# 《系统架构设计师考试》易混淆知识点

# 系统工程与信息系统基础

## 易混淆点1：系统工程生命周期与信息系统的生命周期

**1、系统工程生命周期阶段**

**探索性研究→概念阶段→开发阶段→生产阶段→使用阶段→保障阶段→退役阶段**

**2、信息系统的生命周期产生阶段→ 开发阶段（单个系统开发：总体规划、系统分析、系统设计、系统实施、系统验收）→运行阶段→消亡阶段**

## 易混淆点2：政府对公民与公民对政府

**1、政府对公民（G2C，Government To Citizen）：**政府对公民提供的服务。

社区公安和水、火、天灾等与公共安全有关的信息。**户口、各种证件和牌照的管理。**

**2、公民对政府（C2G，Citizen To Government）：**个人应向政府缴纳税费和罚款及公民反馈渠道。个人应向政府缴纳的税费款。了解民意，征求群众意见。报警服务（盗贼、医疗、急救、火警）。

**3、政府对政府（G2G，Government To Government）**：政府之间的互动及政府与公务员之间互动。基础信息的采集、处理和利用，如人口信息；各级政府决策支持。

G2G原则上包含：政府对公务员（G2E，Government To Employee）：内部管理信息系统

## 易混淆点3：业务流程管理BPM和业务流程重组BPR

**BPR**是对企业的业务流程进行根本性的再思考和彻底性的再设计，从而获得可以用诸如成本、质量、服务和速度等方面的业绩来衡量的显著性的成就。

**BPM**是一种以规范化的构造端到端的卓越业务流程为中心，以持续的提高组织业务绩效为目的的系统化方法。

**BPM与BPR区别：**

BPM与BPR管理思想最根本的不同就在于流程管理并不要求对所有的流程进行再造。构造卓越的业务流程并不是流程再造，而是根据现有流程的具体情况，对流程进行规范化的设计。

## 易混淆点4：企业门户

**企业网站：**注重单向信息传递，缺互动。

**企业信息门户（EIP）：**把各种应用系统、数据资源和互联网资源统一集成到企业门户之下。强调增强了互动。

**企业知识门户（EKP）：**企业网站的基础上增加知识性内容。

**企业应用门户（EAP）：**实际上是对企业业务流程的集成。它以业务流程和企业应用为核心，把业务流程中功能不同的应用模块通过门户技术集成在一起。

**企业通用门户：**集以上四者于一身。

## 易混淆点5：企业应用集成

（1）**表示集成（界面集成）**

把各应用系统的界面集成起来，统一入口，产生“整体”感觉。



（2）**数据集成**

数据集成是应用集成和业务过程集成的基础。把不同来源、格式、特点性质的数据在逻辑上或物理上有机地集中，从而为企业提供全面的数据共享。ETL、数据仓库、联邦数据库都可视为数据集成。



（3）**控制集成（功能集成、应用集成、API集成）**

业务逻辑层次集成，可以借助于远程过程调用或远程方法调用、面向消息的中间件等技术。



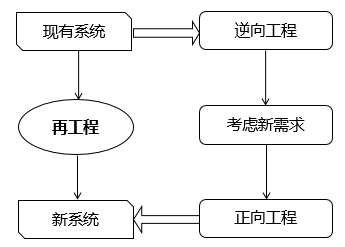
（4）**业务流程集成（过程集成）**

进行业务流程集成时，企业必须对各种业务信息的交换进行定义、授权和管理，以便改进操作、减少成本、提高响应速度。

# 软件工程

## 易混淆点1：逆向工程，正向工程，再工程和设计恢复

（1）**逆向工程**



**实现级：**包括程序的抽象语法树、符号表、过程的设计表示

**结构级：**包括反映程序分量之间相互依赖关系的信息，例如调用图、结构图、程序和数据结构

**功能级：**包括反映程序段功能及程序段之间关系的信息，例如数据和控制流模型

**领域级：**包括反映程序分量或程序诸实体与应用领域概念之间对应关系的信息，例如实体关系模型

（2）**再工程。**再工程是指在逆向工程所获得信息的基础上，修改或重构已有的系统，产生系统的一个新版本。再工程是对现有系统的重新开发过程，包括逆向工程、新需求的考虑过程和正向工程三个步骤。它不仅能从已存在的程序中重新获得设计信息，而且还能使用这些信息来重构现有系统，以改进它的综合质量。在利用再工程重构现有系统的同时，一般会增加新的需求，包括增加新的功能和改善系统的性能。

（3）**正向工程。**正向工程是指不仅从现有系统中恢复设计信息，而且使用该信息去改变或重构现有系统，以改善其整体质量。

（4）**设计恢复。**设计恢复是指借助工具从已有程序中抽象出有关数据设计、总体结构设计和过程设计等方面的信息。

## 易混淆点2：用例关系：包含关系、扩展关系、泛化关系

包含关系：其中这个提取出来的公共用例称为抽象用例，而把原始用例称为基本用例或基础用例，当可以从两个或两个以上的用例中提取公共行为时，应该使用包含关系来表示它们。

扩展关系：如果一个用例明显地混合了两种或两种以上的不同场景，即根据情况可能发生多种分支，则可以将这个用例分为一个基本用例和一个或多个扩展用例，这样使描述可能更加清晰。

