芝士架构公共知识红宝书

2026 年 10 月终稿 - 芝士架构凯恩编辑整理 v6.0.0

系统架构设计师公共基础知识



知识产权备案盗版封号

目录

芝士架构公共知识红宝书 1
目录 2
前言 8
使用说明 9
常见问题 10
在线刷题 13
红宝书 6.0 章节安排思路 14
1. 软件工程概述（重点★★★★★） 15
1.1. 软件工程相关定义（次重点★★★☆☆） 15
1.2. 软件开发方法（重点★★★★★） 15
1.3. 软件过程/生命周期模型（重点★★★★★） 24
1.4. 软件过程管理（次重点★★★☆☆） 33
2. 需求工程（超级重点★★★★★） 35
2.1. 需求获取（次重点★★★☆☆） 36
2.2. 需求分析简介（非重点☆☆☆☆☆） 37
2.3. 结构化分析方法/建模 SA（超级重点★★★★★） 37
2.4. 面向对象分析方法/建模（超级重点★★★★★） 43
2.5. 建模语言（重点★★★★★） 49
2.6. 需求定义（非重点☆☆☆☆☆） 56
2.7. 需求验证（非重点☆☆☆☆☆） 56
2.8. 需求管理（重点★★★★★） 57

3. 系统设计（超级重点★★★★★）.....62
3.1. 系统设计在软件工程中的位置（非重点★☆☆☆☆）.....62
3.2. 系统设计概述（次重点★★★☆☆）.....63
3.3. 处理流程设计概述（次重点★★★☆☆）.....64
3.4. 面向对象设计（重点★★★★★）.....69
3.5. 结构化设计（重点★★★★★）.....71
3.6. 设计模式（次重点★★★☆☆）.....74
4. 系统架构设计（超级重点★★★★★）.....75
4.1. 基本定义（重点★★★★★）.....76
4.2. 软考中的 4+1 视图（重点★★★★★）.....76
4.3. 软件架构设计与生命周期（非重点☆☆☆☆☆）.....77
4.4. 基于架构的软件开发方法 ABSD（次重点★★★☆☆）.....78
4.5. 软件架构风格（超级重点★★★★★）.....80
4.6. 软件架构复用（重点★★★★★）.....84
4.7. 特定领域软件体系结构 DSSA（次重点★★★☆☆）.....85
4.8. 系统质量属性与架构评估（重点★★★★★）.....86
5. 软件测试（次重点★★★☆☆）.....96
5.1. 测试分类方法（重点★★★★★）.....96
5.2. 功能测试（次重点★★★☆☆）.....100
5.3. 性能测试（次重点★★★☆☆）.....100
5.4. 安全测试（次重点★★★☆☆）.....101
5.5. 软件测试模型 W/H/X 模型（次重点★★★☆☆）.....102
6. 系统运行与维护（次重点★★★☆☆）.....102

6.1. 运维技术指标（次重点★★★☆☆）.....102
6.2. 系统运行管理（非重点☆☆☆☆☆）.....103
6.3. 系统故障管理（非重点☆☆☆☆☆）.....104
6.4. 软件系统维护（次重点★★★☆☆）.....105
6.5. 遗留系统处置（次重点★★★☆☆）.....106
6.6. 新旧系统转换（次重点★★★☆☆）.....107
7. 系统可靠性（次重点★★★☆☆）.....108
7.1. 软件可靠性设计（次重点★★★☆☆）.....108
7.2. 软件可靠性的定量描述（重点★★★★★）.....109
7.3. 影响软件可靠性的 5 大因素（非重点☆☆☆☆☆）.....111
7.4. 程序复杂程度的定量度量 McCabe 方法（次重点★★★☆☆）.....112
7.5. 串并联系统（重点★★★★★）.....114
8. 项目管理（次重点★★★☆☆）.....115
8.1. 范围管理（次重点★★★☆☆）.....115
8.2. 进度管理（重点★★★★★）.....116
8.3. 成本管理（次重点★★★☆☆）.....118
8.4. 配置管理（次重点★★★☆☆）.....119
8.5. 风险管理（次重点★★★☆☆）.....119
9. 计算机组成原理（次重点★★★☆☆）.....120
9.1. 计算机系统层次结构（次重点★★★☆☆）.....120
9.2. 数据表示与编码（次重点★★★☆☆）.....123
9.3. 存储器系统（次重点★★★☆☆）.....126
9.4. 输入输出系统（次重点★★★☆☆）.....132

9.5. 指令系统（次重点★★★☆☆）.....133
9.6. 流水线技术（重点★★★★★）.....134
9.7. 阿姆达尔解决方案（次重点★★★☆☆）.....136
9.8. 系统性能评估（次重点★★★☆☆）.....136
10. 计算机网络（次重点★★★☆☆）.....137
10.1. 数据通信基础（次重点★★★☆☆）.....137
10.2. 网络体系结构与协议（次重点★★★☆☆）.....142
10.3. 常见的应用层协议（次重点★★★☆☆）.....144
10.4. 网络地址（次重点★★★☆☆）.....146
10.5. 子网和子网掩码（次重点★★★☆☆）.....147
10.6. 无分类域间路由（CIDR）.....149
10.7. IPv4 VS IPv6（次重点★★★☆☆）.....150
10.8. 局域网（次重点★★★☆☆）.....151
10.9. 无线局域网（次重点★★★☆☆）.....153
10.10. 网络互连与常用设备（重点★★★★★）.....154
10.11. 网络工程（次重点★★★☆☆）.....155
11. 操作系统（次重点★★★☆☆）.....156
11.1. 进程管理（次重点★★★☆☆）.....156
11.2. 内存管理（重点★★★★★）.....163
11.3. 文件系统（次重点★★★☆☆）.....166
12. 数据库系统（超级重点★★★★★）.....169
12.1. 数据库模式（重点★★★★★）.....169
12.2. 关系模型（重点★★★★★）.....171

12.3. 数据库设计与建模（超级重点★★★★★）.....177
12.4. 概念设计（超级重点★★★★★）.....177
12.5. 逻辑设计（超级重点★★★★★）.....180
12.6. 数据库的控制功能（重点★★★★★）.....192
12.7. 数据库性能优化（次重点★★★☆☆）.....196
12.8. 备份与恢复技术（次重点★★★☆☆）.....199
12.9. 数据仓库技术（次重点★★★☆☆）.....200
12.10. 分布式数据库（次重点★★★☆☆）.....201
12.11. 数据分片（次重点★★★☆☆）.....202
13. 信息安全（次重点★★★☆☆）.....203
13.1. 信息系统安全体系（次重点★★★☆☆）.....203
13.2. 数据加密技术（重点★★★★★）.....204
13.3. 认证技术（重点★★★★★）.....205
13.4. 密钥管理体系（次重点★★★☆☆）.....209
13.5. 通信与网络安全技术（次重点★★★☆☆）.....209
13.6. 系统访问控制技术（次重点★★★☆☆）.....210
14. 嵌入式系统分析与设计（次重点★★★☆☆）.....212
14.1. 嵌入式系统概述（重点★★★★★）.....212
14.2. 嵌入式数据库系统（重点★★★★★）.....214
14.3. 嵌入式操作系统（重点★★★★★）.....216
14.4. 多任务调度算法（次重点★★★☆☆）.....217
15. 法律法规（重点★★★★★）.....220
15.1. 保护对象（重点★★★★★）.....221

15.2. 保护期限（重点★★★★★）..... 221

15.3. 职务作品（重点★★★★★）..... 222

15.4. 侵权判定（重点★★★★★）..... 224

15.5. 专利法（补充重点★★★★★）..... 224

16. 后记..... 225

知识产权备案盗版封号

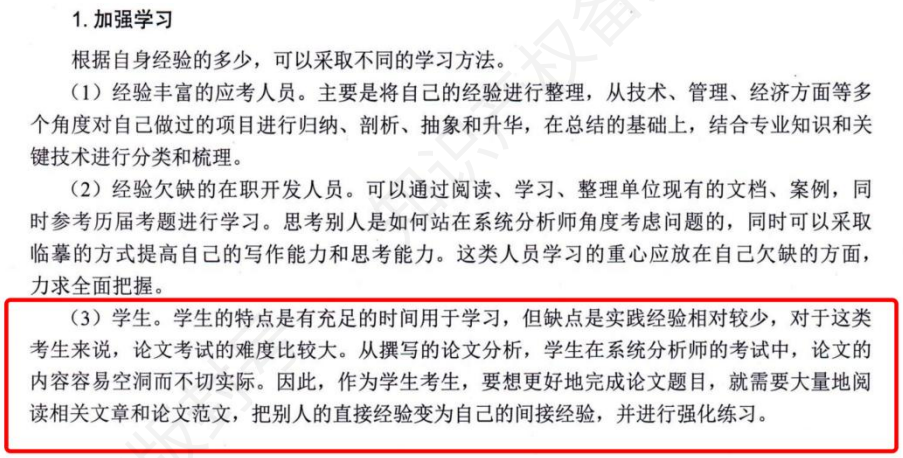
前言

版权提醒：本资料已通过国家版权局登记（登记号：渝作登字-2025-A-00624260），（一旦发现资料恶意流出，直接封号不退款，后果严重的话会请求平台披露你的个人信息，并于互联网法院起诉，请珍惜账号权益和个人名誉，谢谢）。

