据路由协议提供的功能自动计算数据传输的最佳路径，由此得到动态路由表。

根据路由算法，动态路由协议可分为距离向量路由协议（Distance Vector Routing Protocol）和链路状态路由协议（Link State Routing Protocol）。距离向量路由协议基于 Bellman-Ford 算法，主要有 RIP、IGRP（IGRP 为 Cisco 公司的私有协议）；链路状态路由协议基于图论中非常著名的 Dijkstra 算法，即最短优先路径（Shortest Path First， SPF）算法，如 OSPF。在**距离向量路由协议中**，路由器将部分或全部的路由表传递给与其相邻的路由器；而**在链路状态路由协议中**，路由器将链路状态信息传 递给在同一区域内的所有路由器。 根据路由器在自治系统（AS）中的位置，可将路由协议分为内部网关协议 （Interior Gateway Protocol，IGP）和外部网关协议（External Gateway Protocol，EGP，也叫域 间路由协议）。域间路由协议有两种：外部网关协议（EGP）和边界网关协议（BGP）。EGP 是为一个简单的树型拓扑结构而设计的，在处理选路循环和设置 选路策略时，具有明显的缺点，目前已被 BGP 代替。

99. 通信的同步异步

100. 比特率波特率

比特率：在数字信道中，比特率是数字信号的传输速率，它用单位时间内传输的二进制代码的有效位(bit)数来表示，其单位为每秒比特数 bit/s(bps)、每秒千比特数(Kbps)或每秒兆比特数(Mbps)来表示(此处 K 和 M 分别为 1000 和 1000000，而不是涉及计算机存储器容量时的 1024 和 1048576)。

波特率：波特率指数据信号对载波的调制速率，它用单位时间内载波调制状态改变次数来表示，其单位为波特(Baud)。

波特率与比特率的关系为：比特率=波特率 X 单个调制状态对应的二进制位数。

101. 网络服务质量包括哪些方面？

QoS（Quality of Service）服务质量，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 应用，或 E-mail 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或丢弃，同时保证网络的高效运行。

QoS 关键指标：

可用性、吞吐量、时延、时延变化(包括抖动和漂移)和丢失

可用性：是当用户需要时网络即能工作的时间百分比。
吞吐量：是在一定时间段内对网上流量(或带宽)的度量。
时延：指一项服务从网络入口到出口的平均经过时间。
时延变化：是指同一业务流中不同分组所呈现的时延不同。
丢包：不管是比特丢失还是分组丢失，对分组数据业务的影响比对实时业务的影响都大。

102. 什么是信道

信道：表示向某一方向传送信息的媒体

103. 信道分类？

模拟信道：传送模拟信号的信道
数字信道：传送数字信号的信道

104. 什么是模拟信号？什么是数字信号？

模拟信号：连续信号，例如：话音信号和广播信号
数字信号：离散信号，二进制代码 0、1 组成的信号

105. 什么是基带信号？什么是宽带信号？

基带信号：将数字信号 1 或 0 直接用不同的电压来表示，然后送到电路上去传输。
宽带信号：将基带信号调制后形成的频分复用模拟信号。由于基带信号经过调制，其频谱移动到较高的频率处。由于每一路基带信号的频谱都被移动到不同的频段上，因此合在一起后并不会互相干扰，这样可以在一条电缆中传送多路的数字信号，因而提高了线路的利用率。

106. java 与 C++区别他说他想问的是地址方面的.

1．指针

JAVA 语言让编程者无法找到指针来直接访问内存无指针，并且增添了自动的内存管理功能，从而有效地防止了 c / c++语言中指针操作失误，如野指针所造成的系统崩溃。但也不是说 JAVA 没有指针，虚拟机内部还是使用了指针，只是外人不得使用而已。这有利于 Java 程序的安全。

2．多重继承

c++支持多重继承，这是 c++的一个特征，它允许多父类派生一个类。尽管多重继承功能很强，但使用复杂，而且会引起许多麻烦，编译程序实现它也很不容易。**Java 不支持多重继承，但允许一个类继承多个接口**(extends+implement)，实现了 c++多重继承的功能，又避免了 c++中的多重继承实现方式带来的诸多不便。

3．数据类型及类

Java 是完全面向对象的语言，所有函数和变量部必须是类的一部分。除了基本数据类型之外，其余的都作为类对象，包括数组。对象将数据和方法结合起来，把它们封装在类中，这样每个对象都可实现自己的特点和行为。而 c++允许将函数和变量定义为全局的。此外，**Java 中取消了 c／c++中的结构和联合，消除了不必要的麻烦。**

4．自动内存管理

Java 程序中所有的对象都是用 new 操作符建立在内存堆栈上，这个操作符类似于 c++的 new 操作符。下面的语句由一个建立了一个类 Read 的对象，然后调用该对象的 work 方法：

```
Read  r＝new  Read();
r.work();
```

语句 Read r＝new Read()；在堆栈结构上建立了一个 Read 的实例。**Java 自动进行无用内存回收操作，不需要程序员进行删除。**而 c 十十中必须由程序贝释放内存资源，增加了程序设计者的负扔。Java 中当一个对象不被再用到时，无用内存回收器将给它加上标签以示删除。**JAVA 里无用内存回收程序是以线程方式在后台运行的，利用空闲时间工作。**

