软件设计师考试背记精要

一 计算机组成与体系结构

1、寻址方式：立即寻址最快（操作数本身），寄存器寻址次之（操作数地址的地址），直接寻址最慢（操作数的地址）。

2、数据传输方式

（1）程序控制（查询）方式：分为无条件传送和程序查询方式两种。方法简单，硬件开销小，但 I/O 能力不高，严重影响 CPU 的利用率（不可与 CPU 并行）。

（2）程序中断方式：与程序控制方式相比，中断方式因为 CPU 无需等待而提高了传输请求的响应速度（可与 CPU 并行）。

（3）DMA 方式：DMA 方式是为了在主存与外设之间实现高速、批量数据交换而设置的，DMA 方式比程序控制方式与中断方式都高效（可与 CPU 并行）。

3、存储设备访问速度：通用寄存器>Cache>内存>硬盘。

4、流水线执行时间=单条指令所需时间+（n-1）*流水线周期(指令分段执行中时间最长的一段)。

5、可靠性：用 MTTF/（1+MTTF）来表示。

（1）失效率计算

比如：假设统一型号的 1000 台计算机，在规定的条件下工作 1000 小时，其中 10 台故障。其失效率λ=10/(1000*1000)=1*10-5

（2）千小时可靠度计算

千小时可靠性 R(t)=1-t*λ=1-1000*(1-10-5)=1-0.01=0.99

6、可用性：用 MTBF/（1+MTBF）来表示。

7、校验码

（1）奇偶校验：拼接在信息头部，可检奇数位错，不可纠错。

（2）CRC 循环冗余校验：拼接在信息尾部，可检错，不可纠错。

（3）海明校验：插入在信息中间，可检错，可纠错。

8、运算符的优先顺序：！>算术运算符>关系运算符>&&>||>赋值运算符。

9、浮点数的运算

（1）阶码的位数决定数的表示范围，位数越多范围越大。

（2）尾数的位数决定数的有效精度，位数越多精度越高。

（3）对阶时，小数向大数看齐，较小数的尾数右移实现。

二 操作系统

1、线程共享的内容包括：进程代码段、进程的公有数据（利用这些共享的数据，线程很容易实现相互之间的通讯）、进程打开的文件描述符、信号的处理器、进程的当前目录、进程用户 ID 与进程组 ID 。

2、线程独有的内容包括：线程 ID、寄存器组的值、线程的堆栈（比如，栈指针）、错误返回码、线程的信号屏蔽码。

3、绝对路径从根目录开始的路径，相对路径从当前目录开始的路径。

4、进程的状态

（1）运行：当一个进程在 CPU 运行时。

（2）就绪：一个进程获得了除 CPU 外的一切所需资源，一旦得到处理机即可运行。

（3）阻塞：也称等待或睡眠状态，一个进程正在等待某一事件发生而暂时停止运行，此时即使把 CPU 分配给进程也无法运行。

三 程序设计语言基础

1、解释与编译的区别：

（1）解释程序，也称解释器；直接解释执行源程序，或者将源程序翻译成某种中间代码后再加以执行。

（2）编译程序，也称编译器；将源程序翻译成目标语言程序，然后在计算机上运行目标程序。

（3）两者的根本区别：编译方式下，机器上运行的是与源程序等价的目标程序，源程序和编译程序都不再参与目标程序的执行过程，因此执行时效率较高；解释方式下，解释程序和源程序（或某种等价表示）要参与到程序的运行过程中，运行程序的控制权在解释程序，边解释边执行，执行效率较低。即：解释方式，翻译程序不生成独立的目标程序，而编译方式则生成独立保持的目标程序。

2、程序控制结构主要有：顺序结构、选择结构和循环结构。

3、词法分析：非法字符、单词拼写错误。

4、语法分析：标点符号错误、表达式中缺少操作数、括号不匹配等有关语言结构上的错误。

5、静态语义分析：运算符与运算对象类型不合法、取余时用浮点数等错误。

四 数据结构

1、数组与矩阵：

数组类型	存储地址计算

一维数组 a[n]	a[i]的存储地址为：a+i*len
二维数组 a[m][n]	a[i][j]的存储地址（按行存储）为：a+(i*n+j)*len a[i][j]的存储地址（按列存储）为：a+(j*m+i)*len