泛化关系：当多个用例共同拥有一种类似的结构和行为的时候，可以将它们的共性抽象成为父用例，其他的用例作为泛化关系中的子用例。在用例的泛化关系中，子用例是父用例的一种特殊形式，子用例继承了父用例所有的结构、行为和关系。

## 易混淆点3：需求的分类

**【需求的层次分类】**

**业务需求**：是指反应企业或客户对系统高层次的目标要求，通常来自项目投资人、购买产品的客户、客户单位的管理人员、市场营销部门或产品策划部门等。通过业务需求可以确定项目视图和范围，为以后的开发工作奠定了基础。

**用户需求**：描述的是用户的具体目标，或用户要求系统必须能完成的任务。也就是说，用户需求描述了用户能使用系统来做些什么。

**系统需求**：是从系统的角度来说明软件的需求，**包括功能需求、非功能需求和设计约束**等。

**功能需求**也称为行为需求，它规定了开发人员必须在系统中实现的软件功能，用户利用这些功能来完成任务，满足业务要求。

**性能需求（非功能需求）**是指系统必须具备的属性或品质，又可细分为软件质量属性和其他非功能需求。

**设计约束**也称为限制条件或补充规约，通常是对系统的一些约束说明。

**【需求的QFD分类】**

质量功能部署QFD是一种将用户要求转化成软件需求的技术，其目的是最大限度地提升软件工程过程中用户的满意度。QFD将软件需求分为三类：

**常规需求（基本需求）**：用户认为系统应该做到的功能或性能，实现越多用户会越满意。

**期望需求**：用户想当然认为系统应具备的功能或性能，但并不能正确描述自己想要得到的这些功能或性能需求。如果期望需求没有得到实现，会让用户感到不满意。

**兴奋需求（意外需求）**：是用户要求范围外的功能或性能，实现这些需求用户会更高兴，但不实现也不影响其购买的决策。

## 易混淆点4：系统测试 （主要区分压力测试和强度测试）

负载测试：各种工作负载下系统的性能压力测试【测上限】：系统的瓶颈或不能接受的性能点强度测试【测下限】：系统资源特别低的情况下运行容量测试【并发测试】：同时在线的最大用户数可靠性测试：MTTF之类的参数

## 易混淆点5：系统维护分类

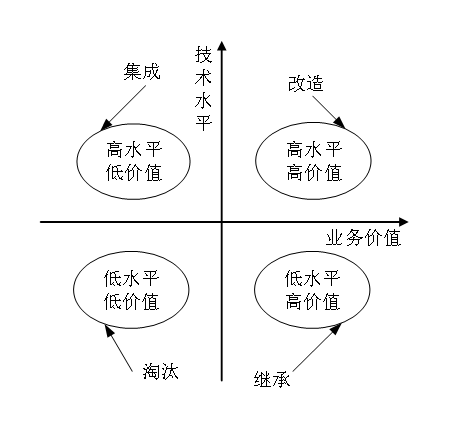
**改正性维护：**指改正在系统开发阶段已发生而系统测试阶段尚未发现的错误。

**适应性维护：**指使应用软件适应信息技术变化和管理需求变化而进行的修改。企业的外部市场环境和管理需求的不断变化也使得各级管理人员不断提出新的信息需求。

**完善性维护：**扩充功能和改善性能而进行的修改。对已有的软件系统增加一些在系统分析和设计阶段中没有规定的功能与性能特征。

**预防性维护：**为了改进应用软件的可靠性和可维护性，为了适应未来的软硬件环境的变化，应主动增加预防性的新的功能，以使用系统适应各类变化而不被淘汰。如将专用报表功能改成通用报表生成功能，以适应将来报表格式的变化。

## 易混淆点6： 遗留系统演化策略（注意区分继承策略和集成策略）



**淘汰策略**：遗留系统的技术含量较低，且具有较低的业务价值。对遗留系统的完全淘汰是企业资源的根本浪费，系统分析师应该善于“变废为宝”，通过对遗留系统功能的理解和借鉴，可以帮助新系统的设计，降低新系统开发的风险。

**继承策略**：遗留系统的技术含量较低，已经满足企业运作的功能或性能要求，但具有较高的商业价值，目前企业的业务尚紧密依赖该系统。对这种遗留系统的演化策略为继承。在开发新系统时，需要完全兼容遗留系统的功能模型和数据模型。为了保证业务的连续性，新老系统必须并行运行一段时间，再逐渐切换到新系统上运行。

**改造策略**：遗留系统具有较高的业务价值，基本上能够满足企业业务运作和决策支持的需要。这种系统可能建成的时间还很短，对这种遗留系统的演化策略为改造。改造包括系统功能的增强和数据模型的改造两个方面。系统功能的增强是指在原有系统的基础上增加新的应用要求，对遗留系统本身不做改变；数据模型的改造是指将遗留系统的旧的数据模型向新的数据模型的转化。

**集成策略**：遗留系统的技术含量较高，但其业务价值较低，可能只完成某个部门（或子公司）的业务管理。这种系统在各自的局部领域里工作良好，但对于整个企业来说，存在多个这样的系统，不同的系统基于不同的平台、不同的数据模型，形成了一个个信息孤岛，对这种遗留系统的演化策略为集成。