各位会员同学好，这是凯恩自己结合大量资料（官方教材讲义、历年真题、专业教材）和自己对于考试的理解，研究编制的系统架构设计师和系统分析师公共知识的内部讲义。有的同学知道，凯恩之前的资料架构和系分放在一起的，但是这样有很多不便。比如同学会问某个章节到底要不要看，会有很多迷惑的地方。所以干脆把架构和系分红宝书还是拆开来。

除了形式上的变化，另外在内容上，不单单是书本的内容了，还有专业教材和其他有效的互联网资料。因为软考的教材拼凑感严重，逻辑跳跃比较严重，有时候上下文都没办法联系起来，所以需要搜集大量其他专业教材来让它变得更加科学和系统。和之前的红宝书版本一样，为了方便大家快速突破重难点，凯恩对关键的内容进行了标注，其中下划线标出的内容非常有可能在选择题中考到，必须熟悉起来，其他内容形成表格帮助大家快速梳理相关的核心概念，用于短时间突破选择题。

再来说点题外话，软考最牛的一点在于以考代评和没有有效期限制。对于一个以考代评的考试，实际上是要体现传统评审的东西的，其中最主要的就是体现你的业绩。传统评审主要看你参与的项目论文（一般需要单位认可），造假成本比较高（这个不多说懂的都懂）。而以考代评的考试，目前考试的内容和考试形式没法做到对你业绩真实性进行认定。这意味着，即使你手头没有任何资源（没做过项目没做过架构师没做过系统分析师），也可以在论文里说自己从事架构设计师/系统分析师的岗位，编一些内容来糊弄阅卷人，只要让他相信你做过就行。在官方的教材中，其实默许了这个行为（如下图）。



1. 加强学习

根据自身经验的多少，可以采取不同的学习方法。

（1）经验丰富的应考人员。主要是将自己的经验进行整理，从技术、管理、经济方面等多个角度对自己做过的项目进行归纳、剖析、抽象和升华，在总结的基础上，结合专业知识和关键技术进行分类和梳理。

（2）经验欠缺的在职开发人员。可以通过阅读、学习、整理单位现有的文档、案例，同时参考历届考题进行学习。思考别人是如何站在系统分析师角度考虑问题的，同时可以采取临摹的方式提高自己的写作能力和思考能力。这类人员学习的重心应放在自己欠缺的方面，力求全面把握。

（3）学生。学生的特点是有充足的时间用于学习，但缺点是实践经验相对较少，对于这类考生来说，论文考试的难度比较大。从撰写的论文分析，学生在系统分析师的考试中，论文的内容容易空洞而不切实际。因此，作为学生考生，要想更好地完成论文题目，就需要大量地阅读相关文章和论文范文，把别人的直接经验变为自己的间接经验，并进行强化练习。

除非以后考试改革，取消论文写作，否则永远不会对你的论文真实性进行考察（成本太高也没必要）。这就相当于软考让获取职称的方式变简单了（以考代评不花钱不求人），这对普通人来说就是一个超级无敌巨大的利好。我们普通人不知道什么时候软考会改革，不知道以后以考代评会不会退化成考评结合。普通人要做的，能做的就是赶紧上岸赶紧拿证。体制内的同学祈祷早点聘用，体制外的同学早点领取人才补贴落袋为安。

使用说明

（1）初期筑基。先看这个红宝书一本全。红宝书总计 200 页，合计 11 万字。内容基本覆盖所有综合题（选择题）考试细节，配合我的公共课视频（综合知识视频）特别适合第一轮系统搭建知识框架。通过这份资料可以帮助考生吃透选择题考点。但是在二次复习的时候，有的同学觉得罗嗦，想要一个 lite 版本，纯背的那种。第一出两个版本维护成本很高，第二其实是没必要。等你把这里的资料翻烂了，你会看得一遍比一遍快，你心中就会诞生一个 lite 版本。

（2）重点导航。凯恩对各个知识点进行了标注，其中标注重点的，属于重要知识点。重点内容可以说不看必挂！标注超级重点的那就是在选择题和案例题都会考的，一处背了两个科目都可以用。次重点章节可看可不看，但是真题还是要做要会。

红宝书不要试图一次逐字看完，一次性记住。这大错特错！凯恩建议先配合我的 B 站视频（搜索芝士架构凯恩）快速扫完一遍。假如遇到非重点章节，真题刷到再回过头来看。这样重复 3-4 回

就可以上考场了。相信我，跟着我这样学，你的选择题大概率没问题。这里要记得是一定要配合刷题，否则无法学以致靠考，效果很差。

【关键提醒】

每学完 1 章必须完成对应真题训练（推荐「芝士架构」APP 每道题我都看过好几次确保解析没问题），否则知识留存率下降 60%。对红宝书标注或真题解析存疑的，建议通过微信私信凯恩提交具体题号+困惑点，可获得定制化解题路径分析。下面是一个基本的备考方案，建议大家根据自己的实际情况进行调整。

阶段名称	周期	核心任务	具体步骤	工具/方法	注意事项
阶段一：速通筑基	15 天	快速搭建知识框架	每日观看凯恩 B 站「芝士架构」视频（1.5 倍速） 精读红宝书★星标重点章节 非重点章节仅浏览小标题	B 站视频 公共知识红宝书	单日学习量≤10 页 非重点章节只需建立印象，无需深读
阶段二：真题淬炼	15 天	真题实战与错题沉淀	完成「做真题→查红宝书→建错题本」闭环 标注真题中冷门考点并补充到红宝书 通读次重点章节	历年真题 红宝书 错题本	冷门考点用荧光笔标注 次重点章节通读后理解度需达 70%
阶段三：循环迭代	3-4 轮（15 天）	多维度强化记忆	每轮侧重不同维度	错题本（APP） 红宝书	首轮：构建知识体系框架 次轮：聚焦真题高频考点 末轮：集中攻克错题集内容
阶段四：冲刺提效	考前 7 天	考点浓缩与实战模拟	制作思维导图（含★重点） 分析解题逻辑与红宝书关联	XMind 工具	模考严格计时（上午/下午各 1 次） 错题需标注对应红宝书页码及考点逻辑

常见问题

Q1：有了红宝书还用看书吗？

实际上红宝书的内容是远大于书本知识体系的，所以红宝书就够了。但是红宝书的内容仍然不能覆盖所有的考点。因为不管是系统架构设计师还是系统分析师，实际考试的内容是一个非常庞大的知识体系，考察的内容非常广泛，我们只能抓重点，抓主干进行学习。假如你想胡子眉毛一把抓，就必须要把书从头看到尾，从头背到尾。这种方式时间成本太高，对于大部分打工人来说不现实，另外实际考题中，书本上的内容实际上就重点考察若干章，如此大费周章，投产比 ROI 太低。相信我，只要红宝书掌握，真题掌握，加上你的积累（后端开发经验/项目参与经验），一次合格的概率极大。

Q2：红宝书背出就稳了吗？

不一定。前面说了，目前特别是案例部分，变数很大。从真题角度来看，它不喜欢考常规，不喜欢考书本内容。因此这个方面的知识红宝书会尽可能猜测，尽可能补充，但是不能说 100%覆盖。所以你平时需要有一定的后端技术积累（架构考生）或者一定的信息化项目参与经验（系分），否则无法应付。从 24 年 11 月开始，案例的阅卷故意压分非常明显。这就意味着可能你觉得自己答得还行，和别人一对答案，百度一搜索，觉得会拿一部分分。这是因为实际阅卷过程中要控制通过率，所以会临时提高阅卷标准（这个已经私下和阅卷人核实过了）。所以你要在考场上动用你所有能力，把想到的都写上去，把答案丰富起来，才能取得好成绩。

Q3：红宝书和其他辅导资料相比，优势在哪里？

红宝书通常是经过凯恩精心整理编写迭代了 N 多个版本（目前已经 6 个大版本），对考试大纲的覆盖度较高，知识点讲解较为系统全面。凯恩通过将复杂的知识体系进行梳理，以更清晰易懂的方式呈现给考生，帮助考生构建完整的知识框架。

Q4：学习红宝书的时候，有没有什么高效的记忆方法？

可以采用分模块记忆的方式，将红宝书的内容按照知识板块划分，逐个击破。特别是标注重点的章节，一定要结合笔记、思维导图来强化理解和记忆，把重点、难点以及自己的理解感悟记录下来，通过思维导图梳理知识点之间的逻辑关系，背诵记忆口诀提高记忆效率。另外，你可以利用艾

宾浩斯遗忘曲线，合理安排复习时间。

Q5：除了红宝书和真题，还有哪些学习资料对备考有帮助？

相关专业的经典教材是很好的补充资料，例如关于软件工程、计算机网络等基础学科的权威教材，能帮助你深入理解底层知识。行业内的技术博客、论坛也是不错的信息来源。这个适合有时间有余力的同学学习。并且只是补充，而不是主要的复习资料来源。

Q6：网上的一些免费学习资料可以用吗？

网上的免费学习资料可以作为辅助，但要注意筛选。有些免费资料，特别是论文范文这种曝光度很高的，你可以学底层逻辑，但是不要去抄。那么红宝书范文也不能抄？不能。也是同样的道理。你只能去用 AI 助手生成。