2、顺序表与链表对比：

性能类别	具体项目	顺序存储	链式存储
空间性能	存储密度	=1，更优	<1
	容量分配	事先确定	动态改变，更优
时间性能	查找运算	O（n）	O（n）
	读运算	O（1），更优	O（n），最好情况为 1，最坏情况为 n
	插入运算	O（n），最好情况为 0，最坏情况为 n	O（1），更优
	删除运算	O（n）	O（1），更优

3、循环链表：队空条件：head=tail；队满条件：(tail+1)%size=head。

4、树的概念

（1）双亲、孩子和兄弟：结点的子树的根称为该结点的孩子；相应地，该结点称为其子结点的双亲。具有相同双亲的结点互为兄弟。

（这里涉及到 2 个层次，第一个层次的子树，这棵子树的根是第一层结点的孩子结点，第一层结点是其子节点的双亲节点/父节点）。

（2）结点的度：一个结点的子树的个数记为该结点的度。

（3）叶子节点：也称为终端结点，指度为 0 的结点。

（4）内部结点：指度不为 0 的结点，也称为分支节点或非终端节点。除根结点之外，分支结点也称为内部结点。

（5）结点的层次：根为第一层，根的孩子为第二层，依次类推，若某节点在第 i 层，则其孩子结点在

第 i+1 层。

（6）树的高度：一颗树的最大层次数记为树的高度（深度）。

5、二叉树的重要特性

（1）在二叉树的第 i 层上最多有 $2^{i}-1$ 个结点（i≥1）。

（2）深度为 k 的二叉树最多有 $2^{k}-1$ 个结点（k≥1）。

（3）对任何一棵二叉树，如果其叶子结点数为 n0，度为 2 的结点数为 n2，则 n0=n2+1。

（4）如果对一棵有 n 个结点的完全二叉树的结点按层序编号（从第 1 层到$\lfloor \log_2 n \rfloor+1$ 层，每层从左到右），则对任一结点 i（1≤i≤n），有：