# 软件架构设计

## 易混淆点1：架构4+1视图和UML4+1视图





## 易混淆点2：微服务与SOA的对比

|  |  |
| --- | --- |
| 微服务 | SOA |
| 能拆分的就拆分 | 是整体的，服务能放一起的都放一起 |
| 纵向业务划分 | 是水平分多层 |
| 由单一组织负责 | 按层级划分不同部门的组织负责 |
| 细粒度 | 粗粒度 |
| 两句话可以解释明白 | 几百字只相当于SOA的目录 |
| 独立的子公司 | 类似大公司里面划分了一些业务单元（BU） |
| 组件小 | 存在较复杂的组件 |
| 业务逻辑存在于每一个服务中 | 业务逻辑横跨多个业务领域 |
| 使用轻量级的通信方式，如HTTP | 企业服务总线（ESB）充当了服务之间通信的角色 |

|  |  |
| --- | --- |
| 微服务架构实现 | SOA实现 |
| 团队级，自底向上开展实施 | 企业级，自顶向下开展实施 |
| 一个系统被拆分成多个服务，粒度细 | 服务由多个子系统组成，粒度大 |
| 无集中式总线，松散的服务架构 | 企业服务总线，集中式的服务架构 |
| 集成方式简单（HTTP/REST/JSON） | 集成方式复杂（ESB/WS/SOAP） |
| 服务能独立部署 | 单块架构系统，相互依赖，部署复杂 |

## 易混淆点3：敏感点和权衡点

**风险点：**系统架构风险是指架构设计中潜在的、存在问题的架构决策所带来的隐患。

**敏感点：**指为了实现某种特定的质量属性，一个或多个构件所具有的特性。

**权衡点：**影响多个质量属性的特性，是多个质量属性的敏感点。

## 易混淆点4：产品线建立方式

|  |  |  |
| --- | --- | --- |
|  | 演化方式 | 革命方式 |
| 基于现有产品 | 基于现有产品架构设计产品线的架构，经演化现有构件，开发产品线构件 | 核心资源的开发基于现有产品集的需求和可预测的、将来需求的超集 |
| 全新产品线 | 产品线核心资源随产品新成员的需求而演化 | 开发满足所有预期产品线成员的需求的核心资源 |

**易混淆点**5**：MVP与MVP**

MVP是MVC的变种。

MVP实现了V与M之间的解耦（V不直接使用M，修改V不会影响M）。

MVP更好的支持单元测试（业务逻辑在P中，可以脱离V来测试这些逻辑；可以将一个P用于多个V，而不需要改变P的逻辑）。

MVP中V要处理界面事件，业务逻辑在P中，MVC中界面事件由C处理。

**MVP模式与MVC模式的主要区别**：

在组件耦合度方面：在MVP模式中，视图并不直接使用模型，它们之间的通信通过Presenter进行，从而实现了视图与模型的分离，而在MVC模式中，视图直接与模型交互。

在组件分工方面：在MVP模式中，视图需要处理鼠标及键盘等触发的界面事件，而在MVC模式中这通常是由控制器完成的工作；在MVP模式中，系统核心业务逻辑组织集中在Presenter中，而在MVC模式中，相应的控制器通常只完成事件的分发。

在开发工程化支持方面：MVP模式可更好地支持单元测试，而在MVC模式中，由于模型与视图绑定，因此难以实施相应的单元测试；在MVP模式中，Presenter基于约定接口与视图和模型交互，可更好地支持组件的重用。

## 易混淆点6：构件、对象、模块的对比

|  |  |  |
| --- | --- | --- |
| 构件的特性 | 对象的特性 | 模块的特性 |
| 1、独立部署单元；  2、作为第三方的组装单元；  3、没有（外部的）可见状态。 | 1、一个实例单元，具有唯一的标志。2、可能具有状态，此状态外部可见。  3、封装了自己的状态和行为。 | 结构化开发的产物 |

## 易混淆点7：无状态服务和有状态服务

**无状态服务**（stateless service）对单次请求的处理，不依赖其他请求，也就是说，处理一次请求所需的全部信息，要么都包含在这个请求里，要么可以从外部获取到（比如说数据库），服务器本身不存储任何信息。

**有状态服务**（stateful service）则相反，它会在自身保存一些数据，先后的请求是有关联的。

## 易混淆点8：Redis和Memcache

**Redis和Memcache对比：**

Redis和Memcache都是将数据存放在内存中，都是内存数据库。他们都支持key-value数据类型。同时Memcache还可用于缓存其他东西，例如图片、视频等等，Redis还支持list、set、hash等数据结构的存储。

Redis支持数据的持久化，可以将内存中的数据保持在磁盘中，重启的时候可以再次加载进行使用。Memcache挂掉之后，数据就没了。

灾难恢复-Memcache挂掉后，数据不可恢复; Redis数据丢失后可以恢复。

在Redis中，并不是所有的数据都一直存储在内存中的。这是和Memcache相比一个最大的区别。当物理内存用完时，Redis可以将一些很久没用到的value交换到磁盘。