5．操作符重载

Java 不支持操作符重载。操作符重载被认为是 c 十十的突出特征，在 Java 中虽然类大体上可以实现这样的功能，但操作符重载的方便性仍然丢失了不少。Java 语言不支持操作符重载是为了保持 Java 语言尽可能简单。

6．预处理功能

Java 不支持预处理功能。c／c++在编译过程中都有一个预编泽阶段，即众所周知的预处理器。预处理器为开发人员提供了方便，但增加丁编译的复杂性。JAVA 虚拟机没有预处理器，但它提供的引入语句(import)与 c 十十预处理器的功能类似。

7. Java 不支持缺省函数参数，而 c++支持

在 c 中，代码组织在函数中，函数可以访问程序的全局变量。c 十十增加了类，提供了类算法，该算法是与类相连的函数，c 十十类方法与 Java 类方法十分相似，然而，由于 c 十十仍然支持 c，所以不能阻止 c 十十开发人员使用函数，结果函数和方法混合使用使得程序比较混乱。

Java 没有函数，作为一个比 c 十十更纯的面向对象的语言，Java 强迫开发人员把所有例行程序包括在类中，事实上，用方法实现例行程序可激励开发人员更好地组织编码。

8 字符串

c 和 c 十十不支持字符串变量，在 c 和 c 十十程序中使用 Null 终止符代表字符串的结束，在 Java 中字符串是用类对象(strinR 和 stringBuffer)来实现的，这些类对象是 Java 语言的核心，用类对象实现字符串有以下几个优点：

(1)在整个系统中建立字符串和访问字符串元素的方法是一致的；

(2)J3 阳字符串类是作为 Java 语言的一部分定义的，而不是作为外加的延伸部分；

(3)Java 字符串执行运行时检空，可帮助排除一些运行时发生的错误；

(4)可对字符串用“十”进行连接操作。

9“goto 语句

“可怕”的 goto 语句是 c 和 c++的“遗物”，它是该语言技术上的合法部分，引用 goto 语句引起了程序结构的混乱，不易理解，goto 语句子要用于无条件转移子程序和多结构分支技术。鉴于以广理由，Java 不提供 goto 语句，它虽然指定 goto 作为关键字，但不支持它的使用，使程序简洁易读。

10. 类型转换

在 c 和 c 十十中有时出现数据类型的隐含转换，这就涉及了自动强制类型转换问题。例如，在 c 十十中可将一浮点值赋予整型变量，并去掉其尾数。Java 不支持 c 十十中的自动强制类型转换，如果需要，必须由程序显式进行强制类型转换。

11.异常

JAVA 中的异常机制用于捕获例外事件，增强系统容错能力

```
try{/ / 可能产生例外的代码
}catch(exceptionType  name){
//处理
}
```

其中 exceptionType 表示异常类型。而 C++则没有如此方便的机制。

107. 传送介质和无线网络协议

网络传输介质是网络中发送方与接收方之间的物理通路，它对网络的数据通信具有一定的影响。常用的传输介质有：**双绞线**、**同轴电缆**、**光纤**、**无线传输媒介**。

无线网络协议介绍 WLAN 家族成员逐个说明

802.11a

高速 WLAN 协议，使用 5G 赫兹频段。最高速率 54Mbps，实际使用速率约为 22-26Mbps。与 802.11b 不兼容，是其最大的缺点。

802.11b

目前最流行的 WLAN 协议，使用 2.4G 赫兹频段。最高速率 11Mbps，实际使用速率根据距离和信号强度可变 （150 米内 1-2Mbps，50 米内可达到 11Mbps）。802.11b 的较低速率使得无线数据网的使用成本能够被大众接受。另外，通过统一的认证机构认证所有厂商的产品，802.11b 设备之间的兼容性得到了保证。兼容性促进了竞争和用户接受程度。

802.11e

基于 WLAN 的 QoS 协议，通过该协议 802.11a,b,g 能够进行 VoIP。也就是说，802.11e 是通过无线数据网实现语音通话功能的协议。该协议将是无线数据网与传统移动通信网络进行竞争的强有力武器。

802.11g

802.11g 是 802.11b 在同一频段上的扩展。支持达到 54Mbps 的最高速率。兼容 802.11b。该标准已经战胜了 802.11a 成为下一步无线数据网的标准。

802.11h

802.11h 是 802.11a 的扩展，目的是兼容其他 5G 赫兹频段的标准，如欧盟使用的 HyperLAN2。

802.11i

802.11i 是新的无线数据网安全协议，已经普及的 WEP 协议中的漏洞，将成为无线数据网络的一个安全隐患。802.11i 提出了新的 TKIP 协议解决该安全问题。

108. 什么是 SHELL

在计算机科学中，Shell 俗称壳（用来区别于核），是指“提供使用者使用界面”的软件（命令解析器）。它类似于 DOS 下的 command.com。它接收用户命令，然后调用相应的应用程序。同时它又是一种程序设计语言。作为命令语言，它交互式解释和执行用户输入的命令或者自动地解释和执行预先设定好的一连串的命令；作为程序设计语言，它定义了各种变量和参数，并提供了许多在高阶语言中才具有的控制结构，包括循环和分支。

109. 网络里的时延、带宽

时延是指一个报文或分组从一个网络的一端传送到另一个端所需要的时间。它包括了发送时延，传播时延，处理时延，排队时延。（时延=发送时延+传播时延+处理时延+排队时延）一般，发送时延与传播时延是我们主要考虑的。对于报文长度较大的情况，发送时延是主要矛盾；报文长度较小的情况，传播时延是主要矛盾。（计算机网络方面的时延概念）