如果 i=1，则结点 i 无父结点，是二叉树的根；如果 i>1，则父结点是$\lfloor i/2 \rfloor$；

如果 2i>n，则结点 i 为叶子结点，无左子结点；否则，其左子结点是结点 2i；

如果 2i+1>n，则结点 i 无右子叶点，否则，其右子结点是结点 2i+1。

6、特殊的树

（1）二叉树：二叉树是每个结点最多有两个孩子的有序数，可以为空树，可以只有一个结点。

（2）满二叉树：任何结点，或者是树叶，或者恰有两棵非空子树。

（3）完全二叉树：最多只有最下面的两层结点的度可以小于 2，并且最下面一层的结点全都集中在该层左侧的若干位置。

（4）平衡二叉树：树中任一结点的左右子树高度之差不超过 1。

（5）查找二叉树：又称之为排序二叉树。任一结点的权值，大于其左孩子结点，小于其右孩子结点。

（6）线索二叉树：在每个结点中增加两个指针域来存放遍历时得到的前驱和后继信息。

（7）最优二叉树：又称为哈夫曼树，它是一类带权路径长度最短的树。

7、最优二叉树（哈夫曼树）的构造过程：

（1）根据给定的权值集合，找出最小的两个权值，构造一棵子树将这两个权值作为其孩子结点，二者权值之和作为根结点；

（2）在原集合中删除这两个结点的权值，并引入根节点的权值；

（3）重复步骤（1）和步骤（2），直到原权值集合为空。

8、二叉树的遍历：遍历是按某种策略访问树中的每个结点，且仅访问一次的过程。

（1）前序遍历：又称为先序遍历，按根→左→右的顺序进行遍历。

（2）后序遍历：按左→右→根的顺序进行遍历。

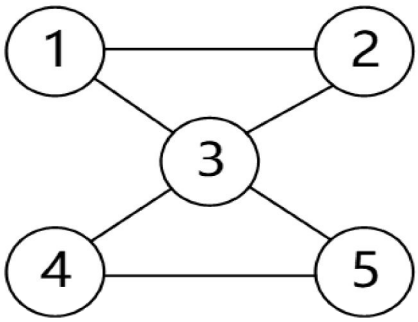
（3）中序遍历：按左→根→右的顺序进行遍历。

（4）层次遍历：按层次顺序进行遍历。

9、图的邻接矩阵表示：用一个 n 阶方阵 R 来存放图中各结点的关联信息，其矩阵元素 R_{ij} 定义为：

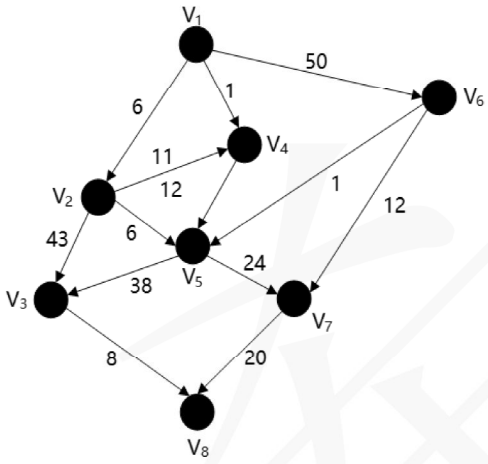
$$R_{ij}=\begin{cases}1 & \text{若顶点i到顶点j有邻接边}\\ 0 & \text{若顶点i到顶点j无邻接边}\end{cases}$$

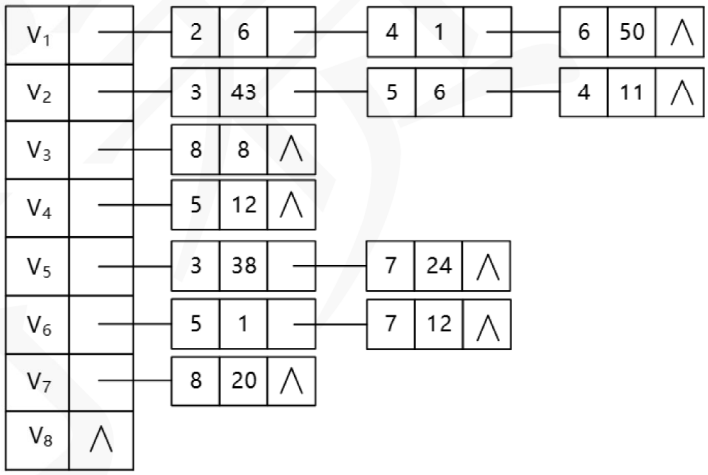
如：



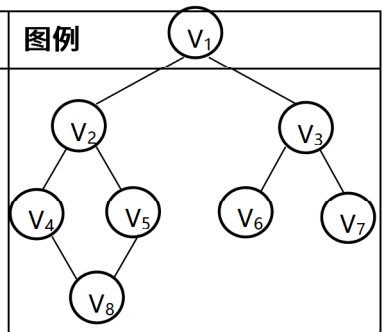
$$R_1=\begin{bmatrix}0&1&1&0&0\\1&0&1&0&0\\1&1&0&1&1\\0&0&1&0&1\\0&0&1&1&0\end{bmatrix}$$

10、图的邻接表表示：首先把每个顶点的邻接顶点用链表示出来，然后用一个一维数组来顺序存储上面每个链表的头指针。如：





11、图的遍历：

遍历方法	说明	示例	图例
深度优先	1.首先访问出发顶点 V； 2.依次从 V 出发搜索 V 的任意一个邻接点 W； 3.若 W 未访问过，则从该点出发继续深度优先遍历； 它类似于树的前序遍历。	V_1，V_2，V_3，V_4，V_5，V_6，V_7	

广度优先	1.首先访问出发顶点 V； 2.然后访问与顶点 V 邻接的全部未访问顶点 W、X、Y…； 3.然后再依次访问 W、X、Y…邻接的未访问的顶点。	V_1，V_2，V_3，V_4，V_5，V_6，V_7，V_8	

12、图的拓扑排序：

在有向图中，若顶点表示活动，用有向边表示活动之间的优先关系（先后关系），则称这样的有向图为活动图，简称 AOV 网。

按照箭头方向把有向图的节点连起来就会形成一个序列。如果该序列没有违反途中节点的先后顺序，就叫做 top 序列，也就是先后序列。

五 算法基础

1、顺序查找的思想：将待查找的关键字为 key 的元素从头到尾与表中元素进行比较，如果中间存在关键字为 key 的元素，则返回成功；否则，则查找失败。

2、二分法查找的基本思想是：（又称折半查找，设 R[low,…,high]是当前的查找区）

（1）确定该区间的中点位置：mid=(low+high)/2。

（2）将待查的 k 值与 R[mid].key 比较，若相等，则查找成功并返回此位置，否则需确定新的查找区间，继续二分查找，具体方法如下：

若 R[mid].key＞k，则由表的有序性可知 R[mid,…,n].key 均大于 k，因此若表中存在关键字等于 k 的结点，则该结点必定是在位置 mid 左边的子表 R[low,…,mid–1]中。因此，新的查找区间是左子表 R[low,…,high]，其中 high=mid–1。