Redis在很多方面支持数据库的特性，可以这样说他就是一个数据库系统，而Memcache只是简单的K/V缓存。

所以在选择方面如果有持久方面的需求或对数据类型和处理有要求的应该选择Redis。

如果简单的key/value存储应该选择Memcache。

## 易混淆点9：Redis持久化

RDB：传统数据库中快照的思想。指定时间间隔将数据进行快照存储。

AOF：传统数据库中日志的思想，把每条改变数据集的命令追加到AOF文件末尾，这样出问题了，可以重新执行AOF文件中的命令来重建数据集。

|  |  |  |
| --- | --- | --- |
| 对比维度 | RDB持久化 | AOF持久化 |
| 备份量 | 重量级的**全量备份**，保存整个数据库 | 轻量级**增量备份**，一次只保存一个修改命令 |
| 保存间隔时间 | 保存**间隔时间长** | 保存间隔时间短，默认1秒 |
| 还原速度 | 数据还原速度快 | 数据还原速度慢 |
| 阻塞情况 | save会阻塞，但bgsave或者自动不会阻塞 | 无论是平时还是AOF重写，都不会阻塞 |
| 数据体积 | 同等数据体积：小 | 同等数据体积：大 |
| 安全性 | 数据安全性：**低**，容易丢数据 | 数据安全性：**高**，根据策略决定 |

# 信息安全技术基础知识

## 易混淆点1： 网络攻击分类

**被动攻击：**收集信息为主，破坏保密性。

|  |  |  |
| --- | --- | --- |
| 攻击类型 | 攻击名称 | 描述 |
| 被动攻击 | 窃听（网络监听） | 用各种可能的合法或非法的手段窃取系统中的信息资源和敏感信息。 |
| 业务流分析 | **通过对系统进行长期监听，利用统计分析方法对诸如通信频度、通信的信息流向**、通信总量的变化等参数进行研究，从而发现有价值的信息和规律。 |
| 非法登录 | 有些资料将这种方式归为被动攻击方式。 |

**主动攻击：**主动攻击的类别主要有：中断（破坏可用性），篡改（破坏完整性），伪造（破坏真实性）

|  |  |  |
| --- | --- | --- |
| 攻击类型 | 攻击名称 | 描述 |
| 主动攻击 | 假冒身份 | 非法用户冒充成为合法用户，特权小的用户冒充成为特权大的用户。 |
| 抵赖 | 否认自己曾经发布过的某条消息、伪造一份对方来信等。 |
| 旁路控制  【旁路攻击】 | 密码学中是指绕过对加密算法的繁琐分析，利用密码算法的硬件实现的运算中泄露的信息。如执行时间、功耗、电磁辐射等，结合统计理论快速的破解密码系统。 |
| 重放攻击 | 所截获的某次合法的通信数据拷贝，出于非法的目的而被重新发送。  加**【时间戳】**能识别并应对重放攻击。 |
| 拒绝服务（DOS） | **破坏服务的【可用性】，**对信息或其他资源的合法访问被无条件的阻止。 |
| XSS跨站脚本攻击 | 通过利用网页【开发时留下的漏洞】，通过巧妙的方法注入恶意指令代码到网页。 |
| CSRF跨站请求伪造攻击 | 攻击者通过一些技术手段欺骗用户的浏览器与访问一个自己曾经认证过的网站并执行一些操作（如转账或购买商品等）。 |
| 缓冲区溢出攻击 | 利用**【缓冲区溢出漏洞】**所进行的攻击。在各种操作系统、应用软件中广泛存在。 |
| SQL注入攻击 | 攻击者把SQL命令插入到Web表单，欺骗服务器执行恶意的SQL命令。  **SQL注入攻击的方式：**【恶意拼接查询】、【利用注释执行非法命令】、【传入非法参数】、【添加额外条件】。  **抵御SQL攻击的方式包括**：【使用正则表达式】、【使用参数化的过滤性语句】、【检查用户输入的合法性】、【用户相关数据加密处理】、【存储过程来执行查询】、【使用专业的漏洞扫描工具】。 |

## 易混淆点2：安全保护等级

（1）用户自主保护级：适用于普通内联网用户

系统被破坏后，对公民、法人和其他组织权益有损害，但不损害国家安全社会秩序和公共利益。

（2）系统审计保护级：适用于通过内联网或国际网进行商务活动，需要保密的非重要单位

系统被破坏后，对公民、法人和其他组织权益有严重损害，或损害社会秩序和公共利益，但不损害国家安全。

（3）安全标记保护级：适用于地方各级国家机关、金融机构、邮电通信、能源与水源供给部门、交通运输、大型工商与信息技术企业、重点工程建设等单位

系统被破坏后，对社会秩序和公共利益造成严重损害，或对国家安全造成损害。

（4）结构化保护级：适用于中央级国家机关、广播电视部门、重要物资储备单位、社会应急服务部门、尖端科技企业集团、国家重点科研机构和国防建设等部门

系统被破坏后，对社会秩序和公共利益造成特别严重损害，或对国家安全造成严重损害。

（5）访问验证保护级：适用于国防关键部门和依法需要对计算机信息系统实施特殊隔离的单位，对国家安全造成特别严重的影响

## 易混淆点3：各模型安全防范功能

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 预警 | 保护 | 检测 | 响应 | 恢复 | 反击 | 管理 |
| PDR | 无 | **有** | **有** | **有** | 无 | 无 | 无 |
| PPDR | 无 | **有** | **有** | **有** | 无 | 无 | 无 |
| PDRR | 无 | **有** | **有** | **有** | **有** | 无 | 无 |
| MPDRR | 无 | **有** | **有** | **有** | **有** | 无 | **有** |
| WPDRRC | **有** | **有** | **有** | **有** | **有** | **有** | **有** |