Q7：红宝书和 APP 为什么不参照书本大纲进行编排？

真实的考试中不同的内容侧重点和深度都不同，假如按照大纲来你会发现有的知识点无法归类。所以红宝书是根据考试知识点来进行分类整理的，这样可以过滤不少低频的无用的信息。

Q8：红宝书知识点没有联系，看不懂怎么办？

很遗憾，软考考试内容多，假如放在大学期末考那至少整个 4 门专业课。所以知识点很多，红宝书做不到也不打算把知识点写的很详细。因为知识点太多了，展开写的话 100 万字都打不住。所以也不要指望单看红宝书就能掌握全部的选择，全部的案例，还是要配合 B 站芝士架构凯恩的视频+刷题去理解。

Q9：我看了红宝书看了视频，选择题还是不会做，为什么？

不会做正常，会做不正常。因为软考选择范围考察极大，全部覆盖我估计要 100 万字内容。这些内容你要再短时间内全部记住显然不现实。所以芝士架构推出的红宝书+视频解决的是关键的知识、高频的内容（没错红宝书 10 万字，案例冲刺 7 万字只能解决高频考点，大概占比 40%-50%），剩下的内容你只能通过自己的积累了。

这里又引出一个问题，为什么选择题那么变态，为什么要那么多超纲的。换位思考，假如你是

出题人，软考办给你的任务是每次考试只能让 5%-10%的人通过，你会怎么做。**会不会给考生全部的考点，重点，会不会就一个题库里这些老题反复考？显然不会**。当然你会说，可以通过主观题控分呀。但是主观题控分多麻烦，软考办每次都要和各个阅卷中心沟通协调，还不如提高选择题的难度，先卡掉一部分人再说。

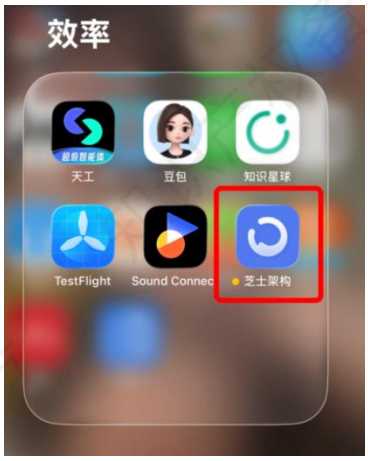
但是话说回来，这种抽象的出题模式对有工作经验的，基础比较好的同学非常友好。你只要掌握核心内容，剩下的就拼基础，不用卷死卷活。看看隔壁高项，案例、论文题目都是有知识点范围限制的，最后就在主观阅卷部分强行卡你，压你分，通过率奇低无比。

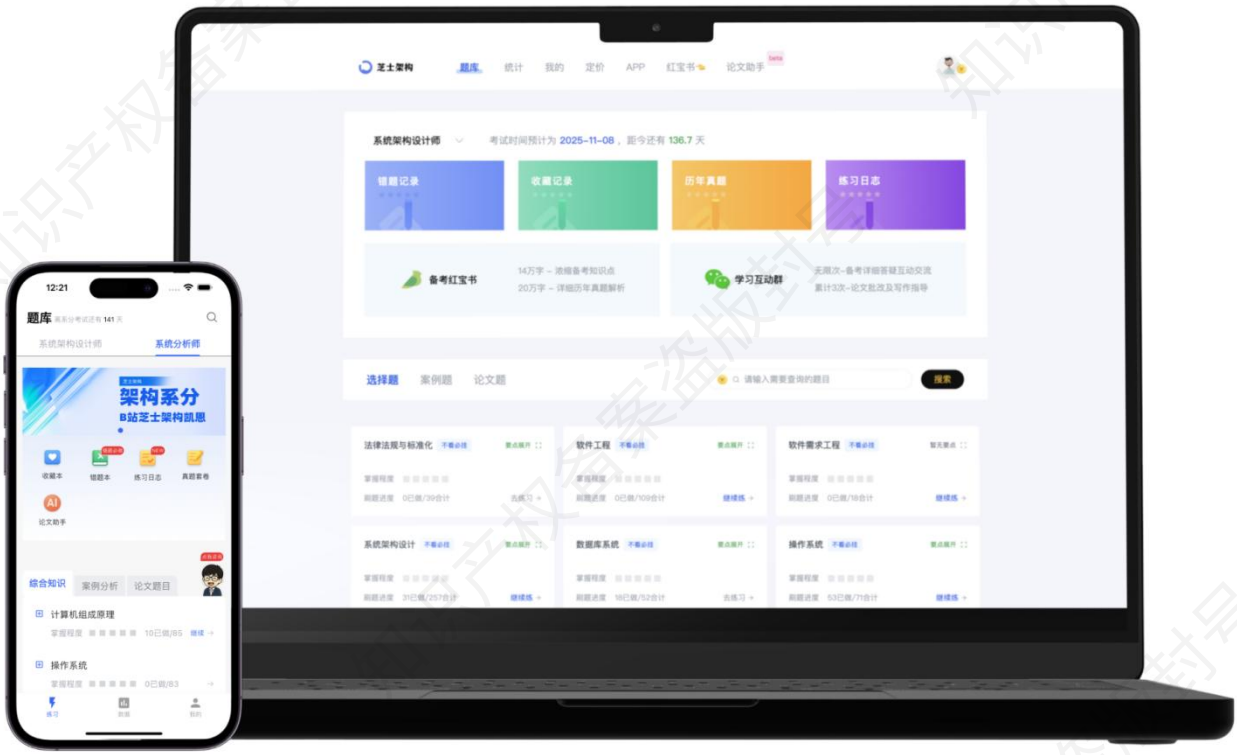
在线刷题

想要检测你的学习效果推荐你去使用我们的 PC 刷题网站 www.cheko.cc。看到这个资料的同学可以用上面的错题筛选，每日一练，搜索，真题导出，论文助手等功能。



或者 APP 刷题应用市场搜索芝士架构（蓝白图标的就是），和 Web 版本是同步的。





红宝书 6.0 章节安排思路

软件工程概述这一章节实际上对应了架构设计师教材第二版上第五章软件工程基础知识中部分章节。实际上软件工程不单单是包括开发方法，软件生命周期，还要包括需求分析、系统设计、架构设计、软件测试、部署运维、项目管理等多方面内容。但是，由于软考高级对于这些知识的考察都比较细致，远超书本所提及的内容，假如都放在一起，那么这一章节过于臃肿，层级会很深，

不方便阅读。所以在这里，我把软件需求工程、系统设计、系统架构设计、项目管理、软件测试、系统运行与维护，按照软件生命周期的顺序，拆解成独立的章节，进一步完善和补充考试要求的相关知识，方便你更好的阅读和学习。软件工程这些内容没有计算，纯概念题，基本只在选择或者论文出现，也是属于不看必挂的内容。

1.软件工程概述（重点★★★★★）

软件工程概述主要在选择题、论文题中都有考察。这一章非常枯燥，都是软件理论，直接听我视频课，过一遍就行。建议通过我的思维导图去回忆，配合真题加深记忆。对于综合题来说，本章所有下划线标出的内容都要熟悉，包括生命周期，开发方法，开发模型，开发环境和工具，软件过程管理等，这些概念需要配合刷题进行记忆。在架构考试中，软工考察得比较少，可以不关注。论文部分只关注一个是敏捷，一个是 RUP，另一个是测试开发，具体以论文押题为准。

1.1.软件工程相关定义（次重点★★★☆☆）

软件工程定义从不同的角度出发就会有不同的概念，所以这个一点也不重要。凯恩建议你只要记住这里的 PDCA 循环即可（因为选择题考过一次）。

软件工程由方法、工具和过程三个部分组成。包括以下 4 个方面 PDCA。（1）P（Plan）——软件规格说明。规定软件的功能及其运行时的限制。（2）D（Do）——软件开发。开发出满足规格说明的软件。（3）C（Check）——软件确认。确认开发的软件能够满足用户的需求。（4）A（Action）——软件演进。软件在运行过程中不断改进以满足客户新的需求。

1.2.软件开发方法（重点★★★★★）

实际考试不会来问你怎么划分软件开发方法。但是这里的内容，实际可以看作是后面章节内容的目录，故凯恩这里仍然保留。

分类方法	具体方法	描述
开发风格	自顶向下方法	系统的整体结构首先被定义和设计，然后逐步细化为更具体的模块和功能
	自底向上方法	是一种从具体模块开始逐步构建整个系统的方法
性质	形式化方法	形式化方法是一种具有坚实数学基础的方法
	非形式化方法	非形式化方法则不把严格性作为其主要着眼点
适应范围	整体性方法	适用于软件开发全过程的方法称为整体性方法
	局部性方法	适用于开发过程某个具体阶段的软件方法称为局部性方法

1.2.1.形式化方法（非重点☆☆☆☆☆）

形式化方法是指在软件开发过程中，使用严格的数学模型和形式化语言来描述系统的需求、设计和行为，并通过逻辑推理或模型检测等形式化验证手段，证明软件在所有可能的运行情况下都满足既定规范，从而获得可证明的软件正确性。它关注的不仅是测试能否发现错误，而是从理论上保证程序在功能、行为、安全性和活性等方面“必然正确”，常用于安全攸关或高可靠性系统的软件开发与验证。