若 R[mid].key<k，则要查找的 k 必在 mid 的右子表 R[mid+1,…,high]中，即新的查找区间是右子表 R[low,…,high]，其中 low=mid+1。

若 R[mid].key=k，则查找成功，算法结束。

（3）下一次查找是针对新的查找区间进行，重复步骤（1）和（2）。

（4）在查找过程中，low 逐步增加，而 high 逐步减少。如果 high<low，则查找失败，算法结束。

折半查找的前提是数据有序顺序存储。

折半查找在查找成功时关键字的比较次数最多为 $\log_2 n + 1$ 次。

折半查找的时间复杂度为 $O(\log_2 n)$次。

3、散列表查找的基本思想是：已知关键字集合 U，最大关键字为 m，设计一个函数 Hash，它以关键字为自变量，关键字的存储地址为因变量，将关键字映射到一个有限的、地址连续的区间 T[0..n-

1](n<<m)中，这个区间就称为散列表，散列查找中使用的转换函数称为散列函数。

4、各种排序算法对比：

类别	排序方法	时间复杂度		空间复杂度	稳定性
		平均情况	特殊情况	辅助存储	
插入排序	直接插入	O（n^2）	基本有序最优 O（n）	O（1）	稳定
	Shell 排序	O（$n^{1.3}$）	-	O（1）	不稳定
选择排序	直接选择	O（n^2）	-	O（1）	不稳定
	堆排序	O（$n\log_2 n$）	-	O（1）	不稳定
交换排序	冒泡排序	O（n^2）	基本有序最优 O（n）	O（1）	稳定
	快速排序	O（$n\log_2 n$）	基本有序最差（n^2）	O（1）	不稳定
归并排序		O（$n\log_2 n$）	--	O（n）	稳定
基数排序		O（d（n+rd））	--	O（rd）	稳定

5、排序算法应用情景对比：

（1）若待排序列的记录数目 n 较小，可采用直接插入排序和简单选择排序。由于直接插入排序所需的记录移动操作比选择排序较简单，因而当记录本身信息量大时，用简单选择排序方法较好。

（2）若待排记录按关键字基本有序，宜采用直接插入排序或冒泡排序。

（3）当 n 很大且关键字位数较少时，采用基数排序较好。

（4）若 n 很大，则应采用时间复杂度为 O($n\log_2 n$)的排序方法，例如快速排序、堆排序或归并排序：

快速排序目前被认为是内部排序中最好的方法，当待排序的关键字为随机分布时，快速排序的平均运行时间最短；

堆排序只需要一个辅助空间，并且不会出现在快速排序中可能出现的最快情况。

快速排序和堆排序都是不稳定的排序方法，若要求排序稳定，可选择归并排序。

6、常见的对算法执行所需时间的度量：$O(1)<O(\log_2 n)<O(n)<O(n\log_2 n)<O(n^2)<O(n^3)<O(2^n)$。

7、常见算法逻辑的时间复杂度：

（1）单个语句，或程序无循环和复杂函数调用：$O(1)$。

（2）单层循环：$O(n)$；双层嵌套循环：$O(n^2)$；三层嵌套循环：$O(n^3)$。

（3）树形结构、二分法、构建堆过程：$O(\log_2 n)$。

（4）堆排序、归并排序：$O(n\log_2 n)$。

（5）所有不同可能的排列组合：$O(2^n)$。

8、常见算法特征总结：

算法名称	关键点	特征	典型问题
分治法	递归技术	把一个问题拆分成多个小模块的相同子问题，一般可用递归解决。	归并排序、快速排序、二分搜索
贪心法	一般用于求满意解，特殊情况可求最优解（部分背包）	局部最优，但整体不见得最优。每步有明确的，既定的策略。	背包问题（如装箱）、多机调度、找零钱问题
动态规划法	最优子结构和递归式	划分子问题（最优子结构），并把子问题结果使用数组存储，利用查询子问题结果构造最终问题结果。	矩阵乘法、背包问题、LCS 最长公共子序列
回溯法	探索和回退	系统的搜索一个问题的所有解或任一解。有试探和回退的过程。	N 皇后问题、迷宫、背包问题