## 易混淆点4：区块链和比特币

**【区块链】 ≠ 比特币，**比特币底层采用了区块链技术。比特币交易在我国定性为**【非法运用】。**

# 第5章 系统可靠性分析与设计

## 易混淆点1：恢复块方法与N版本程序设计

**恢复块方法与N版本程序设计对比：**

|  |  |  |
| --- | --- | --- |
|  | 恢复块方法 | N版本程序设计 |
| 硬件运行环境 | 单机 | 多机 |
| 错误检测方法 | 验证测试程序 | 表决 |
| 恢复策略 | 后向恢复 | 前向恢复 |
| 实时性 | 差 | 好 |

## 易混淆点2：可靠性和可用性

**可靠性：**系统可靠性是系统在规定的时间内及规定的环境条件下，完成规定功能的能力，也就是系统无故障运行的概率。

**可用性:**系统可用性是指在某个给定时间点上系统能够按照需求执行的概率。

## 易混淆点3：可靠性指标



平均无故障时间 → (MTTF) MTTF=1/λ，λ为失效率

平均故障修复时间 → (MTTR) MTTR=1/μ，μ为修复率

平均故障间隔时间 → (MTBF) MTBF = MTTR + MTTF

系统可用性 → MTTF/(MTTR+MTTF)×100%

在实际应用中，一般MTTR很小，所以通常认为MTBF≈MTTF。

# 第6章 项目管理

## 易混淆点1：PERT图和Gantt图

**PERT（项目评估与评审技术）图**是一种图形化的网络模型，描述一个项目中任务和任务之间的关系，每个节点表示一个任务，通常包括任务编号、名称、开始和结束时间、持续时间和松弛时间。

**Gantt图**是一种简单的水平条形图，它以一个日历为基准描述项目任务，横坐标表示时间，纵坐标表示任务，图中的水平线段表示对一个任务的进度安排，线段的起点和终点对应在横坐标上的时间分别表示该任务的开始时间和结束时间，线段的长度表示完成该任务所需的时间。

PERT图主要描述不同任务之间的依赖关系；Gantt图主要描述不同任务之间的重叠关系。

## 易混淆点2：质量保证与质量控制

（1）**质量保证**一般是每隔一定时间（例如，每个阶段末）进行的，主要通过系统的质量审计和过程分析来保证项目的质量。

（2）**质量控制**是实时监控项目的具体结果，以判断它们是否符合相关质量标准，制订有效方案，以消除产生质量问题的原因。

（3）一定时间内质量控制的结果也是质量保证的质量审计对象。质量保证的成果又可以指导下一阶段的质量工作，包括质量控制和质量改进。

## 易混淆点3：总时差和自由时差

1）**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。**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。

（2）**自由时差：**

在不影响紧后活动的最早开始时间前提下，该活动的机动时间。

对于有紧后活动的活动，其自由时差等于所有紧后活动最早开始时间减本活动最早完成时间所得之差的最小值。

对于没有紧后活动的活动，也就是以网络计划终点节点为完成节点的活动，其自由时差等于计划工期与本活动最早完成时间之差。

对于网络计划中以终点节点为完成节点的活动，其自由时差与总时差相等。此外，由于活动的自由时差是其总时差的构成部分，所以，当活动的总时差为零时，其自由时差必然为零，可不必进行专门计算。

# 第7章 计算机组成与体系结构

## 易混淆点1：CISC与RISC（CISC：复杂指令集；RISC：精简指令集。）

**CISC与RISC对比：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指令系统类型 | 指令 | 寻址方式 | 实现方式 | 其它 |
| CISC（复杂） | 数量多，使用频率差别大，可变长格式 | 支持多种 | 微程序控制技术（微码） | 研制周期长 |
| RISC（精简） | 数量少，使用频率接近，定长格式，大部分为单周期指令，操作寄存器，只有Load/Store操作内存 | 支持方式少 | 增加了通用寄存器；硬布线逻辑控制为主；适合采用流水线 | 优化编译，有效支持高级语言 |

## 易混淆点2：流水线公式

**流水线执行时间（理论公式）：(t1+t2+..+tk)+(n-1)\*∆t**

**流水线执行时间（实践公式）：k\*∆t +(n-1)\*∆t**

**流水线吞吐率：**

**流水线最大吞吐率（即流水线周期的倒数）**

**流水线加速比：顺序执行时间/流水线执行时间**

## 易混淆点3：特殊的操作系统

|  |  |
| --- | --- |
| 分类 | 特点 |
| 批处理操作系统 | 单道批：一次一个作业入内存，作业由程序、数据、作业说明书组成  多道批：一次多个作业入内存，特点：多道、宏观上并行微观上串行 |
| 分时操作系统 | 采用时间片轮转的方式为多个用户提供服务，每个用户感觉独占系统  特点：多路性、独立性、交互性和及时性 |
| 实时操作系统 | 实时控制系统和实时信息系统  交互能力要求不高，可靠性要求高**（规定时间内响应并处理）** |
| 网络操作系统 | 方便有效共享网络资源，提供服务软件和有关协议的集合  主要的网络操作系统有：Unix、Linux和Windows Server系统 |
| 分布式操作系统 | 任意两台计算机可以通过通信交换信息  是网络操作系统的更高级形式，具有透明性、可靠性和高性能等特性 |
| 微机操作系统 | Windows：Microsoft开发的图形用户界面、多任务、多线程操作系统  Linux：免费使用和自由传播的类Unix操作系统，多用户、多任务、多线程和多CPU的操作系统 |
| 嵌入式操作系统 | 运行在智能芯片环境中  特点：微型化、**可定制（针对硬件变化配置）**、实时性、可靠性、**易移植性（HAL和BSP支持）** |