计算机网络带宽：比特/秒

电信带宽：频带的两个端频率之间的差值

110. 网络拥塞

拥塞现象是指到达通信子网中某一部分的分组数量过多，使得该部分网络来不及处理，以致引起这部分乃到整个网络性能下降的现象，严重时甚至会导致网络通信业务陷入停顿即出现死锁现象。

拥塞产生原因：

（1）多条流入线路有分组到达，并需要同一输出线路，此时，如果路由器没有足够的内存来存放所有分组，那么有的分组就会丢失。

（2）路由器的慢速处理器的缘故，以至于难以完成必要的处理工作（如缓冲区排队、更新路由表等）。那么，即使有多余的线路容量，分组也需要进入到队列中。

防止拥塞的方法：

（1）传输层可采用：重传策略、乱序缓存策略、确认策略、流控制策略和确定超时策略。

（2）网络层可采用：子网内部的虚电路与数据报策略、分组排队和服务策略、分组丢弃策略、路由算法和分组生存管理。

（3）数据链路层可采用：重传策略、乱序缓存策略、确认策略和流控制策略。

111. CSMA/CD 的原理

CSMA/CD（Carrier Sense Multiple Access/Collision Detect）即载波监听多路访问/冲突检测方法 在以太网中，**所有的节点共享传输介质**。如何保证传输介质有序、高效地为许多节点提供传输服务，就是以太网的介质访问控制协议要解决的问题。

CSMA/CD 是一种争用型的介质访问控制协议。它起源于美国夏威夷大学开发的 ALOHA 网所采用的争用型协议，并进行了改进，使之具有比 ALOHA 协议更高的介质利用率。

CSMA/CD 应用在 OSI 的第二层 数据链路层

工作原理：发送数据前，先侦听信道是否空闲，若空闲，则立即发送数据，在发送数据时，边发送边继续侦听，若侦听到冲突，则立即停止发送数据，等待一段随机时间，再重新尝试。

先听后发，边发边听，冲突停发，随机延迟后重发。

112. 数据缓存 cache 的基本概念

高速缓冲存储器（Cache）其原始意义是指存取速度比一般随机存取记忆体（RAM）来得快的一种 RAM，一般而言它不像系统主记忆体那样使用 DRAM 技术，而使用昂贵但较快速的 SRAM 技术，也有快取记忆体的名称。

高速缓冲存储器是存在于主存与 CPU 之间的一级存储器， 由静态存储芯片(SRAM)组成，容量比较小但速度比主存高得多， 接近于 CPU 的速度。

主要由三大部分组成：

Cache 存储体：存放由主存调入的指令与数据块。
地址转换部件：建立目录表以实现主存地址到缓存地址的转换。
替换部件：在缓存已满时按一定策略进行数据块替换，并修改地址转换部件。[1]

113. 应用层有什么协议，作用

简单电子邮件传输（SMTP）、文件传输协议（FTP）、网络远程访问协议（TELNET）、DNS

114. 网络各层的设备是什么和工作原理

传输层：四层交换机、也有工作在四层的路由器
网络层：路由器、三层交换机
数据链路层：网桥、以太网交换机（二层交换机）、网卡（其实网卡是一半工作在物理层、一半工作在数据链路层）
物理层：中继器、集线器、还有我们通常说的双绞线也工作在物理层

115. 传统的搜索引擎基本原理？基于内容的搜索？原理及实现？

116. 客机被迫降到水面上，什么姿势才能保证平稳不栽倒水里面？

流体动力学、牛顿撞击理论

1、为了不发生翻转，飞机的两翼应该保持水平
2、为了使迫降是受到较小的冲击力，应该减少载重，收起起落架，放下侧翼。
3、控制好降落速度
4、最重要的是保持好飞机迫降的角度，经过研究表明飞机迫降仰角大约为 12 度，飞机所有的压力最小，可以安全迫降。

117. 数据传输方式

串行传输
并行传输

118. 数据链路层有哪些协议，举 1~2 例

PPP 协议：当前世界使用最多的数据链路层协议
SLIP 协议：
HDLC

119. 电路交换和分组交换的区别及联系

电路交换：

1、 基于位置，即在某一位置的比特经交换后变更到另一个位置上。
2、 面向连接的，电路交换必定是面向连接的，但面向连接的却不一定是电路交换。
3、 通话过程中始终占用端到端的固定传输带宽。

分组交换：

1、 采用存储转发技术
2、 基于标记的，将欲发送的整块数据称为报文(message)，将较长的报文划分为一个个更小等长数据段（比如 1024bit），在每个数据段前面加上首部(header)，后就构成一个分组

(packet)，分组又称为包，分组的头部称为包头。分组首部包含了目的地址和源地址等重要的控制信息，每一个分组才能在分组交换网中独立地选择路由。

3、 无连接的

120. 电路交换、报文交换、分组交换主要的区别

1、 电路交换，需要先建立链接，才能进行通信。而报文交换和分组交换不需要先建立链接。
2、 若连续传送大量数据，传送时间大于链路建立时间，比较适合用电路交换。
3、 报文交换和分组交换不需要预先分配传输带宽，在传送突发数据时，可提高整个网络的信道利用率。
4、 分组交换比报文交换时延小，但结点交换机必须具备更强的处理能力。