六 系统开发基础

1、软件可维护性：可以用 1/（1+MTTR）来表示。

2、软件开发模型：

（1）瀑布模型：以文档为驱动、适合需求明确的项目。

（2）V 模型：验证确认应用于早期，测试贯穿始终。

（3）原型（演化）模型：是迭代的过程模型，逐步开发出更完整的软件，适用需求不准确项目。

（4）螺旋模型：瀑布和原型模型结合，加入了风险分析，适用庞大、复杂、高风险的系统。

（5）增量模型：第 1 个增量是核心产品，每一增量可分别开发。

（6）喷泉模型：用户需求为动力，对象为驱动模型，迭代无间隙，适用面向对象开发。

3、模块设计：保持模块的大小适中、尽可能减少调用的深度、多扇入少扇出、单入口单出口、模块的作用域应该在模块之内、功能是可预测的。

4、内聚性：

偶然聚合：模块完成的动作之间没有任何关系，或者仅仅是一种非常松散的关系。

逻辑聚合：模块内部的各个组成在逻辑上具有相似的处理动作，但功能用途上彼此无关。

时间聚合：模块内部的各个组成部分所包含的处理动作必须在同一时间内执行。

过程聚合：模块内部各个组成部分所要完成的动作虽然没有关系，但必须按特定的次序执行。

通信聚合：模块的各个组成部分所完成的动作都使用了同一个数据或产生同一输出数据。

顺序聚合：模块内部的各个部分，前一部分处理动作的最后输出是后一部分处理动作的输入。

功能聚合：模块内部各个部分全部属于一个整体，并执行同一功能，且各部分对实现该功能都必不可少。

5、耦合性：

非直接耦合：两个模块之间没有直接关系，它们的联系完全是通过主模块的控制和调用来实现的。

数据耦合：两个模块彼此间通过数据参数交换信息。

标记耦合：一组模块通过参数表传递记录信息，这个记录是某一个数据结构的子结构，而不是简单变量。

控制耦合：两个模块彼此间传递的信息中有控制信息。

外部耦合：一组模块都访问同一全局简单变量而不是同一全局数据结构，而且不是通过参数表传递该全局变量的信息。

公共耦合：两个模块之间通过一个公共的数据区域传递信息。

内容耦合：一个模块需要涉及到另一个模块的内部信息。

七 项目管理

1、项目的关键路径（全部活动完成最短时间）：从开始到结束，需要时间最长的路径。

2、项目工期：项目全部活动完成的最少时间，注意由关键路径即最长路径决定。

3、活动的总时差（松弛时间）：在不延误总工期的前提下，该活动的机动时间，增加项目弹性。

活动的总时差等于该活动最迟完成时间与最早完成时间之差，或该活动最迟开始时间与最早开始时间之差。利用单代号网络图和双代号网路图，可以方便求出活动的松弛时间。 所有松弛时间为 0 的活动连起来就是最长路径。