## 易混淆点4：互斥信号量和同步信号量的区分

信号量一般与PV操作结合在一起，对于同步模型的描述中，涉及的就是同步信号量，一般同步是多个进程之间通过PV操作进行制约，此时P操作和成对的V操作一般是位于不同的进程中的，而互斥模型中PV操作控制的是对资源的访问限制，此时P操作和成对的V操作一般出现在资源使用前和使用后，一般位于同一个进程中。

# 第8章 嵌入式系统

## 易混淆点1：嵌入式微处理器分类

（1）微处理器（Micro Processor Unit，MPU）：将微处理器装配在专门设计的电路板上，只保留与嵌入式应用有关的母板功能。微处理器一般以某一种微处理内核为核心，每一种衍生产品的处理器内核都是一样的，不同的是存储器和外设的配置和封装。

（2）微控制器（Micro Control Unit，MCU）：又称单片机。与MPU 相比MCU的最大优点在于单片化，体积大大减小，从而使功耗和成本下降，可靠性提高。

（3）信号处理器（Digital Signal Processor，DSP）：DSP 处理器对系统结构和指令进行了特殊设计（通常，DSP 采用一种哈佛结构），使其适合于执行 DSP 算法，编译效率高，指令执行速度也高。

（4）**图形处理器（Graphics Processing Unit，GPU）：**

GPU 是图形处理单元的缩写，是一种可执行渲染3D 图形等图像的半导体芯片（处理器）。

GPU 可用于个人电脑、工作站、游戏机和一些移动设备上做图像和图形相关运算工作的微处理器。

它可减少对 CPU的依赖，并进行部分原本 CPU的工作，尤其是在3D图形处理中，GPU 采用了核心技术（如： 硬件T&L、纹理压缩等）保证了快速3D渲染能力。

GPU目前已广泛应用于各行各业，GPU中集成了同时运行在GHz的频率上的成千上万个core，可以高速处理图像数据。**最新的GPU峰值性能可高达100 TFlops以上。**

（5）**片上系统（System on Chip，SoC）：**

追求产品系统最大包容的集成器件。

它是**一个产品**，是一个有专用目标的集成电路，其中包含完整系统并有嵌入软件的全部内容。

同时它又是**一种技术**，用以实现从确定系统功能开始，到软/硬件划分，并完成设计的整个过程。

成功实现了软硬件的无缝结合，直接在微处理器片内嵌入操作系统的代码模块。

减小了系统的体积和功耗、提高了可靠性和设计生产效率。

狭义角度：信息系统核心的芯片集成，是将系统关键部件集成在一块芯片上；

广义角度：**SoC是一个微小型系统**，如果说中央处理器（CPU）是大脑，那么SoC就是包括大脑、心脏、眼睛和手的系统。

## 易混淆点2：哈佛结构和冯·诺依曼结构

|  |  |  |  |
| --- | --- | --- | --- |
| **体系结构分类** | 定义 | 特点 | 典型应用 |
| **冯·诺依曼结构** | 冯·诺依曼结构也称普林斯顿结构，是一种将程序指令存储器和数据存储器合并在一起的存储器结构。 | 指令与数据存储器合并在一起。  指令与数据都通过相同的数据总线传输。 | 一般用于PC处理器，如I3、I5、I7处理器。  **注：常规计算机属于冯·诺依曼结构** |
| **哈佛结构** | 哈佛结构是一种并行体系结构，它的主要特点是将程序和数据存储在不同的存储空间中，即程序存储器和数据存储器是两个独立的存储器，每个存储器独立编址、独立访问。 | 指令与数据分开存储，可以并行读取，有较高数据的吞吐率。  有4条总线：指令和数据的数据总线与地址总线。 | 一般用于嵌入式系统处理器。  **注：DSP属于哈佛结构** |

## 易混淆点3：总线的分类

（1）**从功能上来对总线进行划分：数据总线、地址总线和控制总线**

（2）**从数据传输的方式划分为并行总线和串行总线**

串行总线：数据是一位一位地进行传输的，在传输中每一位数据都占据一个固定的时间长度。常见串行总线：RS232、SPI、I2C、USB、CAN、IEEE 1394等。【长距离，传输波特率可调整，正确性依赖于校验码，数据传输方式可以使用多种】