121. 什么是 PN 结？（模电数电）

PN 结（PN junction）。采用不同的掺杂工艺，通过扩散作用，将 P 型半导体与 N 型半导体制作在同一块半导体（通常是硅或锗）基片上，在它们的交界面就形成空间电荷区称 PN 结。PN 结具有单向导电性。P 是 positive 的缩写，N 是 negative 的缩写，表明正荷子与负荷子起作用的特点。

122. CDMA 全称及原理

CDMA：Code Division Multiple Access，码分复用。

抗干扰性强，信号类似白噪声，用于军事，现在也用于民用。

原理：

码片：每个比特时间再划分为 m 个短的间隔，称为码片。

码片序列：
CDMA 系统中，每个站被指定一个 m bit 的码片序列，当站点用发送 1 时，就发送自己的码片序列，当要发送 0 时，就发送自己码片序列的二进制反码。CDMA 系统中的各个站点要求码片序列满足正交性，规格化内积为 1，且一个码片向量和该码片反码向量的内积为-1。

若 A 站要接受 B 站的信息，首先 A 站需要知道 B 站的码片序列，A 站将此码片序列和接收到的码片序列进行内积，当非 B 站发来的码片序列，内积全为 0，全部过滤掉了。剩下的内积为 1 或-1 的信息，就是 B 发给 A 的信息。

123. ICMP

ICMP 是（Internet Control Message Protocol）Internet 控制报文协议。它是 TCP/IP 协议族的一个子协议，用于在 IP 主机、路由器之间传递控制消息。控制消息是指网络通不通、主机是否可达、路由是否可用等网络本身的消息。这些控制消息虽然并不传输用户数据，但是对于用户数据的传递起着重要的作用。

ICMP 协议是一种面向无连接的协议，用于传输出错报告控制信息。它是一个非常重要的协议，它对于网络安全具有极其重要的意义。

我们在网络中经常会使用到 ICMP 协议，比如我们经常使用的用于检查网络通不通的 Ping 命令（Linux 和 Windows 中均有），这个“Ping”的过程实际上就是 ICMP 协议工作的过程。还有其他的网络命令如跟踪路由的 Tracert 命令也是基于 ICMP 协议的。

124. 问我什么是非对称加密？什么是数据安全的特征？

1976 年，美国学者 Dime 和 Henman 为解决信息公开传送和密钥管理问题，提出一种新的密钥交换协议，允许在不安全的媒体上的通讯双方交换信息，安全地达成一致的密钥，这就是“公开密钥系统”。相对于“对称加密算法”这种方法也叫做“非对称加密算法”。

与对称加密算法不同，非对称加密算法需要两个密钥：公开密钥（publickey）和私有密钥（privatekey）。公开密钥与私有密钥是一对，如果用公开密钥对数据进行加密，只有用对应的私有密钥才能解密；如果用私有密钥对数据进行加密，那么只有用对应的公开密钥才能解密。因为加密和解密使用的是两个不同的密钥，所以这种算法叫作非对称加密算法。

125. 保护频带

在给定信道的上下限处留出的未占用窄频带。其目的是确保信道间有足够隔离，防止相邻信道干扰。

126. 问到 PPP 协议

点对点协议（PPP）为在点对点连接上传输多协议数据包提供了一个标准方法。PPP 最初设计是为两个对等节点之间的 IP 流量传输提供一种封装协议。在 TCP-IP 协议集中它是一种用来同步调制连接的**数据链路层协议**（OSI 模式中的第二层），替代了原来非标准的第二层协议，即 SLIP。除了 IP 以外 PPP 还可以携带其它协议，包括 DECnet 和 Novell 的 Internet 网包交换（IPX）。

PPP 协议是一种点——点**串行**通信协议。PPP 具有处理错误检测、支持多个协议、允许在连接时刻协商 IP 地址、允许身份认证等功能，还有其他。PPP 提供了 3 类功能：成帧；链路控制协议 LCP；网络控制协议 NCP。PPP 是**面向字符**类型的协议。

127. 流量控制在哪些层实现

流量控制：实质就是限制发送方的数据流量，使其发送的速率不要超过接收方处理的速率。

1、 **传输层**：端到端的流量控制
2、 **数据链路层**：相邻结点间的流量控制

128. 二层交换机是哪一层的设备，与三层交换机之间的区别？

二层交换技术是发展比较成熟，二层交换机属**数据链路层设备**，可以识别数据包中的 MAC 地址信息，根据 MAC 地址进行转发，并将这些 MAC 地址与对应的端口记录在自己内部的一个地址表中。

三层交换机就是具有部分**路由器功能**的交换机

三层交换技术就是二层交换技术＋三层转发技术。传统交换技术是在 OSI 网络标准模型第二层——数据链路层进行操作的，而三层交换技术是在网络模型中的第三层实现了数据包的高速转发，既可实现网络路由功能，又可根据不同网络状况做到最优网络性能。

129. 三网，指哪三网

电信网络：主要业务是电话，传真等。
有线电视网络：单向电视节目传输网络
计算机网络：我们现在用的很多的局域网和 Internet 等。

130. 分组交换的优点及缺点

优点：

1、 高效：逐段占用通信链路，动态分配资源。
2、 迅速：通信前，不用首先建立链接，可以直接发送。
3、 可靠：完善的网络协议；分布式路由
4、 灵活：每个分组可以独立路由