这里需要掌握形式化方法的特点。（1）正确性验证难且耗时，数学模型和证明困难。（2）仍然需要和传统测试相结合。形式化方法提供理论保证，而测试则负责实际运行和发现运行时错误，不能说有了形式化方法，软件开发就不需要测试了。

1.2.2.净室软件工程（次重点★★★☆☆）

净室软件工程（Cleanroom Software Engineering， CSE）首先是一种上面提到的“形式化方法”，这一点非常关键。它不是测试方法、不是管理模型、也不是单纯的过程模型，而是强调用数学与统计理论，在设计和规约阶段消除错误的软件工程方法。它的理论基础是函数理论和抽样理论，如下表所示。

理论基础	展开描述
函数理论	函数理论净室软件工程的首要理论基础是函数理论，其核心主张将程序本质定义为从输入集合（定义域）到输出集合（值域）的严格映射关系，因此程序规约与函数规约具有完全等价性。围绕这一理论，相关考核常聚焦三大核心性质：一是完备性，二是一致性，三是正确性（辅助记忆：玩一阵）。
抽样理论	净室软件工程的第二大理论基础是抽样理论，其核心逻辑源于软件测试的固有局限：由于软件输入空间的无限性，穷尽所有可能场景的测试在实践中无法实现。基于此，净室方法引入统计学思想，将所有潜在使用场景视为整体样本空间，通过科学抽样选取代表性测试用例，进而推断整个系统的可靠性水平。

既然是软件开发过程，那么他也有开发步骤和测试步骤，主要关注其中四个，了解即可。统计过程控制下的增量式开发、基于函数的规约 + 盒子结构方法、正确性验证、统计测试与软件认证。

这里需要掌握的是，净室并非软件研发的“银弹”，其应用受技术门槛、实践场景等多重因素限制。（1）其正确性验证依赖严格的理论推导与数学基础，对工程师专业能力要求极高且验证成本显著。（2）净室不依赖传统单元测试（但还是需要传统的模块测试），但其工程实践中仍需考虑编译器、操作系统等底层环境缺陷。（3）净室本质仍属于传统软件工程体系，并未突破该体系的固有边界，无法解决所有工程问题。

1.2.3.逆向工程（次重点★★★☆☆）

次重点章节，出过 3-4 次选择。这里要区分逆向工程的几个概念。特别是再工程和逆向工程的区别一定要掌握。选择题考了多次。

概念	说人话	定义
逆向工程	从低层（代码）推回高层设计	分析程序，在比源代码更高抽象层次上建立程序表示，是设计恢复过程（注意和再工程的区分）
重构	同一抽象层里改结构，不升降层次	在同一抽象级别上转换系统描述形式
设计恢复	借助工具抽取设计信息	借助工具从已有程序中抽象出数据设计、总体结构设计和过程设计等信息

再工程（必背）	逆向工程 + 新需求 + 正向工程	在逆向工程信息基础上,修改或重构已有系统产生新版本，包括逆向工程、新需求考虑和正向工程三步
正向工程	从设计到代码	从现有系统恢复设计信息，用此信息改变或重构系统，改善整体质量(这个书上是放在逆向工程的语境下来说的，实际上我们都知道正向工程都是从零开始，这里主要起强调的是利用逆向工程的信息,解答逆向工程的作用）

这里要注意逆向工程和再工程的区别。逆向工程是从代码、程序这些低层内容出发，反推出系统的设计、结构、流程和文档，重点是分析和恢复设计，不一定修改系统。而再工程是看懂旧系统以后，再改造升级。所以考试看到“分析程序、恢复设计”，选逆向工程；看到“修改、重构、升级、产生新版本”，选再工程。

不同的逆向工程抽象层次从低到高如下表所示，要记忆。

实现级	包括程序的抽象语法树、符号表等信息，最低级的抽象
结构级	包括反映程序分量相互依赖关系信息，如调用图、结构图等
功能级	包括反映程序段功能及程序段之间关系的信息
领域级	包括反映程序分量或实体与应用领域概念对应关系的信息，最高级的抽象

这里每个层次反映出来的到底是哪些信息必须要利用下面的口诀加以记忆，选择题必考。

记忆口诀：世仇结分工龄（世仇结婚还要计算工龄）。世（实现）仇（抽象信息）结（结构）分（程序分量）工（功能级）龄（领域级）。功能级和领域级包含的信息就是功能和领域所以不单独展开。

1.2.4.基于构件的软件工程 CBSE（超级重点★★★★★）

超级重点章节，这一章不看不背就和那些裸考的一样，没区别。重要之处在于选择、论文反复考，不会不行。构件这块 23 年架构出了至少 5 分选择题。需要重点关注。

进入 1990 年代，随着分布式对象技术的发展，业界对独立部署、组装软件单元的需求愈发强烈。另一位 UML 创建者 Ivar Jacobson 在 1993 年将构件描述为“已经实现、用于增强编程语言

构造的单元”也体现了早期对构件的理解。直到组件化软件工程（CBSE）兴起，软件构件的概念才得到系统深入的定义。基于构件的软件工程（Component-Based Software Engineering，CBSE）是一种新型的软件开发模式,旨在通过复用可重用的软件构件来构造高质量、高效率的应用软件系统。这种模式强调将软件系统分解为独立的构件,每个构件都具有明确定义的功能和接口,可以独立开发、测试和部署。

1.2.4.1.构件的定义（重点★★★★★）

书上没有提到，实际上，这是伊恩•萨默维尔提出的构件的定义，选择喜欢考，所以需要尊重一下。这里唯一要补充的是可见状态，参见下表的解释。

构件特性	描述
可组装性	对于可组装的构件，所有外部交互必须通过公开定义的接口进行。另外，它还必须提供对自身信息的外部访问，例如它的方法和属性
可部署性	为使之可部署，构件需要是自包含的，它必须能作为一个独立实体在提供其构件模型实现的构件平台上运行。因而意味着构件总是二进制形式的且无须在部署前编译。如果一个构件实现为一项服务，它不必由用户来部署，而是由服务的提供者来部署
文档化	构件必须是完全文档化的，这样所有用户能确定是否构件满足他们的需要。应该定义所有构件接口的语法甚至语义
独立性	构件应该是独立的，它应该可以在无其他特殊构件的情况下进行组装和部署。如果在某些情况下构件需要外部提供的服务，应该在 “请求” 接口描述中显式地声明
标准化	构件标准化意味着在 CBSE 过程中使用的构件必须符合某种标准化的构件模型。此模型会定义构件接口、构件元数据、文档管理、组成以及部署
可见性	没有（外部的）可见状态（这里包含两个特殊情况，1.构件本身不直接暴露外部可见状态，但可以通过运行时容器（如应用服务器、框架等）管理状态并对外提供可见性，2.对于不影响构件功能的某些属性可以对外部可见）

记忆口诀：竹鼠挡毒标。竹（可组装性）鼠（可部署性）档（文档化）毒（独立性）镖（标准化）。

1.2.4.2.构件分类（次重点★★★☆☆）

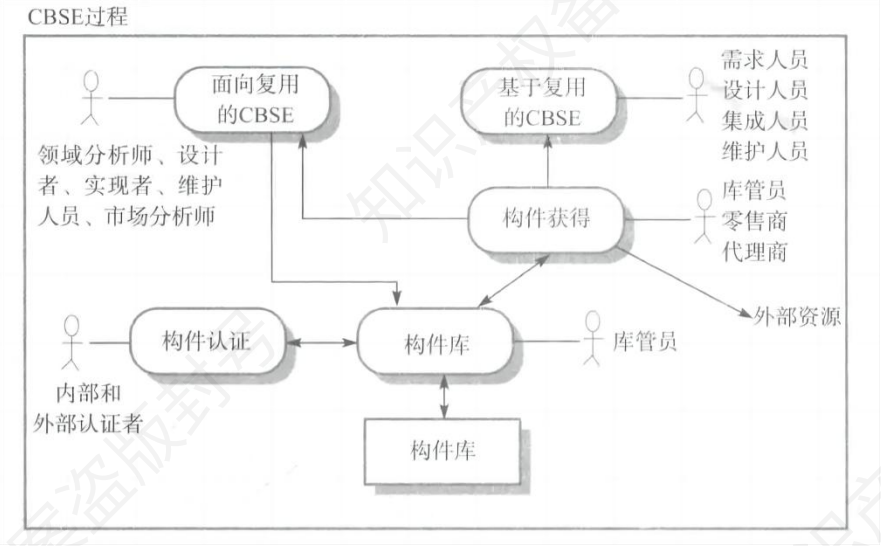
从构件的外部形态来看，构件可分为五类，有一年选择题考察过。

名称	内容	例子
独立而成熟的构件	成熟度高，已在实际环境多次检验；完全隐藏所有接口，用户只需通过规定命令使用。	数据库管理系统、操作系统
有限制的构件	提供接口并指出使用条件和前提；装配时可能产生资源冲突、覆盖等问题，需测试后使用。	面向对象程序设计语言的基础类库
适应性构件	通过包装或接口技术处理了不兼容性、资源冲突等问题，可直接在各种环境中使用。	ActiveX 组件
装配的构件	安装时已装配在操作系统、数据库管理系统或信息系统不同层次上，使用胶水代码即可连接。	多数软件商提供的成品软件产品
可修改的构件	可修改错误、增加新功能，通过重新 “包装” 或写接口实现构件版本替换。	应用系统开发中常用的功能模块