并行总线：将数据字节的各位用多条数据线同时进行传送。常见并行总线ISA、PCI、VME等。【短距离】

## 易混淆点4：嵌入式实时操作系统调度算法

优先级调度算法：系统为每个任务分配一个**相对固定的优先顺序**。

抢占式优先级调度算法：根据**任务的紧急程度**确定该任务的优先级。大多数RTOS调度算法都是抢占方式（可剥夺方式）。

最早截止期调度算法（EDF算法）：**根据任务的截止时间头端来确定其优先级**，对于时间期限最近的任务，分配最高的优先级。

最晚截止期调度算法：**根据任务的截止时间末端来确定其优先级**，对于时间期限最近的任务，分配最高的优先级。

## 易混淆点5：微内核与单体内核对比

|  |  |  |  |
| --- | --- | --- | --- |
|  | **实质** | **优点** | **缺点** |
| **单体内核** | 将图形、设备驱动及文件系统等功能全部在内核中实现，运行在内核状态和同一地址空间。 | 减少进程间通信和状态切换的系统开销，获得较高的运行效率。 | 内核庞大，占用资源较多且不易剪裁。  系统的稳定性和安全性不好。 |
| **微内核**  **【鸿蒙操作系统】** | 只实现基本功能，将图形系统、文件系统、设备驱动及通信功能放在内核之外。 | 微内核系统**结构相当清晰，有利于协作开发。**  内核精练，便于**剪裁和移植（灵活性、可扩展性）**。  系统服务程序运行在用户地址空间，**系统的可靠性、稳定性和安全性**较高。  可用于**分布式系统**。 | 用户状态和内核状态需要频繁切换，从而导致系统效率不如单体内核，**性能偏低。** |

## 易混淆点6：嵌入式数据库分类

|  |  |  |
| --- | --- | --- |
|  | 定义 | 特点 |
| **基于内存的数据库MMDB** | 基于内存的数据库系统是**实时系统和数据库系统的有机结合。** | 内存数据库是支持实时事务的最佳技术，其本质特征是**以其“主拷贝”或“工作版本”常驻内存，即活动事务只与实时内存数据库的内存拷贝打交道。** |
| **基于文件的数据库FDB** | 基于文件的数据库系统就是以文件方式存储数据库数据，即**数据按照一定格式储存在磁盘中。**使用时由应用程序**通过相应的驱动程序甚至直接对数据文件进行读写。** | 这种数据库的访问方式是被动式的，只要了解其文件格式，任何程序都可以直接读取，因此它的**安全性很低**。虽然文件数据库存在诸多弊端，但是，**可以满足嵌入式系统在空间、时间等方面的特殊要求。** |
| **基于网络的数据库NDB** | 基于网络的数据库系统是**基于手机4G/5G的移动通信基础之上**的数据库系统，**在逻辑上可以把嵌入式设备看作远程服务器的一个客户端。** | 嵌入式网络数据库系统的特点是：  **无需解析SQL语句；支持更多的SQL操作；客户端小、无须支持可剪裁性；有利于代码重用。** |

# 第9章 数据库系统

## 易混淆点1：外模式，模式和内模式

**三级模式：外模式**对应视图，**模式**（也称为概念模式）对应数据库表，**内模式**对应物理文件。

**两层映像：**外模式-模式映像，模式-内模式映像；两层映像可以保证数据库中的数据具有较高的逻辑独立性和物理独立性。

**物理独立性：**即数据库的内模式发生改变时，应用程序不需要改变。

**逻辑独立性：**即逻辑结构发生改变时，用户程序不需要改变。（逻辑独立性比物理独立性更难实现）

## 易混淆点2：候选码，主码，外码，全码

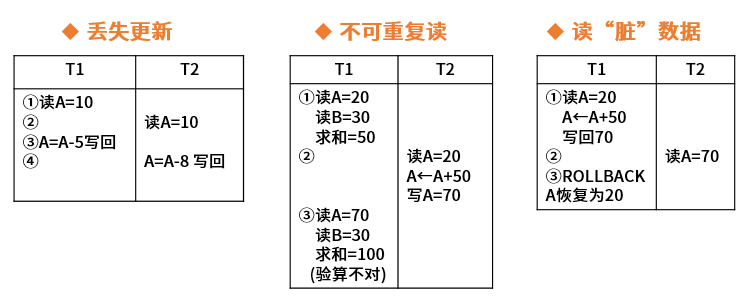
**候选键/候选码/码：**如果在一个关系中，存在一个属性（或属性组）都能用来唯一标识该关系的元组，并不含有多余的属性，这个属性（或属性组）称为该关系的候选码或候选键。候选键可以是单属性也可以是多属性集合，候选键可以是一个也可以有多个。

**主键/主码：**可以有多个不同的候选键，在其中任选一个作为主键。其值能唯一地标识元组的一个或多个属性，称为主码或关键字。组成候选码的属性就是主属性，其他为非主属性。

**外键：**关系中的某个属性（或属性组）不是该关系的主码或只是主码的一部分，但却是另一个关系的主码时，该属性（或属性组）称为这个关系的外码。

**全码：**关系中的所有属性组成这个关系模式的候选码。

## 易混淆点3：丢失更新/丢失修改，不可重复读，读“脏”数据



## 易混淆点4：冷备份和热备份

**冷备份**也称为静态备份，是将数据库正常关闭，在停止状态下，将数据库的文件全部备份（复制）下来。

**热备份**也称为动态备份，是利用备份软件，在数据库正常运行的状态下，将数据库中的数据文件备份出来。

|  |  |  |
| --- | --- | --- |
| 优缺点  备份方式 | 优点 | 缺点 |
| **冷备份** | 非常快速的备份方法(只需复制文件)；容易归档(简单复制即可)；容易恢复到某个时间点上(只需将文件再复制回去)；能与归档方法相结合，做数据库“最佳状态”的恢复 ；低度维护，高度安全 | 单独使用时，只能提供到某一时间点上的恢复；在实施备份的全过程中，数据库必须要作备份而不能做其他工作；若磁盘空间有限，只能复制到磁带等其他外部存储设备上，速度会很慢；不能按表或按用户恢复 |
| **热备份** | 可在表空间或数据库文件级备份，备份的时间短；备份时数据库仍可使用；可达到秒级恢复(恢复到某一时间点上)；可对几乎所有数据库实体做恢复；恢复是快速的 | 不能出错，否则后果严重；若热备份不成功，所得结果不可用于时间点的恢复；因难于维护，所以要特别小心，不允许“以失败告终” |