缺点：

1、时延：分组在各个结点存储转发时因要排队，造成一定的时间延迟。
2、开销：因为包头要携带控制信息，造成一定的网络开销。

131. 组成网络协议的三个要素

1、 语法：数据和控制信息的结构或格式
2、 语意：需要发出何种控制信息，完成何种动作以及做出何种应答。
3、 同步：事件实现顺序的详细说明

132. DNS and DHCP

DNS 是计算机域名系统 (Domain Name System 或 Domain Name Service) 的缩写，它是由解析器和域名服务器组成的。域名服务器是指保存有该网络中所有主机的域名和对应 IP 地址，并具有将域名转换为 IP 地址功能的服务器。

动态主机设置协议（Dynamic Host Configuration Protocol, DHCP）是一个局域网的网络协议，使用 UDP 协议工作，主要有两个用途：给内部网络或网络服务供应商自动分配 IP 地址，给用户或者内部网络管理员作为对所有计算机作中央管理的手段。

133. 网络安全有哪些方面

网络安全是指网络系统的硬件、软件及其系统中的数据受到保护，不因偶然的或者恶意的原因而遭受到破坏、更改、泄露，系统连续可靠正常地运行，网络服务不中断。 网络安全从其本质上来讲就是网络上的信息安全。从广义来说，凡是涉及到网络上信息的保密性、完整性、可用性、真实性和可控性的相关技术和理论都是网络安全的研究领域。网络安全是一门涉及计算机科学、网络技术、通信技术、密码技术、信息安全技术、应用数学、数论、信息论等多种学科的综合性学科。

134. P2P 协议

135. 停止等待协议

停止等待协议（stop-and-wait)是最简单但也是最基础的数据链路层协议。很多有关协议的基本概念都可以从这个协议中学习到。

只有收到序号正确的确认帧 ACKn 后，才更新发送状态变量 V(S)一次，并发送新的数据帧。

接收端接收到数据帧时，就要将发送序号 N(S) 与本地的接收状态变量 V(R) 相比较。

若二者相等就表明是新的数据帧，就收下，并发送确认。

否则为重复帧，就必须丢弃。但这时仍须向发送端发送确认帧 ACKn，而接收状态变量 V(R) 和确认序号 n 都不变。

连续出现相同发送序号的数据帧，表明发送端进行了超时重传。连续出现相同序号的确认帧，表明接收端收到了重复帧。

发送端在发送完数据帧时，必须在其发送缓存中暂时保留这个数据帧的副本。这样才能在出差错时进行重传。只有确认对方已经收到这个数据帧时，才可以清除这个副本。

实用的 CRC 检验器都是用硬件完成的。

CRC 检验器能够自动丢弃检测到的出错帧。因此所谓的“丢弃出错帧”，对上层软件或用户来说都是感觉不到的。

发送端对出错的数据帧进行重传是自动进行的，因而这种差错控制体制常简称为 ARQ (Automatic Repeat reQuest)，直译是自动重传请求，但意思是自动请求重传。

136. ipv4 地址匮乏解决办法

IP 地址的编址共经过了三个基本阶段：分类的 IP 地址、子网的划分、构造超网

分类的 IP 地址二层结构：IP 地址::={<网络号>，<主机号>}
子网划分的三层结构 IP 地址：IP 地址::{<网络号>，<子网号>，<主机号>}
CIDR 地址：IP 地址::{<网络前缀>，<主机号>}

子网划分：在 ip 地址中添加一个“子网号字段”，使两级的 ip 地址变成三级的 ip 地址，这种做法叫做划分子网。

VPN 一般采用隧道技术

A：1~126
B：128~191
C：192~223
D：224~239
E：240~255

1、**IPv6**：根本解决办法。

ipv4:32 位
ipv6:128 位

IPv6 分组由**基本首部**，**扩展首部**，**数据**三部分构成。加快了路由器处理数据报的速度。

IPv6 数据报的目的地址可以是以下三种基本类型的地址之一

单播(unicast)：传统的点对点通信
组播(multicast)：一点对多点的通信
任播(anyeast)：IPv6 新增类型，任播的目的站是一组计算机，但数据报在交付时只交付给其中一个，通常是距离最近的一个。

2、**NAT 技术**：将专用网络内部使用的本地 ip 地址转换成有效的外部使用的全球 ip 地址，使得整个专用网只需要一个全球 ip 地址就可以与英特网联通。此方法可以解决 ip 地址紧缺的问题。

静态 NAT：将内部网络中的每个主机都永久映射成外部外部网络中的某个合法的地址。
动态地址：在外部网络中定义了一系列的合法地址，采用动态分配的方法映射到内部网络。
网络地址端口转换：把内部地址映射到外部网络的一个 ip 地址的不同端口上。

3、**CIDR**：Classless Inter-Domain Routing，消除了传统的 A 类、B 类和 C 类、地址以及划分子网的概念，

因而可以更加有效地分配 IPv4 的地址空间。这是一种为解决地址耗尽而提出的一种措施。

最长前缀匹配==最长匹配==最佳匹配：网络前缀越长，其地址块就越小，因为路由就越具体。

137. 单工、半双工、全双工

单工：单向通信，只能有一个方向的通信，而没有反方向的交互。（例如：无线/有线电广播，电视广播）
半双工：双向交替通信，通信的双方都可以发送信息，但不能同时发送，也不能同时接收。一方发送一方接收，多段时间，再反过来。
全双工：双向同时通信，通信的双方可以同时发送和接收信息。