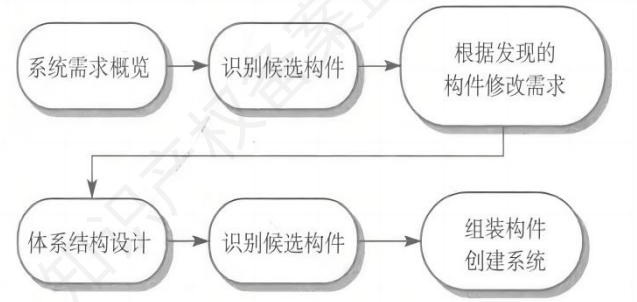
记忆口诀：独（独立）显（限制）时（适应）装（装配）秀（修改）。独立显卡挂在衣服上走时装秀。

1.2.4.3.CBSE 过程的主要活动（次重点★★★☆☆）

实际上按照伊恩•萨默维尔的理论，CBSE 活动分为两块。一个面向复用的开发。这个过程是开发将被复用在其他应用程序中的构件或服务。它通常是对已存在的构件进行通用化处理。另一个是基于复用的开发。这个过程是复用已存在的构件和服务来开发新的应用程序。书本上其实说的是后者。下面是两块活动的图例。



基于复用的开发过程如下所示，最初用户需求概要灵活，需完整以识别复用构件；早期使用可用构件细化修改需求，若不满足让用户改需求要么自己改（让他权衡成本这个是比较违反直觉的）；系统体系结构设计后需进一步构件搜索与设计精化，可能要找替代方案并修改构件功能；构件集成组装来开发系统，包括与基础设施集成、开发适配器等。



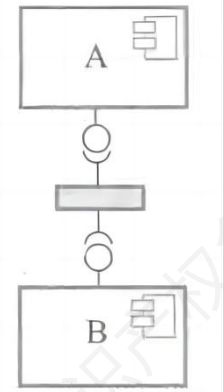
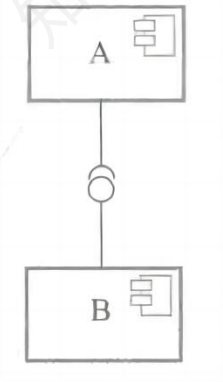
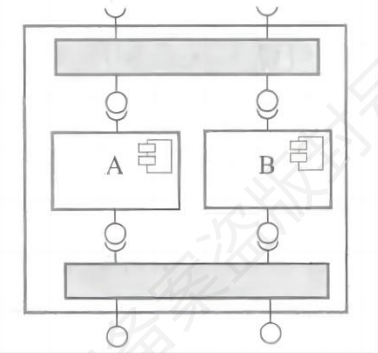
有的同学指出上图有误，指出下方的“识别候选构件”有误，应该是定制和适配，如下所示。（1）系统需求概览；（2）识别候选构件；（3）根据发现的构件修改需求；（4）体系结构设计；（5）构件定制与适配；（6）组装构件，创建系统。

实际上上图出自伊恩·萨默维尔的软件工程著作中的图例，在那本书的上下文里把下面的“识别候选构件”解释为构件定制与适配，但是写还是写成“识别候选构件”。

1.2.4.4.构件组装（次重点★★★☆☆）

组装方式选择题常考，所以还是要背，关键是这三种组装的描述，给你描述要知道属于什么，

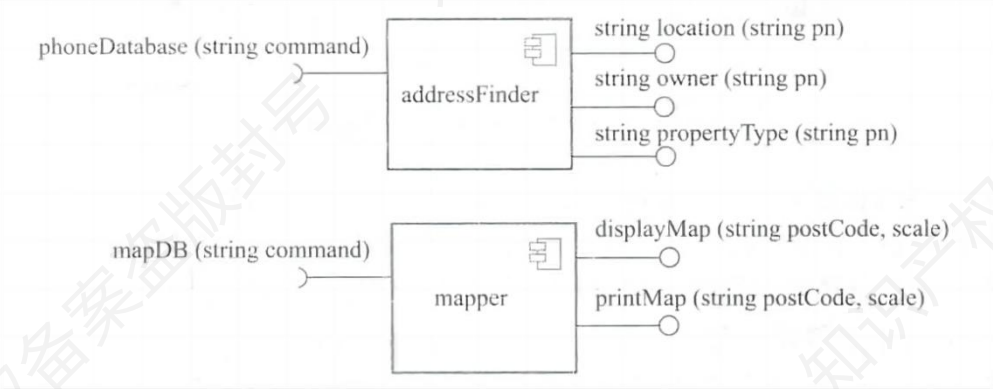
给你名称要知道具体的描述是什么。

组装类型	描述	特点	图示
顺序组装	通过按顺序调用已有构件创造新构件，看作“提供”接口的组装，先调用构件 A 服务，用 A 结果调用构件 B 服务	构件间不相互调用，可能被外部程序调用，需特定胶水代码确保接口兼容	
层次组装	一个构件直接调用另一个构件提供的服务，被调用构件“提供”接口与调用构件“请求”接口需兼容	接口匹配无需额外代码，不匹配需转换代码，构件作为网络服务时不能用	
叠加组装	两个或以上构件叠加创建新构件，新构件接口是原构件接口组合，通过外部接口分别调用原构件	A 和 B 不相互依赖和调用	

在组装构件时可能会遇到接口不兼容的问题，三种情况的判别要记忆。这里凯恩也给出了一个例子，方便你理解什么叫做不兼容。

不兼容类型	描述
参数不兼容	接口两侧操作名相同，但参数类型或数目不同

操作不兼容	“提供” 接口和 “请求” 接口操作名不同
操作不完备	一个构件的 “提供” 接口是另一个构件的 “请求” 接口的子集，或相反



上面是一个构件图，其中

提供接口：表示构件向外部提供的服务。在图形上，它用一个圆形来表示，连接在构件的边缘。从构件引出的线条连接到这个圆形，表示构件通过这个接口提供服务。这里的 location，owner 方法都是服务，括号里是他们的传入参数。

请求接口：代表构件在实现其功能时需要调用外部的服务。在图形上，它用一个半箭头来表示，同样连接在构件的边缘。箭头指向外部，表示构件需要从外部获取相应的服务。例如这里 phoneDatabase 就是 addressFinder 需要调用的外部服务。

这里说个题外话，实际上考虑构件的兼容与否的前提是要对构件做层次组装，换句话说，只有接口兼容才能做层次组装，或者说我们要做层次组装的前提是接口兼容。假如你要做的是顺序或者叠加实际上有中间代码参与可以做兼容和转换。以上图为例子，因为 addressFinder 提供的接口和 mapper 请求的接口不一样，即 addressFinder 的输出不能作为 mapper 的输入，属于操作不兼容，既然操作都不兼容参数兼容也谈不上，也就不能做层次调用，需要通过叠加组装或者顺序组装添加胶水代码来做转换。

1.3.软件过程/生命周期模型（重点★★★★★）

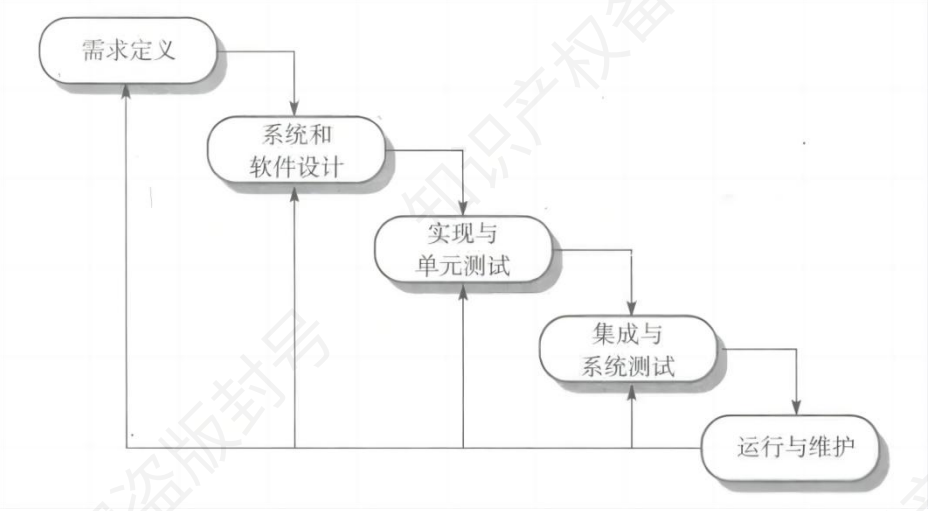
软件开发模型这一章是超级重点，选择爱考，有一年系分案例也着重考察了。经过凯恩的调研，得出的结论是软件过程 = 软件开发生命周期模型。它们是同义词。这点要知道，以免在不同的上下文给你理解带来歧义。

