

NEW PLAN

# 软考高级架构师

• — 段 新 征 程 •





# 大纲介绍





## 历年真题考点分析

| 时间               | 所属范围   | 考察知识点                        |
|------------------|--------|------------------------------|
| 2023 年 11 月 试题 1 | 软件架构   | 大数据架构 Lambda 和 Kappa         |
| 2023 年 11 月 试题 2 | 系统开发   | SysML 需求图和用例图、需求图七类关系等       |
| 2023 年 11 月 试题 3 | 数据库    | 读写分离架构、redis 缓存、主从复制         |
| 2023 年 11 月 试题 4 | Web 应用 | Hibernate 架构、数据持久层、jwt       |
| 2023 年 11 月 试题 5 | 嵌入式    | 数字孪生概念、技术选择、架构图填空            |
| 2022 年 11 月 试题 1 | 软件架构   | 架构风格，质量属性                    |
| 2022 年 11 月 试题 2 | 系统开发   | 结构化分析：数据流图、ER 图、数据字典         |
| 2022 年 11 月 试题 3 | 嵌入式    | 宇航装备架构、看图填空、故障分析             |
| 2022 年 11 月 试题 4 | 数据库    | 同步和异步、缓存分片、布隆过滤器             |
| 2022 年 11 月 试题 5 | Web 应用 | MQTT 协议、看图填空、云计算、边缘计算        |
| 2021 年 11 月 试题 1 | 软件架构   | 架构风格，质量属性                    |
| 2021 年 11 月 试题 2 | 系统开发   | 用例图、顺序图填空、模型对比               |
| 2021 年 11 月 试题 3 | 软件架构   | 数据定义分布管理涵义、基于 FACE 的架构（题目不全） |
| 2021 年 11 月 试题 4 | 数据库    | 反规范化设计方法、数据不一致、redis 同步      |
| 2021 年 11 月 试题 5 | Web 应用 | 云平台智能家居，看图填空，TCP/UDP 区别      |
| 2020 年 11 月 试题 1 | 软件架构   | 架构风格，质量属性                    |
| 2020 年 11 月 试题 2 | 数据库    | 逻辑设计、关系模式、主键、超类实体、派生属性       |
| 2020 年 11 月 试题 3 | 嵌入式    | 需求到架构映射、FACE 架构              |
| 2020 年 11 月 试题 4 | 数据库    | 内存数据库 redis，内存淘汰机制           |
| 2020 年 11 月 试题 5 | Web 应用 | 非功能性需求，SSM 框架，数据访问机制         |
| 2019 年 11 月 试题 1 | 软件架构   | 架构风格，质量属性                    |
| 2019 年 11 月 试题 2 | 系统开发   | 数据流图求实体、加工、补充数据流；系统流程图区别     |
| 2019 年 11 月 试题 3 | 嵌入式    | 信息物理系统三层结构概念、填空；三类安全威胁       |
| 2019 年 11 月 试题 4 | 数据库    | 数据库读写并发操作、key/value 方案探讨     |
| 2019 年 11 月 试题 5 | Web 应用 | 非功能性需求、分布式架构图、SQL 注入攻击       |



## 嵌入式系统讲课特点

1.嵌入式系统这道案例分析题是不推荐同学进行选择的题目，因为出题的范围和跨度太大，哪怕你本身就是做嵌入式的，都不一定能做；

2.老师针对嵌入式系统这部分的案例分析的讲课主要分为以下几个部分：

- 第16章针对嵌入式系统的一些知识，这些知识是非嵌入式专业人士也能理解的；
- 前期的一些知识点（书本的2.4）加上一些拓展，因为这些知识也在嵌入式的考试中出现过，主要是可靠性设计；
- 第16章针对鸿蒙系统架构的描述；
- 一道具有通用性参考价值的嵌入式考试真题；

3.如果你不是做嵌入式开发的，那么跟着老师的课程学习到这里就够了，如果你是做嵌入式开发的，那么建议去把下篇的16章的书本内容进行详细阅读，因为你是有可能在案例分析考试中选择嵌入式这道题的。

# 嵌入式系统-介绍

嵌入式系统 (Embedded System) 是为了特定应用而专门构建的计算机系统，其架构是随着嵌入式系统的逐步应用而发展形成的，嵌入式系统的发展主要分为下面五个阶段

1. 单片微型计算机阶段 (SCM)：功能单一，效率低下，存储容量小，比如计算器
2. 微控制器阶段 (MUC)：主要以嵌入式微处理器为基础，以简单操作系统为核心，性能弱，开销小，效率高，比如汽车电子设备、医疗器械设备
3. 片上系统 (SoC)：能够运行于各种不同类型的微处理器上，兼容性好，操作系统的内核小，效率高，比如麒麟芯片
4. 以互联网为基础的嵌入式系统：集成了网络，应用于网络环境中，比如智能家居
5. 在智能化、云技术推动下的嵌入式系统：面向智能服务的设备，比如智能穿戴、智能汽车

嵌入式系统主要由嵌入式微处理器(控制器 (MCU))、存储器 (RAM/ROM)、内(外)总线逻辑、定时/计数器 (Time)、看门狗电路、I/O接口(串口、网络、USB、JTAG等)和外部设备 (UART、LED等)等部件组成