半双工和全双工通信都需要两条信道。

138. TCP/IP 模型分层

OSI 中的七层	每层完成的功能	TCP/IP 对应协议
应用层	文件传输，电子邮件，文件服务，虚拟终端	HTTP（80）、TELNET、FTP（数据 20、控制 21）、TFTP、SNMP、DNS
表示层	数据格式化、数据加密、解密，数据解压缩	无
会话层	解除、建立和别的结点之间的联系	无
传输层	提供端到端的接口	TCP、UDP
网络层	为数据包选择路由，流量控制，拥塞控制	IP、ICMP、RIP、OSPF、BGP、IGMP
数据链路层	为网络层提供可靠的连接服务，帧为基本单位，组帧，拆帧。	PPP、SLIP、CSLIP、
物理层	以二进制数据形式在物理媒体上传输数据	IEEE802、IEEE802.2

139. IPv4 和 IPv6 怎么相互通信?

从 IPv4 向 IPv6 过渡可以采用**双协议栈**和**隧道技术**

双协议栈：在完全过渡到 IPv6 之前，使一部分主机（或路由器）装有两个协议栈。即一个 IPv4 和一个 IPv6，通过双协议栈进行转换。

隧道技术：是将整个 IPv6 数据报封装到 IPv4 数据报的数据部分，这样，使得 IPv6 数据报可以在 IPv4 网络的隧道中传输。

140. IPv4 的替代方案

IPv4 的替代方案是 IPv6，IPv6 共 128 位。

相对 IPv4 的主要改进

1、 地址变长了，从 32 位变成了 128 位
2、 简化了 IP 分组的基本首部，包含 8 个段（IPv4 包含 12 个段），这一改变使得路由器可以尽快地处理分组，从而可以改善吞吐率。
3、 IPv6 更好地支持选项，这一特征加快了分组处理速度。

IPv6 分组由**基本首部**，**扩展首部**，**数据**三部分构成。加快了路由器处理数据报的速度。

IPv6 数据报的目的地址可以是以下三种基本类型的地址之一

单播(unicast)：传统的点对点通信
组播(multicast)：一点对多点的通信
任播(anyeast)：IPv6 新增类型，任播的目的站是一组计算机，但数据报在交付时只交付给其中一个，通常是距离最近的一个。

IPv6 的地址采用冒号分隔每个 16 位，3CD4:BA23:2254:GFFA:CB4D:674D:DFA5:2345

141. 从网络的作用范围进行分类

1、 广域网：
2、 局域网：
3、 城域网：作用范围在局域网和广域网之间

142. 从网络的使用范围分类

1、 公用网：一般是国家的邮电部门建造的网络
2、 专用网：一般一个单位专用的网络

143. 从网络的拓扑结构分类

1、 星型网络
2、 网状网络
3、 总线网络
4、 令牌环网络
5、 树形网络

144. 信道划分，及其典型应用

信道根据频率来划分

145. 复用的相关概念（频分、时分、码分等）

FDM：Frequency Division Multiplexing， 频分复用
TDM：Time Division Multiplexing，时分复用（同步时分复用），有利于数字信号传输。
STDM：Statistical Time Division Multiplexing，统计时分复用（异步时分复用）
DWDM：Dense Wavelength Division Multiplexing，密集波分复用
CDMA：Code Division Multi Access，码分多址

1、 频分复用的所有用户，在同样的时间占用不同的带宽资源。
2、 时分复用的所有用户，在不同的时间占用同样的频带宽度。
3、 码分多址，每个用户在同一时间使用同样的频带进行通信。
4、 波分复用就是光的频分复用

146. 计算机网络中使用的信道共享技术有哪些？

网络 p84

1、 随机接入
2、 受控接入
3、 信道复用

常用的信道复用（multiplexing）技术

FDM：Frequency Division Multiplexing， 频分复用
TDM：Time Division Multiplexing，时分复用（同步时分复用），有利于数字信号传输。
STDM：Statistical Time Division Multiplexing，统计时分复用（异步时分复用）
DWDM：Dense Wavelength Division Multiplexing，密集波分复用
CDMA：Code Division Multi Access，码分多址

5、 频分复用的所有用户，在同样的时间占用不同的带宽资源。
6、 时分复用的所有用户，在不同的时间占用同样的频带宽度。
7、 码分多址，每个用户在同一时间使用同样的频带进行通信。
8、 波分复用就是光的频分复用

频分复用和时分复用技术比较成熟，缺点是不灵活。

TDM 帧和 STDM 帧都是在物理层中传送的比特流中所划分的帧。这种帧和我们在数据链路层讨论的帧是完全不同的概念，不可混淆。

147. 中国科大软件学院（2012-2013 学年）部分开课老师

许胤龙

电 话：(0551)63601013
E-Mail：ylxu@ustc.edu.cn

主要研究方向：
网络路由算法与协议、网络编码、并行算法等

许胤龙，男，1963年6月生，教授、博导，计算机科学与技术学院副院长，中国科学技术大学第九届学术委员会委员、国家高性能计算中心（合肥）常务副主任、安徽省高性能计算重点实验室副主任、中国计算机学会高性能计算专业委员会委员、国家教育部创新团队"大规模科学工程计算"主要研究骨干。1983年于北京大学数学系获学士学位，1989、2004年于中国科学技术大学计算机系分别获硕士、博士学位。1994-1996年赴德国多特蒙德（Dortmund）大学进修访问。主持完成国家自然科学基金、国家863各1项，参与国家973重大基础研究项目、国家自然科学基金重点项目、国家863重点项目、国家教委博士点基金、国家攀登计划项目多项。目前主持国家863与国家自然科学基金项目各1项。在国内外著名学术刊物、国际学术会议上发表学术论文90余篇。