软件开发模型给出了软件开发活动各阶段之间的关系，它是软件开发过程的概括，是软件工程的重要内容。软件要经历从软件定义、软件开发、软件运行、软件维护（软件生命周期方法学），直至被淘汰这样的全过程，这个全过程称为软件的生命周期。软件生命周期描述了软件从生到死的全过程。为了使软件生命周期中的各项任务能够有序地按照规程进行，需要一定的工作模型对各项任务给予规程约束，这样的工作模型被称为软件过程（开发）模型，有时也称之为软件生命周期模型。下面的表格，凯恩按照不同的场景对场景的开发模型进行了分类。

前提条件	具体模型示例
软件需求完全确定	瀑布模型
软件开发初始阶段仅能提供基本需求	喷泉模型、螺旋模型、统一开发过程、敏捷方法等
以形式化开发方法为基础	变换模型

1.3.1.瀑布模型（重点★★★★★）

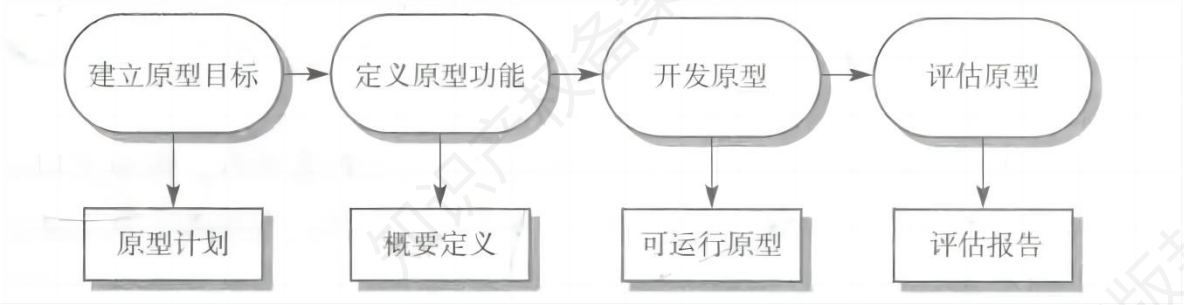
瀑布模型最早由 Winston W. Royce 在 1970 年的一篇论文中提出，它将软件开发的过程分为需求分析和定义、系统和软件设计、实现和单元测试、集成和系统测试和运行维护 6 个阶段，形如瀑布流水，最终得到软件产品，如下图所示。我视频里是另外一个版本的图片和下图不一样，不影响理解，不会考你原图，关键是要知道瀑布模型的缺点。



缺点：依赖于早期进行的需求调查，不能适应需求的变化；软件需求的完整性、正确性等很难确定，甚至是不可能和不现实的。

1.3.2.原型模型/演化模型/快速原型模型（重点★★★★★）

由于瀑布型的缺点，人们提出了原型模型。该模型如下图所示。



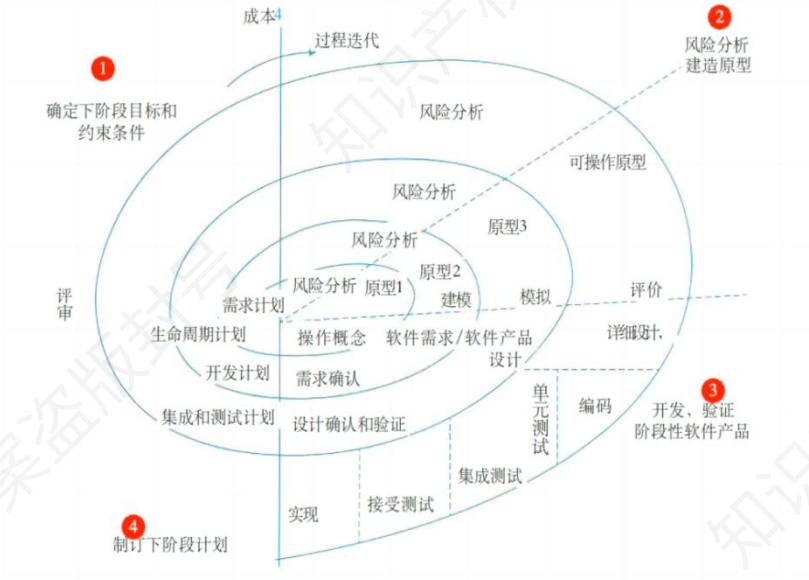
原型模型/演化模型/快速原型模型软考里一般不做区分，当做一个东西就行。它们都是适用于事先需求难完整定义的软件开发，基于快速开发的原型，依据用户反馈改进，不断迭代直至形成最终产品。

其优点是功能开发后可及时测试以验证需求，助于产生高质量要求；缺点是若让用户过早接触不稳定功能，可能对开发人员和用户造成负面影响。原型的特点是迭代，和下面增量模型有区分。

1.3.3.螺旋模型（重点★★★★★）

螺旋模型这里关键是要记忆，螺旋模型 = 快速原型 + 风险分析，选择题特别爱考。教材中

明确指出，螺旋模型是在快速原型的基础上扩展而成。

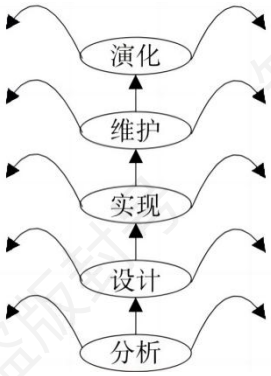


螺旋模型是基于快速原型，并加入风险分析的软件开发模型，使软件增量版本可快速开发，以一系列增量发布的形式进行。螺旋模型特别适用于庞大、复杂并具有高风险的系统，支持用户需求的动态变化，另外过多的迭代次数会增加开发成本，延迟提交时间。

1.3.4.喷泉模型（次重点★★★☆☆）

喷泉模型关键要记忆各个阶段可以交互进行。

喷泉模型是一种以用户需求为动力，以对象为驱动的模型，主要用于描述面向对象的软件开发过程。喷泉模型认为软件开发过程自下而上的各阶段是相互重叠和多次反复的，就像水喷上去又可以落下来，类似一个喷泉。各个开发阶段没有特定的次序要求，并且可以交互进行，可以在某个开发阶段中随时补充其他任何开发阶段中的遗漏。

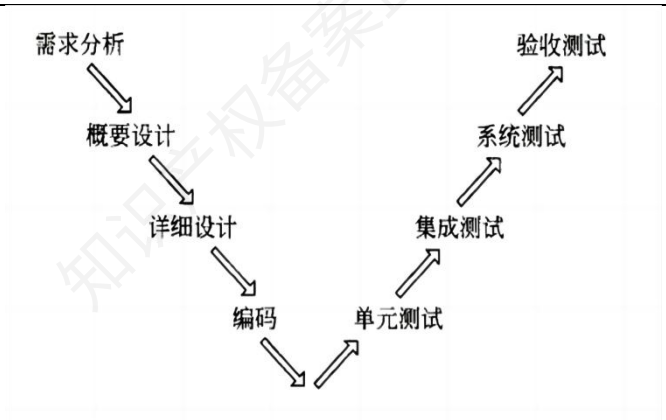


1.3.5.变换模型（次重点★★★☆☆）

变换模型是基于形式化规格说明语言和程序变换的软件开发模型,需要严格的数学理论和一整套开发环境的支持。

1.3.6.V 模型（重点★★★★★）

V 字模型常考什么设计是什么测试的依据,比如问你概要设计的产物是集成测试还是系统测试的依据，直接看下图你会认为概要设计和系统测试是对应的。好问题来了。系统测试的设计依据是什么？是概要设计？非也，实际上概要设计的产物实际上是集成测试的依据。集成测试人员可以根据概要接口定义来设计测试用例。有的同学会问，照你这么说，那么验收测试的依据是什么？那就是用户需求。



这里顺带提一下软考常考的三种测试类型，特别是测试依据需要清楚。

测试类型	测试对象	测试目的	依据（文档）
单元测试	可独立编译或汇编的程序模块、软件构件或 OO 软件中的类（统称为模块）	检查每个模块能否正确实现设计说明中的功能、性能、接口和其他设计约束等条件，发现模块内差错	软件详细（设计说明书）
集成测试	模块之间，以及模块和已集成的软件之间的接口关系	检查接口关系，并验证已集成的软件是否符合设计要求	软件概要设计（文档）

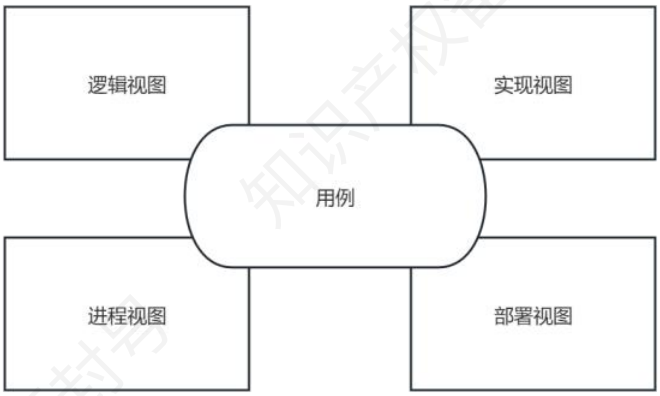
系统测试	完整的、集成的计算机系统	在真实系统工作环境下，验证完整的软件配置项能否和系统正确连接，并满足系统 / 子系统设计文档和软件开发合同规定的要求	需求分析（文档或开发合同）
验收测试	已完成全部开发工作、准备交付用户使用的软件系统	验收测试的目的就是从用户角度验证系统是否满足合同或需求规定的业务目标，确保系统可交付、可上线，并得到用户的正式认可	用户需求