嵌入式微处理器分类

- 微处理器 (MPU)
- 微控制器 (MCU)
- 信号处理器 (DSP)
- 图形处理器 (GPU)
- 片上系统 (SoC)

# 嵌入式系统-嵌入式操作系统

嵌入式系统的典型架构可概括为两种模式：

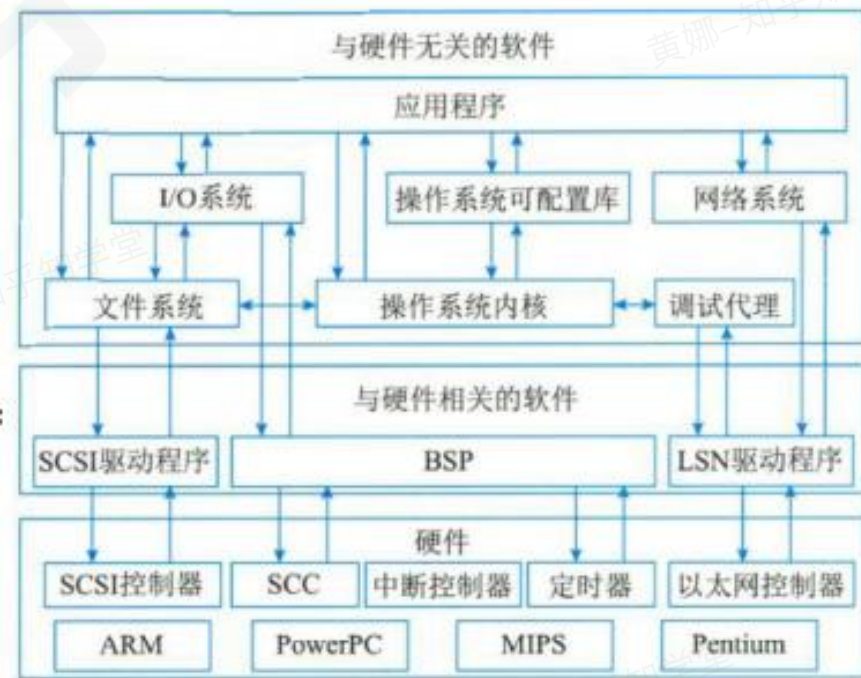
- **层次化模式架构**：是一种将嵌入式系统划分为多个层次的架构。每个层次都具有特定的功能，并与其他层次进行交互
- **递归模式架构**：是一种将嵌入式系统设计为具有自相似性的架构。在递归模式架构中，系统由相同或相似的子系统组成

嵌入式操作系统(EOS)是指用于嵌入式系统的操作系统。通常包括**与硬件相关的底层驱动软件、系统内核、设备驱动接口、通信协议、图形界面或标准化浏览器**等。

嵌入式操作系统特点：可裁剪性、可移植性、强实时性、强紧凑性、高质量代码、强定制性、标准接口、强稳定性、**弱交互性**、强确定性、操作简洁方便、较强的硬件适应性、可固化性。

从嵌入式操作系统体系架构看，主要存在**4种结构整体结构、层次结构、客户/服务器结构和面向对象结构**。

整体结构也称为**模块结构或无序结构**，它是基于**结构化程序设计**的一种软件设计方法，其架构如右图：





## 嵌入式系统-实时操作系统

嵌入式实时系统是一种完全嵌入受控器件内部，为特定应用而设计的专用计算机系统。在嵌入式实时系统中，要求系统在投入运行前即具有确定性和可预测性。

- 可预测性是指嵌入式实时系统能够在预期的时限内完成操作，并且能够提前预测其行为；
- 确定性是指嵌入式实时系统在给定输入和状态条件下，其行为始终是一致的，并且能够在预期的时限内完成操作。

实时操作系统(RTOS)的特点：

- 当外界事件或数据产生时，能够接受并以足够快的速度予以处理，其处理的结果又能在规定的时间之内来控制生产过程或对处理系统做出快速响应，并控制所有实时任务协调一致运行。因而，提供及时响应和高可靠性是其主要特点。
- 实时操作系统有硬实时和软实时之分，硬实时要求在规定的时间内必须完成操作，这是在操作系统设计时保证的；软实时则只要按照任务的优先级，尽可能快地完成操作即可。



# 嵌入式系统-实时操作系统

实时操作系统的特征:

## ✓ 高精度计时系统

计时精度是影响实时性的一个重要因素。在实时应用系统中，经常需要精确确定实时地操作某个设备或执行某个任务，或精确的计算一个时间函数。这些不仅依赖于一些硬件提供的时钟精度，也依赖于实时操作系统实现的高精度计时功能。

## ✓ 多级中断机制

一个实时应用系统通常需要处理多种外部信息或事件，但处理的紧迫程度有轻重缓急之分。有的必须立即作出反应，有的则可以延后处理。因此，需要建立多级中断嵌套处理机制，以确保对紧迫程度较高的实时事件进行及时响应和处理。