徐 云

电 话：(0551)63602441
E-Mail：xuyun@ustc.edu.cn
个人主页：http://staff.ustc.edu.cn/~xuyun/

主要研究方向：
大数据挖掘及生物信息学应用，多核并行编程模型及性能优化等。

徐云，男，1960年生，博士，教授。2002年中国科学技术大学计算机系博士毕业，获计算机软件与理论专业博士学位。现为IEEE、ACM和中国计算机学会会员、中国计算机学会理论计算机科学专委会委员。2002年起在中国科学技术大学计算机学院、国家高性能计算中心（合肥）从事并行计算与应用相关的研究和教学工作。近几年主要研究多核并行编程模型及性能优化、高性能生物序列分析算法等。至今已主持国家重点自然科学基金子课题及面上课题、973子课题、863子课题1项各1项，以及华为合作项目等横向项目，作为骨干参加了863、973、发改委和工信部等课题十余项。在国内外重要学术期刊TCBB、TPDS、Bioinformatics、BMC Bioinformatics、PLOS One、Information Sciences、计算机学报和软件学报等及国际学术会议上发表学术论文50余篇。

黄刘生

电　话：(0551)63602445，(0512)87161118　　E- Mail：lshuang@ustc.edu.cn
课题组研究生招生负责人：杨威　　联系方式：15922408989，　qubit@ustc.edu.cn

主要研究方向：
网络（无线传感网、物联网、移动互联网），信息安全(信息隐藏、安全多方计算、量子信息安全)等。

黄刘生，男，1957年4月生，教授、博士生导师，享受国务院政府特殊津贴。曾任中国科大国家高性能计算中心（合肥）副主任，计算机科学技术系系主任，信息科学技术学院副院长。现任中国科大苏州研究院常务副院长，教育部信息安全类专业教学指导委员会委员，中国计算机学会计算机教育专业委员会常委、传感器网络专业委员会委员（发起人之一）、电子政务和办公室自动化专委会委员，《计算机研究与发展》编委、《小型微型计算机系统》编委、《中国科学技术大学学报》编委，中国科大校学术委员会委员、校学位委员会委员等。担任过10多个国际学术会议的程序委员会委员、组织委员会委员、主席。
近年来作为负责人或主要成员承担过国家973、863重点、国家自然科学基金、中科院、国家部委等20多个项目的研究，曾获国家科技进步二等奖、国家教学成果二等奖、中科院、军队等省部级科技进步一、二、三等奖等多项奖励。先后出版著作6部；在ACM Computing Surveys、IEEE Trans. on Wireless Communications、IEEE Trans. on Service Computing、IEEE Trans. on Vehicular Technology、Journal of Parallel and Distributed Computing、Theoretical Computer Science、Optics Communications、Quantum Information Processing、Int'l J. of Modern Physics、Science China等国内外著名杂志和Mobicom、Infocom、Sensys、ICNP等重要国际会议上发表论文200多篇，其中SCI、EI收录180多篇，并有多篇论文获得过国内外重要学术会议的最佳论文奖。

董兰芳

电　话：(0551)63603484
E- Mail：　lfdong@ustc.edu.cn
个人主页：http://staff.ustc.edu.cn/~lfdong/

主要研究方向：
科学计算可视化、计算机动画、模式识别、图像处理

董兰芳,女，1970年7月生，副教授。1991年毕业于兰州大学计算机系,获学士学位,1994年毕业于中国科学技术大学计算机系，获硕士学位,主要承担计算机图形学、面向对象的分析与设计的教学工作,在科学计算可视化、计算机动画、模式识别、图像处理领域进行了长期的研究。已出版教材两本，发表学术论文20余篇。

金培权

电　话：(0551) 63606085-816
E-Mail：jpq@ustc.edu.cn
个 人主 页： http://staff.ustc.edu.cn/~jpq
实验室网站： http://kdelab.ustc.edu.cn

主要研究方向：
移动对象数据库、时空数据库、闪存数据库、互联网信息抽取与搜索等。

金培权，男，1975年出生，博士，中国科学技术大学计算机科学技术系副教授，计算机应用研究室主任，IEEE、ACM和中国计算机学会会员。2003年获中国科学技术大学计算机软件与理论博士学位，之前分别于2000年和1997在天津财经大学获管理学硕士学位和经济学学士学位。2003年9月至2005年8月在中国科学技术大学信息与通信工程博士后流动站从事博士后工作。2009年在德国University of Kaiserslautern做访问学者。从2005年开始，在中国科学技术大学计算机科学技术系任副教授并从事数据库方向的教学研究工作。近年来主持完成了国家自然科学基金项目和863项目各1项、安徽省省级教学研究项目1项，参与国家863、中科院知识创新工程项目等多项课题的研究工作。目前主持了国家自然科学基金面上项目、国家863项目、华为科技基金等多项课题。近几年在国内外期刊和会议上发表论文40余篇。