1.3.7.统一过程（超级重点★★★★★）

RUP 在 2000 年代初很流行，但是 RUP 因为重流程、节奏慢、成本高，不适应快速变化的互联网环境，而敏捷以短迭代、轻文档、快速交付和灵活应变成为主流。你可以把 RUP 当成之前的陆地霸主恐龙，现在基本灭绝了而已。但是 RUP 不流行不代表没有意义（它的一些迭代增量的思想实际上对敏捷有很大的影响），不代表软考不考。实际上，软考常考 RUP 特点，生命周期。所以这块内容还是要背。

1.3.7.1.RUP **概述（重点★★★★★）**

RUP （Rational Unified Process）将项目管理、业务建模、分析与设计等统一起来，贯穿整个开发过程。RUP 是用例驱动的、以体系结构为中心的、迭代和增量的软件开发过程。RUP 中的开发活动是围绕体系结构展开的。软件体系结构的设计和代码设计无关，也不依赖于具体的程序设计语言。RUP 采用“4+1”视图模型来描述软件系统的体系结构。



这里的和后面的架构 4+1 视图不同注意区别。到这里我们只需要知道，软考里总共有 3 种 4+1 视图，后面会做统一的梳理。

1.3.7.2.生命周期（重点★★★★★）

RUP 软件开发生命周期是一个二维的软件开发模型， RUP 中有 9 个核心工作流，这 9 个核心工作流如下。23 年系分考试直接问你这九大核心工作流是什么，所以必须背。

RUP 涵盖 9 大核心关键词，包括业务建模、需求、分析与设计、实现、测试、部署、配置与变更管理、项目管理、环境。

记忆口诀为：建需分食测。建仓前，需先分好粮食、测好食物够不够。建（建模）需（需求）分（分析）食（实现）测（测试）。

部配变项环。部门配了新工具，遇情况得变步骤，项目每个环节都得盯牢。部（部署）配变（配置变更）项（项目管理）环（环境）。

1.3.8.敏捷方法（超级重点★★★★★）

敏捷常用常考，选择，案例，论文都会考察。选择考概念，案例可能考一个背景，不熟悉敏捷流程不好做，论文也会让你写敏捷论文，所以是超级重点。

敏捷方法相对于“非敏捷”而言，它们更强调开发团队与用户之间的紧密协作、面对面的沟通、频繁交付新的软件版本、紧凑而自我组织型的团队等，也更注重人的作用。敏捷方法是适应型，而

非可预测型。敏捷方法是以人为本，而非以过程为本。敏捷开发是迭代增量式的开发过程。

1.3.8.1.敏捷宣言（重点★★★★★）

敏捷宣言认为，个体和交互胜过过程和工具；可工作的软件胜过大量的文档；客户合作胜过合同谈判；响应变化胜过遵循计划。

1.3.8.2.适用场景（重点★★★★★）

（1）适合于规模较小的项目。（2） 敏捷方法适用于需求初步萌芽、尚未明确定型并且快速改变的情况，如果系统有比较高的关键性、可靠性、安全性方面的要求，则可能不完全适合。

记忆口诀：小变。小（规模小）变（快速改变）。

1.3.8.3.主要敏捷方法（重点★★★★★）

常见的敏捷方法有个概念就好，记住方法名很重要。

方法	简介	核心价值观 / 原则
极限编程（XP）	轻量级、灵巧且严谨周密的软件开发方法，强调通过短开发周期和频繁发布提高软件质量和开发团队生活质量	交流、朴素、反馈、勇气；强调团队合作、客户满意度和应对变化，遵循 YAGNI 和 DRY 原则等
水晶系列方法	提倡 “机动性的” 方法，包含具有共性的核心元素	以人为中心
Scrum	侧重于项目管理的迭代式增量软件开发过程	注重团队协作、响应变化、关注商业价值等
特征驱动开发方法（FDD）	迭代的开发模型，强调人、过程和技术三要素	未明确提及特别核心的价值观表述

1.3.8.4.Scrum（次重点★★★☆☆）

> Scrum 书本提及不多，但是属于高频考点，这里的内容需要熟悉。

Scrum 并不是一套详细的流程规范，而是一个用于指导团队协作、迭代交付的软件开发敏捷框架（framework）。它的核心思想是通过短周期、迭代式的工作方式，让团队围绕价值交付持续调整方向，减少不确定性与浪费，提高软件的交付频率和质量。Scrum 强调经验驱动（empirical process），即不断通过透明的信息、频繁的检查和及时的调整来改进过程与成果。

为了实现协作与过程改进，Scrum 框架定义了固定的角色结构。一个 Scrum 团队通常由三类核心角色组成：产品负责人（Product Owner）、敏捷教练（Scrum Master） 和开发团队（Developers/Team）。如下表所示。

角色名称	定位与核心关注点	核心职责
Product Owner	业务价值代言人；聚焦业务价值、需求优先级、stakeholder 对齐	1. 代表客户与业务利益相关者传递需求；2. 管理并优先级排序产品需求列表（Product Backlog）；3. 确保团队始终聚焦最大价值工作
Scrum Master	Scrum 方法推动者与障碍清除者；聚焦流程合规、交付效率、团队赋能	1. 推动 Scrum 框架落地，确保团队理解并遵循 Scrum 原则；2. 识别并清除团队工作障碍；3. 提升团队整体交付效能
Development Team	价值交付执行者；聚焦技术实现、团队协作、Sprint 目标达成	1. 以跨功能、自组织模式开展工作；2. 具体执行 Sprint 阶段任务，自主决策交付方式；3. 协同完成 Sprint 目标

Scrum 的核心活动围绕着 冲刺（Sprint） 展开。冲刺是一个有时间限制的迭代周期（通常 1–4 周），在每一个冲刺周期里，团队都要交付一个可工作的、潜在可发布的成果增量（Increment）。为了明确目标和保证高效协作，Scrum 规定了一系列固定事件，冲刺规划（Sprint Planning）、每日短会（Daily Scrum）、冲刺评审（Sprint Review）如下表所示。

活动名称	核心定位	关键目的 / 输出

Sprint Planning	明确冲刺范围与核心目标	1. 定义当前冲刺需完成的工作（梳理 Sprint Backlog）；2. 对齐团队冲刺目标；3. 制定具体执行计划
Daily Scrum	同步进展、暴露障碍	1. 团队每日快速同步 “已完成工作、今日计划、遇到的障碍”；2. 及时调整执行方向，确保冲刺目标推进
Sprint Review	展示成果、收集反馈	1. 向客户及利益相关者演示可交付成果；2. 收集需求调整、优化建议；3. 确认成果是否符合业务价值

为了支持透明和可视化的工作进展，Scrum 还定义了几个重要的工件（Artifacts）：产品待办列表（Product Backlog）、冲刺待办列表（Sprint Backlog）和增量成果（Increment），如下表所示。

工件名称	核心定义与特征	核心价值
Product Backlog	动态更新的功能需求清单，是产品所有待实现价值的集合；按业务价值优先级排序，随市场需求、业务反馈持续演进，内容可细化调整	提供产品价值全景，确保需求透明化、优先级明确
Sprint Backlog	源于产品待办列表的高优先级需求，是冲刺规划阶段筛选出的工作项集合；团队自主承诺并负责，明确本次冲刺的具体执行内容	聚焦冲刺目标，可视化当前迭代工作范围与进度
Increment	某次冲刺结束后完成的产品改进部分，满足“完成的定义（Definition of Done）”，具备独立可用价值，可直接交付或集成	沉淀可复用价值，实现产品持续迭代优化

1.3.9.其他模型-增量模型（次重点★★★☆☆）

这种更多的是一种思想。增量模型也叫增量开发，将系统分为多个模块，按照模块完成的顺序逐步实现系统功能，每个迭代周期都会新增一部分功能，最终达到完整系统的目的。增量模型的每一次增量版本都可以作为独立可操作的作品。而原型模型在实现功能之前采用原型设计进行迭代。

迭代和增量是项目开发中两种互补的方法：迭代通过分阶段循环（需求→设计→开发→测试）逐步完善同一功能模块，每次交付完整版本以快速验证需求并优化；增量则将项目拆分为独立子系统逐个开发交付，每次添加新功能模块以扩展系统架构。两者的核心区别在于迭代是功能深度优化

的螺旋上升，增量是模块叠加的线性扩展。增量和迭代都降低了实现需求变更的成本，更容易得到客户对于已完成开发工作的反馈意见。但是增量开发客户可以更早地使用软件并从中获得价值。

1.4.软件过程管理（次重点★★★☆☆）

软件过程管理一开始凯恩觉得放在这里比较奇怪的，后来看了一些权威教材发现，如今的工业界持续面临着提供更便宜、更好的软件的压力，同时，交付时间压力也越来越大。其结果是，许多软件企业都寻求通过软件过程改进来提升软件的质量、降低成本、加速他们的开发过程。

过程改进意味着理解当前的过程，并对其进行改变以提高产品质量，并且/或者降低成本和开发时间。这个其实是对软件开发过程从管理角度的细化补充。所以这个也是软件工程的重要内容。