## ✓ 实时调度机制

实时操作系统不仅要及时响应实时事件中断，同时也要及时调度运行实时任务。但是，处理机调度并不能随心所欲的进行，因为涉及到两个进程之间的切换，只能在确保“安全切换”的时间点上进行，实时调度机制包括两个方面，一是在调度策略和算法上保证优先调度实时任务；二是建立更多“安全切换”时间点，保证及时调度实时任务。



## 嵌入式系统-嵌入式数据库

嵌入式数据库是嵌入到应用程序进程中的数据库，为了消除和客户机服务器配置的开销，所以一般都是轻量级的，运行的时候，只需要较少的内存，允许通过SQL来管理数据，尽可能不需要进行配置，嵌入式数据库要求很高的实时性和可靠性，和传统的数据库相比，嵌入式数据库的特点如下：

- 嵌入式
- 实时性
- 移动性
- 伸缩性

### 嵌入式数据库的分类

- 基于内存的数据库系统
- 基于文件的数据库系统
- 基于网络的数据库系统

### 嵌入式数据库的主要功能

- 足够高效的数据存储机制
- 数据安全控制（锁机制）
- 实时事务管理机制
- 数据库恢复机制

表16-3 主要嵌入式数据库特点比较

| 名称                                | 简介                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SQLite                            | SQLite诞生于2000年5月，这几年增长势头迅猛无比，目前版本是3.3.8。其主要特点是： <ul style="list-style-type: none"><li>● 无须安装配置，应用程序只须携带一个动态链接库</li><li>● 非常小巧，For Windows 3.3.8版本的DLL文件才374KB</li><li>● ACID事务支持，ACID即原子性、一致性、隔离性、和持久性 (Atomic Consistent Isolated 和 Durable)</li><li>● 数据库文件可以在不同字节顺序的机器间自由的共享，比如可以直接从Windows移植到Linux或M A C</li><li>● 支持数据库大小至2TB</li></ul>                                                                                                           |
| Berkeley DB                       | Berkeley DB是由美国Sleepycat Software公司开发的一套开放源码的嵌入式数据库，它于1991年发布，目前Sleepycat 现已被甲骨文（ORACLE）公司收购。其主要特点是： <ul style="list-style-type: none"><li>● 嵌入式，无需安装配置</li><li>● 为多种编程语言提供了API接口，其中包括C C++ Java Perl TCL Python和PHP等</li><li>● 轻便灵活。它可以运行于几乎所有的UNIX和Linux系统及其变种系统、Windows操作系统以及多种嵌入式实时操作系统之下</li><li>● 可伸缩。它的Database library才几百“KB”大小，但它能够管理规模高达256TB的数据库。它支持高并发度，成千上万个用户可同时操作同一个数据库</li></ul>                                                       |
| Firebird 嵌入服务器版 (Embedded Server) | 从Interbase开源衍生出了Firebird版本。虽然它的体积是Interbase的几十分之一，但功能并无丢弃。为了体现Firebird短小精悍的特色，开发小组在增加了超级服务器版本之后，又增加了嵌入版本，最新版本为2.0。其主要特点是： <ul style="list-style-type: none"><li>● 数据库文件与Firebird网络版本完全兼容，差别仅在于连接方式不同，可以实现零成本迁移</li><li>● 数据库文件仅受操作系统的限制，且支持将一个数据库分割成不同文件，突破了操作系统最大文件的限制，提高了I/O吞吐量</li><li>● 完全支持SQL-92标准，支持大部分SQL-99标准功能</li><li>● 丰富的开发工具支持，绝大部分基于Interbase的组件，可以直接使用于Firebird</li><li>● 支持事务、存储过程、触发器等关系数据库的所有特性</li><li>● 可自己编写扩展函数 (UDF)</li></ul> |
| eXtremeDB                         | eXtremeDB 是美国McObject公司开发的一款内存式嵌入式数据库。2013年3月推出5.0版。其主要特点是： <ul style="list-style-type: none"><li>● 内存数据库，实时性强，安全可靠</li><li>● 可最小化支持持久数据所必需的资源</li><li>● 保持极小的必要堆空间</li><li>● 维持极小的代码体积</li><li>● 通过紧密的集成持久存储和宿主应用程序语言消除额外的代码层</li><li>● 提供对动态数据结构的本地支持</li></ul>                                                                                                                                                                                      |



## 嵌入式系统-可靠性基本概念

软件可靠性：是软件产品在规定的条件下和规定的时间区间完成规定功能的能力。

软件可靠性和硬件可靠性区别

- 复杂性：软件复杂性比硬件高，大部分失效来自于软件失效。
- 物理退化：硬件失效主要是物理退化所致，软件不存在物理退化。
- 唯一性：软件是唯一的，软件复制不改变软件本身，而任何两个硬件不可能绝对相同。
- 版本更新周期：硬件更新较慢，软件更新较快。

软件可靠性的定量描述

- 失效概率：软件运行初始时为0，随着时间增加单调递增，不断趋向于1。
- 可靠度：软件系统在规定的条件下、规定的时间内不发生失效的概率。等于1-失效概率。
- 平均失效前时间(MTTF)：平均无故障时间，发生故障前正常运行的时间。
- 平均恢复前时间(MTTR)：平均故障修复时间，发生故障后的修复时间。
- 平均故障间隔时间(MTBF)：失效或维护中所需的平均时间，包括故障时间以及检测和维护设备的时间。