八 面向对象技术

1、面向对象基本概念：

（1）对象：属性（数据）+方法（操作）+对象 ID。

（2）封装：隐藏对象的属性和实现细节，仅对外公开接口（信息隐藏技术）。

（3）类（实体类/控制类/边界类）：对对象的抽象。

（4）接口：一种特殊的类，它只有方法定义没有实现。

（5）继承与泛化：复用机制（单重继承和多重继承）。

（6）重置/覆盖（Overriding）：在子类中重新定义父类中已经定义的方法。

（7）重载：一个类可以有多个同名而参数类型不同的方法。

（8）多态：不同对象收到同样的消息产生不同的结果。

（9）过载多态：同一个名字在不同的上下文中所代表的含义不同。

（10）动态绑定：根据接收对象的具体情况将请求的操作与实现的方法进行连接（运行时绑定）。

（11）消息和消息通信：对象之间进行通信的一种构造叫做消息。消息是异步通信的（消息传递：接收到信息的对象经过解释，然后予以响应）。

2、面向对象设计原则

（1）**单一职责原则**：设计目的单一的类。

（2）**开放-封闭原则**：对扩展开放，对修改封闭。

（3）李氏（Liskov）替换原则：子类可以替换父类。

（4）**依赖倒置原则**：要依赖于抽象，而不是具体实现；针对接口编程，不要针对实现编程。

（5）**接口隔离原则**：使用多个专门的接口比使用单一的总接口要好。

（6）**组合重用原则**：要尽量使用组合，而不是继承关系达到重用目的。

（7）**迪米特（Demeter）原则（最少知识法则）**：一个对象应当对其他对象有尽可能少的了解。

【面向对象其他设计原则】

（1）重用发布等价原则：重用的粒度就是发布的粒度。

（2）共同封闭原则：包中的所有类对于同一性质的变化应该是共同封闭的。一个变化若对一个包产生影响，则将对该包里的所有类产生影响，而对于其他的包不造成任何影响。

（3）共同重用原则：一个包里的所有类应该是共同重用的。如果重用了包里的一个类，那么就要重用包中的所有类。

（4）无环依赖原则：在包的依赖关系图中不允许存在环，即包之间的结构必须是一个直接的无环图形。

（5）稳定依赖原则：朝着稳定的方向进行依赖。

（6）稳定抽象原则：包的抽象程度应该和其稳定程度一致。

九 数据库系统

1、分布式数据透明性

（1）分片透明：是指用户不必关心数据是如何分片的，它们对数据的操作在全局关系上进行，即如何分片对用户是透明的，因此，当分片改变时应用程序可以不变。分片透明性是最高层次的透明性，如果用户能在全局关系一级操作，则数据如何分布，如何存储等细节自不必关心，其应用程序的编写与集中式数据库相同。

（2）复制透明：用户不用关心数据库在网络中各个节点的复制情况，被复制的数据的更新都由系统自动完成。在分布式数据库系统中，可以把一个场地的数据复制到其他场地存放，应用程序可以使用复制到本地的数据在本地完成分布式操作，避免通过网络传输数据，提高了系统的运行和查询效率。但是对于复制数据的更新操作，就要涉及到对所有复制数据的更新。

（3）位置透明：是指用户不必知道所操作的数据放在何处，即数据分配到哪个或哪些站点存储对用户是透明的。

（4）局部映像透明性（逻辑透明）：是最低层次的透明性，该透明性提供数据到局部数据库的映像，即用户不必关心局部 DBMS 支持哪种数据模型、使用哪种数据操纵语言，数据模型和操纵语言的转换是由系统完成的。因此，局部映像透明性对异构型和同构异质的分布式数据库系统是非常重要的。

2、三级模式

（1）外模式：对应数据库的视图。

（2）模式：对应数据库的基本表。

（3）内模式：对应数据库的存储文件。

3、弱实体：即一个实体的存在必须以另外一个实体为前提，将这类实体称为弱实体，如家属与职工，附件与邮件。

4、主属性与非主属性：组成候选码的属性就是主属性，其他就是非主属性。

5、范式

（1）第一范式：在关系模式中，当且仅当所有域只包含原子值，即每个属性都是不可再分的数据项。

（2）第二范式：当且仅当实体 E 是第一范式，且每一个非主属性完全依赖主键（不存在部分依赖）时。

（3）第三范式：当且仅当实体 E 是第二范式，且 E 中没有非主属性传递依赖于主码时。

6、事物的性质

（1）原子性：事物是原子的，要么都做，要么都不做。

（2）一致性：事物执行的结果必须保持数据库从一个一致性状态变到另一个一致性状态。

（3）隔离性：事物相互隔离，当多个事物并发执行时，任一事物的更新操作直到其成功提交的整个

过程，对其他事物都是不可见的。

（4）持久性：一旦事物成功提交，即使数据库崩溃，其对数据库的更新操作也永久有效。

十 计算机网络

1、TCP/IP 协议族

（1）POP3：110 端口，邮件收取。

（2）SMTP：25 端口，邮件发送。

（3）FTP：20 数据端口/21 控制端口，文件传输协议。

（4）HTTP：80 端口，超文本传输协议，网页传输。

（5）DHCP：67 端口，IP 地址自动分配。

（6）SNMP：161 端口，简单网络管理协议。

（7）DNS：53 端口，域名解析协议，记录域名与 IP 的映射关系。

（8）TCP：可靠的传输层协议。

（9）UDP：不可靠的传输层协议。

（10）ICMP：因特网控制协议，PING 命令来自该协议。

（11）IGMP：组播协议。

（12）ARP：地址解析协议，IP 地址转换为 MAC 地址。

（13）RARP：反向地址解析协议，MAC 地址转换成 IP 地址。

（14）Telent：不安全，SSH 是安全的远程协议。

2、常见网络诊断命令

（1）ping：用于检查网络是否连通。

（2）tracert：用于确定 IP 数据包访问目标所采取的路径，若网络不通，能定位到具体哪个节点不通。

（3）ipconfig：显示 TCP/IP 网络配置值。

ipconfig /all 显示本机 TCP/IP 配置的详细信息，能为 DNS 和 WINS 服务器显示它已配置且所要使用的附加信息（如 IP 地址等），并且显示内置于本地网卡中的物理地址。