## 易混淆点5：分区与分表

|  |  |  |
| --- | --- | --- |
|  | 分区 | 分表 |
| 共性 | **1、都针对数据表**  **2、都使用了分布式存储**  **3、都提升了查询效率**  **4、都降低数据库的频繁I/O压力值** | |
| 差异 | **逻辑上还是一张表** | **逻辑上已是多张表** |

## 易混淆点6：分布式透明性

**分片透明性：**分不分片，用户感受不到（不关心如何分片存储）。

（水平分片：按记录分；垂直分片：按字段分；混合分片）

**位置透明性：**数据存放在哪里，用户不用管（用户无需知道数据存放的物理位置）。

**局部数据模型透明性（逻辑透明）：**用户或应用程序无需知道局部场地使用的是哪种数据模型。

# 第10章 计算机网络

## 易混淆点1：递归查询和迭代查询

**递归查询：**服务器必需回答目标IP与域名的映射关系。

**迭代查询：**服务器收到一次迭代查询回复一次结果，这个结果不一定是目标IP与域名的映射关系，也可以是其它DNS服务器的地址。

查询过程：

**主机向本地域名服务器的查询一般采用的都是递归查询：**

如果主机所询问的本地域名服务器不知道被查询域名的 IP 地址，那么本地域名服务器就以 DNS 客户的身份，向其他根域名服务器继续发出查询请求报文。

**本地域名服务器向根域名服务器的查询通常采用迭代查询：**

本地域名服务器向根域名服务器的查询通常是采用迭代查询。当根域名服务器收到本地域名服务器的迭代查询请求报文时，要么给出所要查询的 IP 地址，要么告诉本地域名服务器：“你下一步应当向哪一个域名服务器进行查询”。然后让本地域名服务器进行后续的查询。

根服务器或者流量较大的域名服务器都不使用递归查询，其原因也很简单，大量的递归查询会导致服务器过载。

## 易混淆点2：逻辑网络设计和物理网络设计的产出

（1）逻辑网络设计

（拓扑结构、网络地址分配、安全规划、网络管理、选择路由协议、设备命名规则。）

逻辑网络设计利用需求分析和现有网络体系分析的结果来设计逻辑网络结构，最后得到一份逻辑网络设计文档，输出内容包括以下几点：

逻辑网络设计图、IP地址方案、安全方案、招聘和培训网络员工的具体说明、对软硬件、服务、员工和培训的费用初步估计。

（2）物理网络设计

（设备的具体物理分布、运行环境等确定，设备选型、结构化布线、机房设计）

物理网络设计是对逻辑网络设计的物理实现，通过对设备的具体物理分布、运行环境等确定，确保网络的物理连接符合逻辑连接的要求。输出内容如下：

网络物理结构图和布线方案、设备和部件的详细列表清单、软硬件和安装费用的估算、安装日程表，详细说明服务的时间以及期限、安装后的测试计划、用户的培训计划

# 第11章 知识产权与标准化

## 易混淆点1：各法律法规的保护对象和范围以及保护期限

|  |  |  |
| --- | --- | --- |
| 法律法规名称 | 保护对象及范围 | 注意事项 |
| 著作权法 | 著作权  文学、绘画、摄影等作品 | 1、不需要申请，作品完成即开始保护  2、绘画或摄影作品原件出售（赠予）著作权还归原作者，原件拥有者有：所有权、展览权。 |
| 软件著作权法  计算机软件保护条例 | 软件著作权  软件作品 | 1、不需要申请，作品完成即开始保护  2、登记制度便于举证 |
| 专利法 | 专利权 | 需要申请，专利权有效期是从申请日开始计算 |
| 商标法 | 商标权 | 需要申请，核准之日起商标受保护 |
| 反不正当竞争法 | 商业秘密权 | 1、商业秘密包括技术与经营两个方面  2、必须有保密措施才能认定商业秘密 |

## 易混淆点2：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 | |

## 易混淆点3：常见侵权与合理引用区分

|  |  |
| --- | --- |
| 不侵权 | 侵权 |
| * 个人学习、研究或者欣赏； * 适当引用； * 公开演讲内容 * 用于教学或科学研究 * 复制馆藏作品； * 免费表演他人作品； * 室外公共场所艺术品临摹、绘画、摄影、录像； * 将汉语作品译成少数民族语言作品或盲文出版。 | * 未经许可，发表他人作品； * 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； * 未参加创作，在他人作品署名； * 歪曲、篡改他人作品的； * 剽窃他人作品的； * 使用他人作品，未付报酬； * 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，可扫码添加希赛嘉儿老师，申请入群