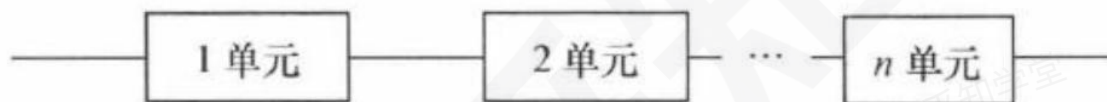
$MTBF=MTTF+MTTR$ 。



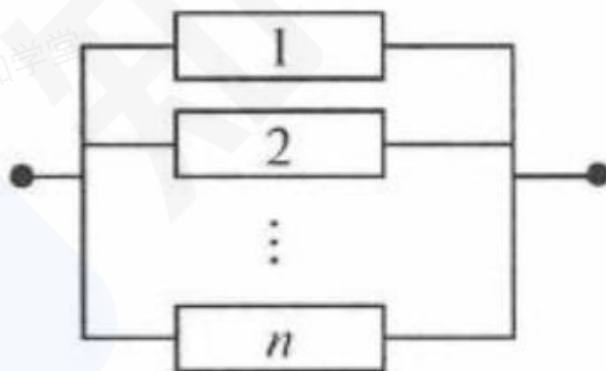
## 嵌入式系统-可靠性基本概念

串并联系统可靠性：无论什么系统，都是由多个设备组成的，协同工作，而这多个设备的组合方式可以是串联、并联，也可以是混合模式，假设每个设备的可靠性为 $R_1, R_2, \dots, R_n$ ，则不同的系统的可靠性公式如下：

- 串联系统，一个设备不可靠，整个系统崩溃，整个系统可靠性： $R = R_1 * R_2 * \dots * R_n$ 。



- 并联系统，所有设备都不可靠，整个系统才崩溃，整个系统可靠性： $R = 1 - (1 - R_1) * (1 - R_2) * \dots * (1 - R_n)$ 。





## 嵌入式系统-可靠性设计

为什么需要可靠性设计：实践证明，保障软件可靠性最有效、最经济、最重要的手段是在**软件设计阶段**采取措施进行可靠性控制。为了从根本上提高软件的可靠性，人们就提出了可靠性设计

可靠性设计其实就是在**常规的软件设计**中，应用各种方法和技术，使程序设计在兼顾用户的功能和性能需求的同时，**全面满足软件的可靠性要求**。

软件可靠性设计原则：

- **软件可靠性设计是软件设计的一部分**，必须在软件的总体设计框架中使用，并且不能与其他设计原则相冲突。
- 软件可靠性设计在满足提高软件质量要求的前提下，以**提高和保障软件可靠性为最终目标**
- 软件可靠性设计应**确定软件的可靠性目标**，不能无限扩大化，并且排在功能度、用户需求和开发费用之后考虑。

软件可靠性设计技术主要有**容错设计、检错设计和降低复杂度设计**等技术。



## 嵌入式系统-可靠性设计

容错：指系统在运行过程中发生一定的硬件故障或软件错误时，仍能保持正常工作而不影响正确结果的一种性能或措施。

容错技术主要有：冗余设计、恢复块设计和N版本程序设计这三种。

### 冗余设计

- 冗余设计是指在系统中增加冗余的硬件或软件资源，以在其中一个或多个资源发生故障时，由冗余资源替代其工作，从而保证系统的正常运行。
- 冗余设计的常见方法包括：
  - 硬件冗余：复制关键硬件组件，例如CPU、内存、存储设备等，当其中一个或多个组件发生故障时，由冗余组件替代其工作。
  - 软件冗余：复制关键软件模块，当其中一个或多个模块发生故障时，由冗余模块替代其工作。
- 冗余设计的优点是简单易行，成本较低，但缺点是会增加系统的复杂性和开销。

### 恢复块设计

- 恢复块设计是指将程序划分成多个独立的恢复块，每个恢复块负责完成特定的功能。当某个恢复块发生故障时，可以重新执行该恢复块，以恢复系统的正常运行。恢复块设计的常见方法包括：
  - 回滚恢复：将系统状态回滚到故障发生前的某个时刻，然后重新执行程序。
  - 前向恢复：重新计算故障发生后的系统状态。
- 恢复块设计的优点是能够快速恢复系统，但缺点是需要额外的存储空间和计算资源

# 嵌入式系统-可靠性设计

- N版本程序设计：其设计思想是用N个具有相同功能的程序同时执行一项计算，结果通过投票机制来选择。其中N个版本的程序必须由不同的人独立设计，使用不同的方法、设计语言、开发环境和工具来实现，目的是减少N个版本的程序在表决点上相关错误的概率。
- 示例：假设我们正在开发一个用于自动飞行控制的软件，其中安全性至关重要。我们创建了两个独立开发的飞行控制软件版本，A 和B。这两个版本在不同的硬件上运行，并在相同的飞行条件下接收传感器数据并控制飞机。如果A 和B 在关键决策上产生了不一致的结果，一个决策模块会进行投票，选择大多数版本的输出，以确保飞机的安全。