ipconfig 在 DHCP 服务中的应用：

ipconfig /all 显示详细信息 ，可查看 DHCP 服务是否已启用。

ipconfig /renew 更新所有适配器。DHCP 客户端手工向服务器刷新请求。

ipconfig /release 释放 IP 地址租约，只能在向 DHCP 服务器租用其 IP 地址的计算机上起作用。

ipconfig 在 DNS 服务中的应用：

ipconfig/ flushdns：清除本地 DNS 缓存。

ipconfig/ displaydns：显示本地 DNS 内容。

ipconfig/ registerdns：DNS 客户端手工向服务器进行注册。

（4）nslookup：查询 DNS 记录。

（5）netstat：用于显示网络连接、路由表和网络接口信息。

3、URL：协议名://主机名.组名.最高层域名。

4、无效的 IP 地址：169.254.X.X（Windows）和 0.0.0.0（linux）。

十一 信息安全

1、对称加密技术：Ke=Kd；加密解密共用一个密钥；

特点：加密强度不高，但效率高；密钥分发困难。

常见对称密钥（共享密钥）加密算法：DES、AES、3DES(三重 DES)、RC-5、IDEA 算法。

2、非对称加密技术：Ke ≠ Kd；密钥必须成对使用（公钥加密，相应的私钥解密）。

特点：加密速度慢，但强度高。

常见非对称密钥（公开密钥）加密算法： RSA、DSA、ECC。

3、典型的摘要算法：SHA，MD5。

4、攻击类型

（1）被动攻击：窃听（网络监听）、业务流分析、非法登录。

（2）主动攻击：假冒身份、抵赖、旁路控制、重放攻击、拒绝服务（DOS）。

5、病毒特性：隐蔽性、传染性、潜伏性、触发性和破坏性。

十二 知识产权与标准化

1、知识产权人确定

情况说明		判断说明	归属
作品	职务作品	利用单位的物质技术条件进行创作，并由单位承担责任的	除署名权外其他著作权归单位
		有合同约定，其著作权属于单位	除署名权外其他著作权归单位
		其他	作者拥有著作权，单位有权在业务范围内优先使用
软件	职务作品	属于本职工作中明确规定的开发目标	单位享有著作权
		属于从事本职工作活动的结果	单位享有著作权
		使用了单位资金、专用设备、未公开的信息等物质、技术条件，并由单位或组织承担责任的软件	单位享有著作权
专利权	职务作品	本职工作中作出的发明创造	单位享有专利
		履行本单位交付的本职工作之外的任务所作出的发明创造	单位享有专利
		离职、退休或调动工作后 1 年内，与原单位工作相关	单位享有专利

情况说明		判断说明	归属
作品软件	委托创作	有合同约定，著作权归委托方	委托方
		合同中未约定著作权归属	创作方
	合作开发	只进行组织、提供咨询意见、物质条件或者进行其他辅助工作	不享有著作权
		共同创作的	共同享有，按人头比例。 成果可分割的，可分开申请。
商标		谁先申请谁拥有（除知名商标的非法抢注） 同时申请，则根据谁先使用（需提供证据） 无法提供证据，协商归属，无效时使用抽签（但不可不确定）	
专利		谁先申请谁拥有 同时申请则协商归属，协商不成则同时驳回双方的专利申请	

2、公民作品的署名权、修改权、保护作品完整权保护期限不受限制。

3、侵权判定

（1）未经许可，发表他人作品。

（2）未经合作作者许可，将与他人合作创作的作品当作自己单独创作的作品发表的。

（3）未参加创作，在他人作品署名。

（4）歪曲、篡改、剽窃他人作品的。

（5）使用他人作品，未付报酬。

（6）未经出版者许可，使用其出版的图书、期刊的版式设计的。

更多备考资料和学习福利，可扫码添加希赛萧萧老师，申请入群