主讲课程：数据库基础（本）、数据库安全（本）、数据库系统实现技术（硕）。

田　野

电　话：(0551)63602443
E- Mail： yetian@ustc.edu.cn
个人主页：http://staff.ustc.edu.cn/~yetian

主要研究方向：
互联网测量、P2P网络和无线网络。

田野，男，副教授，硕士生导师，IEEE、ACM会员，CCF高级会员。1977年生，四川成都人。2001年毕业于中国科学技术大学，获电子信息工程学士学位，2004年毕业于中国科学技术大学，获计算机软件与理论硕士学位，2007年毕业于香港中文大学，获计算机科学博士学位。2008年2月至8月在美国霍华德大学（Howard University）从事博士后研究。2010年9月至2011年6月在美国纽约大学理工学院（Polytechnic Institute of New York University）从事访问研究。目前主持国家自然科学基金项目一项，教育部博士点基金项目一项，安徽省自然科学基金项目一项，入选中国科学院青年创新促进会。

最新信息，参看个人主页：http://staff.ustc.edu.cn/~yetian。

吴俊敏

电 话：13966707299，0512-87161102
E-mail：jmwu@ustc.edu.cn

主要研究方向：
虚拟化技术与云计算，多核计算机系统与应用，计算机系统仿真与性能优化。

吴俊敏，男，1973年生，博士，副教授。1994年本科毕业于中国科学技术大学计算机系，2005年于中国科学技术大学获计算机系统结构专业博士学位，自2000年以来任中国科学技术大学计算机系教员。承担了多项计算机系统结构方面的科研项目，包括2项国家863课题，1项国家自然科学基金课题，并与IBM、SONY、华为等企业有长期科研合作。目前在重点学术期刊和国际会议上发表论文60余篇，其中EI/SCI索引30余篇。合作出版著作2本，申请有关国家专利13项，授权3项。

Thomas Weise

Telephone: (0551) 63600754
E-Mail: tweise@ustc.edu.cn
Personal Website: http://www.it-weise.de/
Laboratory: Nature Inspired Computation and Applications Laboratory (NICAL)

Research Interests:

Evolutionary Computation, Genetic Programming, Transportation Planning, Logistics, Large-Scale Optimization, Numerical Optimization, Data Mining, Service-Oriented Architectures (SOAs)

李 曦

电 话：(0551)63601556
E- Mail: llxx@ustc.edu.cn

主要研究方向：

嵌入式系统设计方法、低功耗计算机系统和计算机操作系统等

李曦，1963年生于北京，博士，高级工程师。本科毕业于成都气象学院，研究生毕业于中国科技大学计算机系。现任中国科技大学软件学院副院长，中国计算机学会理事，中国计算机学会教育专委会委员等职。主讲本科生《计算机组成原理》和研究生《嵌入式系统设计方法》等课程。在国内外学术期刊和学术会议上发表研究论文70余篇（其中SCI/EI检索30余篇），译著一部。负责或参与国家自然基金、863等纵横向科研项目20余项，以及多项省部级教学研究课题。其研究小组在国内最早开展计算机系统级低功耗优化理论和技术、ASIP体系结构设计与验证和操作系统模型研究。持有操作系统技术专利一项。

Pradip K. Srimani

IEEE Fellow, ACM Distinguished Scientist

http://www.cs.clemson.edu/~srimani

Home:
104 Knollwood Drive
Clemson, SC 29631
Tel: (864) 653-9061

Office:
Department of Computer Science
Clemson University
Clemson, SC 29634
Tel: (864) 656-5886, Fax: (864) 656-0145
Email: srimani@CS.Clemson.EDU

Education

1978 Ph. D. in Radiophysics & Electronics (Computer Science), University of Calcutta, India.

1975 M.Tech. in Radiophysics & Electronics, University of Calcutta, India.

1973 B.Tech. in Radiophysics & Electronics, University of Calcutta, India.

1970 B.Sc.(Hons. in Physics), University of Calcutta, India.

Songwu Lu

Professor

Department of Computer Science
University of California, Los Angeles
4731C Boelter Hall
Los Angeles, CA 90095
Email: slu AT cs.ucla.edu
Phone: (310) 794-9289, Fax: (310) 794-5057

Home | **Publications** | **WiNG Lab** | **Students**

I am a Professor in Computer Science Department at University of California, Los Angeles. I am leading Wireless Networking Group (WiNG) at UCLA. My research interests include wireless networking, mobile systems, cloud computing and wireless and Internet security. Prior to UCLA, I graduated with a Ph.D. from University of Illinois at Urbana-Champaign in 1999.

www.marmakoide.org

The homepage of Alexandre Devert

home | research | teaching | code | about

About me

I am Alexandre Devert, born in 1981, French, Computer Science PhD. I am currently living in Suzhou, China> I work as a lecturer for the USTC's Software Engineering School (aka. SSE), and as a researcher for Adagene. Since I obtained my PhD in 2008, I have been living in China and Vietnam.

148. 中国科大软件学院（苏州）美景


热烈欢迎中科大2012级新生
中国科学技术大学苏州研究院


左旋 右旋
点击跳到下一张


中国科学技术大学软件学院


中国科学技术大学软件学院
SCHOOL OF SOFTWARE ENGINEERING OF USTC