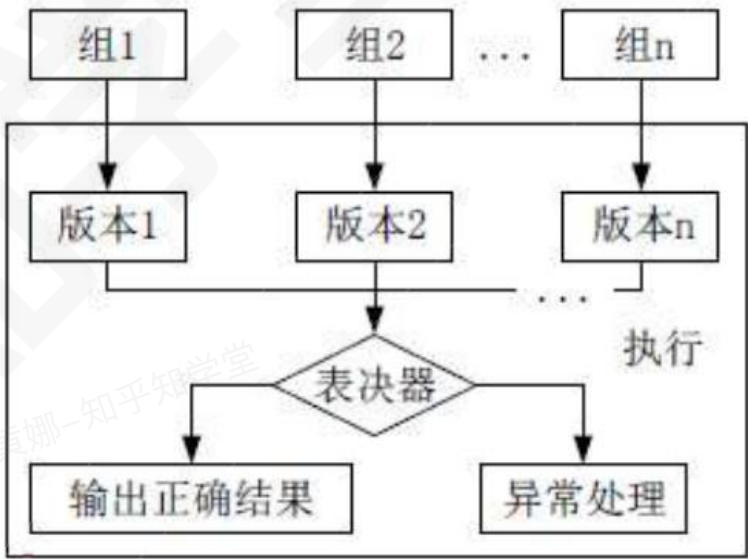


表 19-1 恢复块方法与N版本程序设计的比较

|        | 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        |



## 嵌入式系统-可靠性设计

检错：在软件发生故障后能及时发现并报警，提醒技术人员进行处理。检错的实现代价低于容错和冗余技术，但是它不能自己解决问题，需要人工干预，否则将无法正常运行

检错技术主要考虑如下几个点：**检测对象、检测延时、实现方式和处理方式。**

- 检测对象：包含检测点和检测内容，监测点就是容易出错的地方和出错影响大的地方，检测内容就是选取哪些有代表性、易于判断的地方；比如医学图像的分析对患者的健康和生命至关重要，因此检测和纠正可能的错误对于确保诊断的准确性至关重要。
- 检测延时：就是故障从发生到检测到的一个时间，在医疗领域，及早检测和纠正错误是至关重要的，因为延迟可能导致误诊或延误治疗。
- 实现方式：检测到错误的实现方式，一般都是直接返回检测的结果，结果超出合理范围就代表出现了问题。
- 处理方式：检测到了错误，具体的处理方式是什么，一般都是停止软件执行并报警处理；



## 嵌入式系统-可靠性设计

系统配置技术：在系统配置中加入响应的容错技术来提高系统的可靠性，主要分为**双机热备份技术**和**服务器集群技术**两种方式

- 双机容错技术：**是一种软硬件结合的容错应用方案。该方案是由两台服务器和一个外接共享磁盘阵列及相应的双机软件组成。**双机容错系统采用“心跳”方法保证主系统与备用系统的联系。所谓心跳，是指主从系统之间相互按照一定的时间间隔发送通信信号，表明各自系统当前的运行状态。一旦心跳信号表明主机系统发生故障，或者备用系统无法收到主系统的心跳信号，则系统的高可用性管理软件认为主系统发生故障，立即将系统资源转移到备用系统上，备用系统替代主系统工作，以保证系统正常运行和网络服务不间断。
- 集群技术就是**将多台计算机组织起来进行协同工作**，它是提高系统可用性和可靠性的一种技术。在集群系统中，**每台计算机均承担部分计算任务和容错任务**，当其中一台计算机出现故障时，系统使用集群软件将这台计算机从系统中隔出离去，通过各计算机之间的负载转嫁机制**完成新的负载分担**，同时向系统管理人员发出警报。集群系统通过功能整合和故障过渡，实现了系统的高可用性和可靠性。



## 嵌入式系统-可靠性设计

负载均衡是**集群系统中的一项重要技术**，可以提高集群系统的整体处理能力，也提高了系统的可靠性，最终目的是**加快集群系统的响应速度，提高客户端访问的成功概率**。集群的最大特征是**多个节点的并行和共同工作**，如何让所有节点承受的负荷平均，不出现局部过大负载或过轻负载的情况，是负载均衡的重要目的。

比较常用的负载均衡实现技术主要有以下几种：

- **基于特定软件的负载均衡(应用层)**。基于特定软件的负载均衡技术是在应用层实现的，需要在每个服务器上安装特定的负载均衡软件。这种技术的优点是灵活性和可扩展性好，可以根据应用的具体需求进行配置。缺点是需要额外的软件开销，并且可能存在性能瓶颈。比如Nginx、HAProxy
- **基于DNS的负载均衡属于传输层负载均衡技术**。基于DNS的负载均衡技术是在传输层实现的，通过修改DNS服务器的解析结果来将请求分发到不同的服务器。这种技术的优点是简单易行，不需要在服务器上安装额外的软件。缺点是灵活性较差，并且可能存在性能瓶颈。假设一家在线新闻网站使用基于DNS的负载均衡。该网站在DNS服务器中为其主机名配置了多个IP地址，对应不同的服务器。当用户通过域名访问网站时，DNS服务器会以轮询或其他策略返回不同的IP地址，将用户引导到不同的服务器上，以均衡流量负载。
- **基于NAT的负载均衡**。是通过NAT设备来实现的，NAT设备可以修改数据包的源IP地址或端口号，从而将请求分发到不同的服务器。这种技术的优点是易于部署和管理，并且可以提供NAT功能。缺点是灵活性较差，并且可能存在性能瓶颈。例如一家企业使用基于NAT的负载均衡来管理其内部服务器。它配置了一个公共IP地址，然后将来自外部的连接请求动态映射到内部服务器的不同私有IP地址上。这样，外部用户的请求会被引导到不同的内部节点，以均衡负载并确保高可用性。
- **反向代理负载均衡**。在应用服务器前面部署一个反向代理服务器，反向代理服务器可以将请求分发到不同的应用服务器，并提供额外的功能，例如缓存、压缩、安全等。这种技术的优点是灵活性和可扩展性好，并且可以提供额外的功能。缺点是增加了一层代理，可能会降低性能。
- **混合型负载均衡**。混合型负载均衡技术是将两种或多种负载均衡技术结合在一起使用，以充分发挥每种技术的优势。例如，可以将基于DNS的负载均衡技术与基于反向代理的负载均衡技术结合在一起使用，以实现更灵活、更可扩展的负载均衡方案。



## 鸿蒙操作系统架构案例分析

鸿蒙(HarmonyOS)整体采用分层的层次化设计,从下向上依次为:内核层、系统服务层、框架层和应用层。系统功能按照“系统”→“子系统”→“功能/模块”逐级展开,在多设备部署场景下,支持根据实际需求裁剪某些非必要的子系统或功能/模块

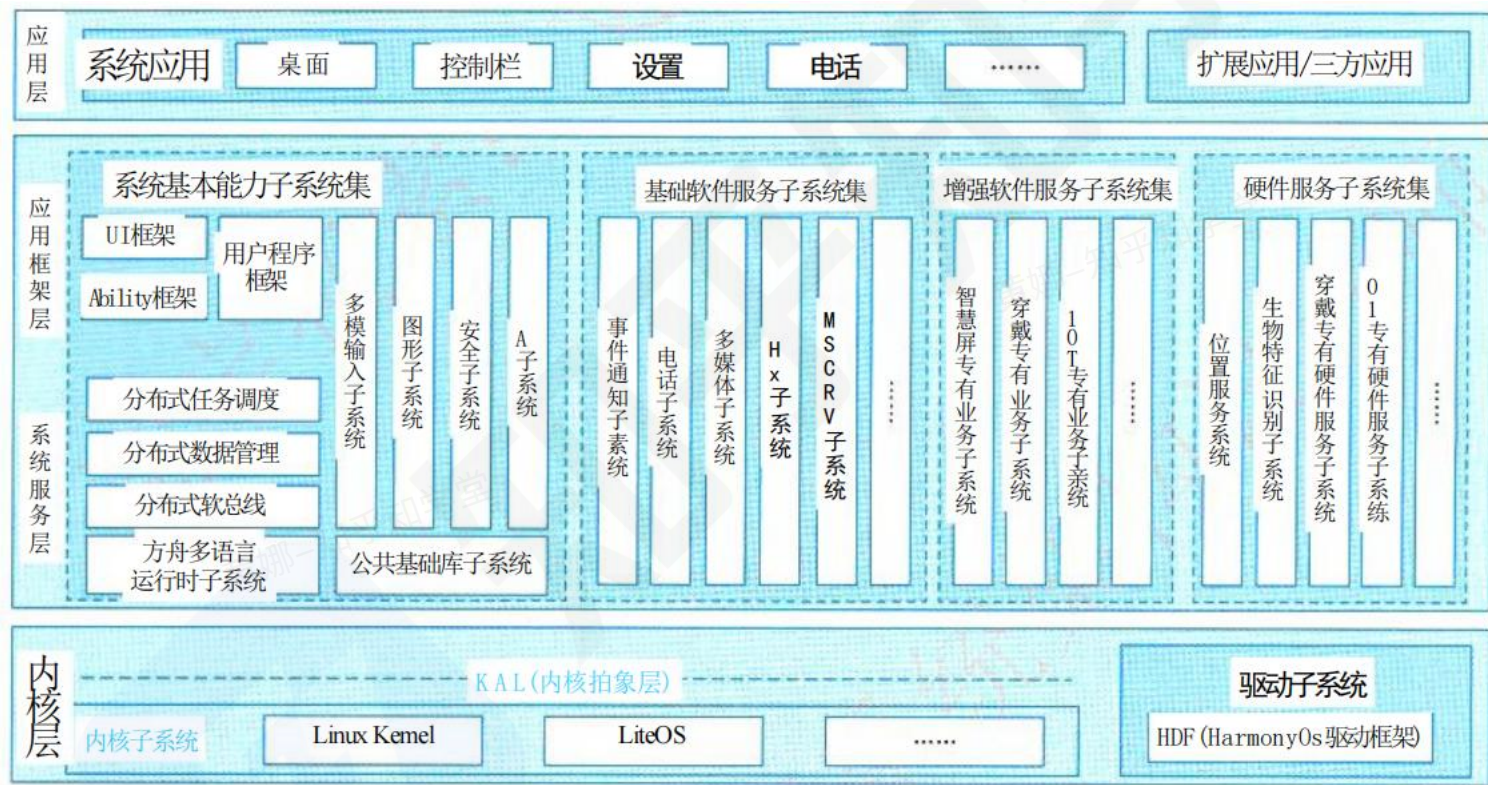


图16-27鸿蒙操作系统的层次化架构



## 鸿蒙操作系统架构案例分析

1)内核层：主要由内核子系统和驱动子系统组成。

- 内核子系统：HarmonyOS采用多内核设计，支持针对不同资源受限设备选用适合的OS内核。内核抽象层通过屏蔽多内核差异，对上层提供基础的内核能力。
- 驱动子系统：提供统一外设访问能力和驱动开发、管理框架。

2)系统服务层：是HarmonyOS的核心能力集合，通过框架层对应用程序提供服务。该层包含4个部分：

- 系统基本能力子系统集：为分布式应用在HarmonyOS多设备上的运行、调度、迁移等操作提供了基础能力。
- 基础软件服务子系统集：为HarmonyOS提供公共的、通用的软件服务。
- 增强软件服务子系统集：为HarmonyOS提供针对不同设备的、差异化的能力增强型软件服务。
- 硬件服务子系统集：为HarmonyOS提供硬件服务。

3)框架层：为HarmonyOS的应用程序提供了Java/C/C++/JS等多语言的用户程序框架和Ability框架，以及各种软硬件服务对外开放的多语言框架API；同时为采用HarmonyOS的设备提供了C/C++/JS等多语言的框架API，不同设备支持的API与系统的组件化裁剪程度相关。

4)应用层：包括系统应用和第三方非系统应用。HarmonyOS的应用由一个或多个FA(Feature Ability)或PA(Particle Ability)组成。其中，FA有UI界面，提供与用户交互的能力；而PA无UI界面，提供后台运行任务的能力以及统一的数据访问抽象。



## 鸿蒙操作系统架构案例分析

鸿蒙操作系统架构具有4个技术特性：

- **分布式架构**首次用于终端OS，实现跨终端无缝协同体验。
- 确定时延引擎和高性能IPC技术实现系统天生流畅。
- 基于微内核架构重塑终端设备可信安全。
- 通过统一的IDE支撑**一次开发，多端部署**，实现跨终端生态共享。

在HarmonyOS架构中，重点关注于分布式架构所带来的优势，主要体现在下面四个方面：

- **分布式软总线**：是多种终端设备的统一基座，为设备之间的互联互通提供了统一的分布式通信能力；
- **分布式设备虚拟化平台**：可以实现不同设备的资源融合、设备管理、数据处理，多种设备共同形成一个超级虚拟终端。针对不同类型的任务，为用户匹配并选择能力合适的执行硬件；
- **分布式数据管理**：基于分布式软总线的能力实现应用程序数据和用户数据的分布式管理用户数据不再与单一物理设备绑定，业务逻辑与数据存储分离，应用跨设备运行时数据无缝衔接；
- **分布式任务调度**：构建统一的分布式服务管理（发现、同步、注册、调用）机制，支持对跨设备的应用进行远程启动、远程调用、远程连接以及迁移等操作，选择合适的设备运行分布式任务。

HarmonyOS 架构的系统安全性主要体现在搭载HarmonyOS 的分布式终端上，可以保证"**正确的人通过正确的设备，正确地使用数据**"。这里通过"分布式多端协同身份认证"来保证"正确的人"，通过"在分布式终端上构筑可信运行环境"来保证"正确的设备"，通过"分布式数据在跨终端流动的过程中，对数据进行分类分级管理"来保证"正确地使用数据"。

## 典型历年真题详解

阅读以下信息系统可靠性的问题，在答题纸上回答问题1至问题3。

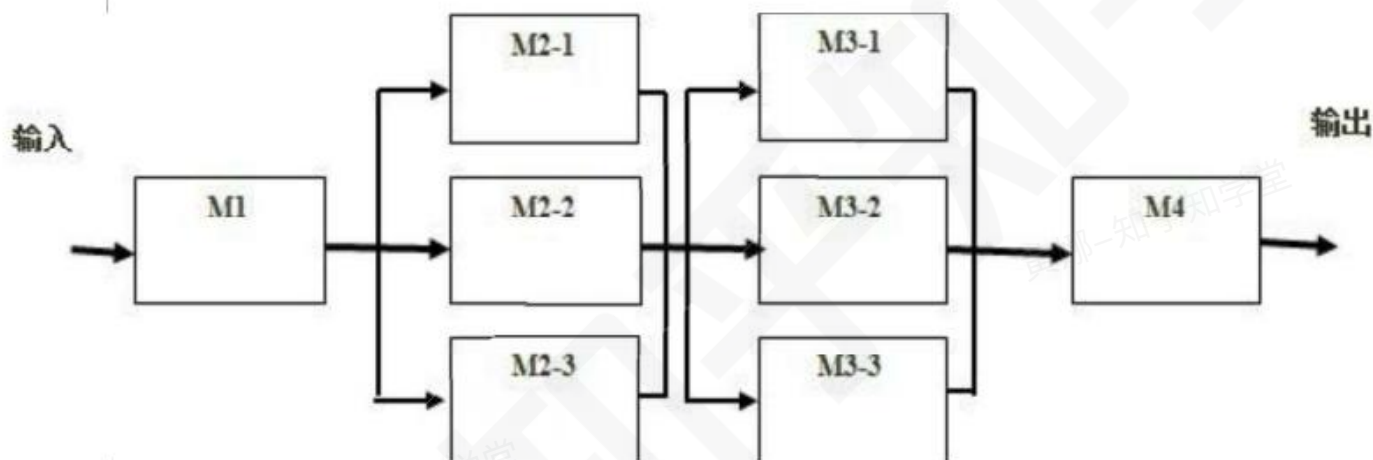
某软件公司开发一项基于数据流的软件，其系统的主要功能是对输入的数据进行多次分析、处理和加工，生成需要的输出数据。需求方对该系统的软件可靠性要求很高，要求系统能够长时间无故障运行。该公司将该系统设计交给王工负责。王工给出该系统的模块示意图如图1所示。王工解释：只要各个模块的可靠度足够高，失效率足够低，则整个软件系统的可靠性是有保证的。



李工对王工的方案提出了异议。李工认为王工的说法有两个问题：第一，即使每个模块的可靠度足够高，假设各个模块的可靠度均为0.99，但是整个软件系统模块之间全部采用串联，则整个软件系统的可靠度为 $0.99^4=0.96$ ，即整个软件系统的可靠度下降明显；第二，软件系统模块全部采用串联结构，一旦某个模块失效，则意味着整个软件系统失效。

## 典型历年真题详解

李工认为，应该在软件系统中采用冗余技术中的动态冗余或者软件容错的N版本程序设计技术，对容易失效或者非常重要的模块进行冗余设计，将模块之间的串联结构部分变为并联结构，来提高整个软件系统的可靠性。同时，李工给出了采用动态冗余技术后的软件系统模块示意图，如图2所示。



刘工建议，李工方案中M1和M4模块没有采用容错设计，但M1和M4发生故障有可能导致严重后果。因此，可以在M1和M4模块设计上采用检错技术，在软件出现故障后能及时发现并报警，提醒维护人员进行处理。

注：假设各个模块的可靠度均为0.99

## 典型历年真题详解

### 【问题1】(4分)

在系统可靠性中，可靠度和失效率是两个非常关键的指标，请分别解释其含义。

### 【问题2】(13分)

请解释李工提出的动态冗余和N版本程序设计技术，给出图1中模块M2采用图2动态冗余技术后的可靠度。请给出采用李工设计方案后整个系统可靠度的计算方法，并计算结果。

### 【问题3】(8分)

请给出检错技术的优缺点，并说明检测技术常见的实现方式和处理方式。



## 典型历年真题详解

### 试题答案

#### 【问题1】

可靠度就是系统在规定的条件下、规定的时间内不发生失效的概率。

失效率又称风险函数，也可以称为条件失效强度，是指运行至此时刻系统未出现失效的情况下，单位时间系统出现失效的概率。

#### 【问题2】

动态冗余又称为主动冗余，它是通过故障检测、故障定位及故障恢复等手段达到容错的目的。其主要方式是多重模块待机储备，当系统检测到某工作模块出现错误时，就用一个备用的模块来替代它并重新运行。各备用模块在其待机时，可与主模块一样工作，也可以不工作。前者叫热备份系统(双重系统)，后者叫冷备份系统(双工系统、双份系统)。

N版本程序设计是一种静态的故障屏蔽技术，其设计思想是用N个具有相同功能的程序同时执行一项计算，结果通过多数表决来选择。其中N个版本的程序必须由不同的人独立设计，使用不同的方法、设计语言、开发环境和工具来实现，目的是减少N个版本的程序在表决点上相关错误的概率。

M2采用动态冗余后的可靠度为：

$$R = 1 - (1 - 0.99)^3 = 0.999999$$

李工给出的方案同时采用了串联和并联方式，其计算方法为首先计算出中间M2和M3两个并联系统的可靠度，再按照串联系统的计算方法计算出整个系统的可靠度。

$$= 0.99 * 0.999999 * 0.999999 * 0.99 = 0.98$$



## 典型历年真题详解

### 【问题3】

检错技术实现的代价一般低于容错技术和冗余技术，但有一个明显的缺点，就是不能自动解决故障，出现故障后如果不进行人工干预，将最终导致软件系统不能正常运行。

检错技术常见的实现方式：最直接的一种实现方式是判断返回结果，如果返回结果超出正常范围，则进行异常处理；计算运行时间也是一种常用技术，如果某个模块或函数运行时间超过预期时间，可以判断出现故障；还有置状态标志位等多种方法，自检的实现方式需要根据实际情况来选用。

检错技术的处理方式：大多数都采用“查处故障-停止软件运行-报警”的处理方式。但根据故障的不同情况，也有采用不停止或部分停止软件系统运行的情况，这一般由故障是否需要实时处理来决定。

T H E E N D

功不唐捐，玉汝于成！

• 开 启 新 征 程 •