1.4.1.软件能力成熟度模型 CMM（次重点★★★☆☆）

软件能力成熟度模型（Capability Maturity Model for Software，简称 CMM）是由美国卡内基梅隆大学软件工程研究所(SEI)于 1987 年提出的一种用于评估和改进软件开发过程的框架。CMM 的目的是帮助组织对软件过程进行管理和改进，增强开发与改进能力，从而能按时地、不超预算地开发出高质量的软件。

这里你需要熟记 CMM 的 5 个成熟度等级，初始级、可重复级、已定义级、已管理级和优化级，以及每个级别需要哪些管理的手段。

CMM 分为五个成熟级别	
初始级	过程没有明确规定，开发随意性强，成功依赖个别高手和运气
可重复级	已建立基本项目管理过程，能对成本、进度、功能进行跟踪，类似项目可重复以前成功经验
已定义级	软件过程已经文档化、标准化，形成组织级标准过程，各项目按标准过程执行
已管理级	对过程和产品质量有详细度量标准，能进行定量分析和控制

优化级	基于量化分析和新技术、新方法，不断持续改进过程

记忆口诀：初重定管优。除虫顶管用。除（初始）虫（可重复）顶（定义）管（管理）优（优化）。

1.4.2.能力成熟度模型集成 CMMI 量化等级（次重点★★★☆☆）

CMMI 是 CMM 的升级版本，强调系统工程和软件工程的整合，适合于信息系统集成企业。其理论基础源于与软件开发紧密相关的四大知识体系，分别是系统工程（SE）、软件工程（SW）、集成产品与流程开发（IPPD）及供应商采购（SS）。

记忆口诀：细（系统工程）软（软件工程）开（开发）工（供应商采购）。拿着细软开工了。

和 CMM 一样 CMMI 也分为五个成熟级别，成熟度等级，初始级、已管理级、已定义级、量化管理级和优化级，如下表所示。

CMMI 分为五个成熟级别	
初始级	过程通常是随意且混乱的。
已管理级	组织要确保策划、文档化、执行、监督和控制项目级的过程。
已定义级	能够根据自身情况定义企业和项目的标准流程，将这套管理体系与流程予以制度化。
量化管理级	组织建立了产品质量、服务质量以及过程性能的定量目标。
优化级	在这一级别，组织能够动态地调整其软件过程以适应不同的项目需求。

记忆口诀：储（初始级）罐（管理）定（定义）粮（量化）油（优化）。储罐定粮油。储罐里放粮食和油。

2.需求工程（超级重点★★★★★）

本章内容属于特别重点内容，选择案例论文都会考察。从选择角度来看，本章内容值得重点关注，特别是软件需求分析中的结构化分析数据流图，数据字典，面向对象分析中的用例模型，类图，顺序图，活动图等，以及需求管理中下划线标出的相关概念。从案例考察角度来看，此章节中结构化分析，面向对象分析中的内容都可能当作一小问进行考察，直接考察相关概念，比如数据流图平衡原则，数据流图基本规范等，面向对象分析中用例图的概念，面向对象分析活动过程，类的关系，活动图，顺序图，状态图，流程图之间的异同比较等。论文部分要看年份，以论文押题为准。

软件需求工程是包括创建和维护软件需求文档所必需的一切活动的过程。

在绝大多数软件项目中，失败往往并非源于技术能力不足，而是源于一个更早、也更隐蔽的问题——我们一开始并不真正清楚要做什么。需求模糊、理解分歧、目标摇摆，使得再先进的架构、再优秀的工程实践，都只能在错误的方向上不断加速。正是在这样的背景下，需求工程成为软件工程体系中最基础、却也最容易被低估的关键环节。

需求工程并不是简单地“收集需求”或“编写需求文档”。它关注的核心问题是：如何在不确定性极高的早期阶段，将多方模糊甚至相互冲突的期望，逐步转化为清晰、可验证、可交付的系统需求。需求工程解决的不是“系统如何实现”，而是“系统是否值得实现、应该实现什么、以及做到什么程度才算成功”。

需求工程贯穿软件生命周期的始终。它始于需求获取，通过访谈、调研和分析理解问题空间；在需求分析与规约阶段，将问题空间映射为系统解决方案的边界；在需求验证阶段，确保需求本身具备正确性与可行性；在需求管理阶段，应对变化、控制风险，维护需求与设计、实现和测试之间的可追溯关系。书上主要介绍了需求开发和需求管理两大过程。

需求开发包括需求获取、需求分析、编写需求规格说明书（需求定义）和需求验证 4 个阶段。其中需求获取（2.1 节），需求分析（2.2 节～2.5 节），需求定义（2.6 节）和需求验证（2.7 节）。

需求管理不太重要，我在 2.8 节简单介绍。

记忆口诀：获分归雁。收获瓜分的大雁。获（获取）分（分析）归（规格说明书）雁（验证）。

需求管理通常包括定义需求基线、处理需求变更和需求跟踪等方面的工作。

记忆口诀：基变更。这里只记名词。基因变更。基（基线）因变（变更）更（跟踪）。

2.1.需求获取（次重点★★★☆☆）

需求获取是一个确定和理解不同的项目干系人的需求和约束的过程。这个章节你需要熟悉有哪些方法。看熟悉就行，不需要一字一句背出来。

需求获取方法	具体内容	优缺点
用户访谈	有结构化和非结构化两种形式。组织方面包括准备（确定访谈目的、用户、问题并安排）、访谈过程（准时、找异常、深入调查、记录、注意措辞）、后续工作（吸收理解信息、复查结果、记录问题、发备忘录）	灵活性强；但存在困难，对系统架构设计师经验和沟通能力要求高
问卷调查	制作调查表需确定问题类型、编写问题、设计格式	优点：高效、低成本、匿名、便于整理统计；缺点：缺乏灵活性、信息不全面、不利于细节回答、返还率低
采样	从种群中选出有代表性样本集的过程，关键是确定样本大小，公式为样本大小 = α ×（可信度系数 / 可接受的错误）²，α 一般取值 0.25	可加快数据收集、提高效率、降低成本、减少偏差；样本规模确定依赖设计师主观因素，对其经验和能力要求高
情节串联板	通过一系列图片辅助讲故事叙述需求，类型有被动式、主动式和交互式，制作工具有静态和动态工具	生动、用户友好、交互性强，能对用户界面早期评审；花费时间多，降低需求获取速度
联合需求计划 JRP	通过客户或最终用户参与加速应用程序设计和开发的方法论，通常通过集会让开发者与顾客深入合作	对需求不清晰领域有用；会议组织和人员能力是难点
收集资料	查阅已有的文档、报告、学术论文、行业标准和其他相关文献获取需求信息	–
获取文档	向相关人员索取与项目相关的文档，如需求说明书、项目计划、流程图等	–
参加业务实践	参与实际业务流程和操作，体验和观察业务活动获取需求信息	–

记忆口诀：访问踩窗帘，合收十文。访问的时候踩到了窗帘，合计收你十文钱。访（访谈）问（问卷）踩（采样）窗（串联）帘，合（联合）收（收集）十（实践）文（文档）。

2.2.需求分析简介（非重点☆☆☆☆☆）

把杂乱无章的用户要求和期望转化为用户需求，这就是需求分析的工作。这一章了解即可，几乎不考。

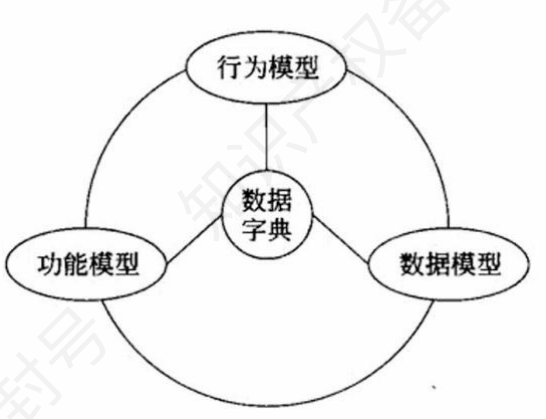
《软件需求》一书指出，需求分析的工作通常包括以下 7 个方面：包括绘制系统上下文范围关系图、创建用户界面原型、分析需求可行性、确定需求优先级、建立需求分析模型、创建数据字典和使用 QFD 等。在软件工程实践过程中，人们提出了许多种需求分析的方法，主要有结构化分析分析 SA 、面向对象分析 OOA 和面向问题域的分析（PDOA） 。其中结构化分析分析 SA 、面向对象分析 OOA 是软考的重点。

在有的书上，同学可能会看到面向对象建模、系统建模等说法。实际上软考教材上提到的分析方法就是建模。所谓的建模是指建立系统抽象模型的过程，其中每一个模型表示系统的一个不同的视角或观点。建模现在通常意味着使用某种图形化的表示法或者工具（比如 UML）表示系统。所以不管是结构化分析方法还是面向对象分析方法，我们都可以看作系统建模的不同方法，主要关注使用到了哪些图。

2.3.结构化分析方法/建模 SA（超级重点★★★★★）

超级重点章节，选择，案例和论文都有可能出。这里主要掌握数据流图的基本概念还有图例。

结构化分析方法的基本思想是自顶向下，逐层分解，把一个大问题分解成若干个小问题，每个小问题再分解成若干个更小的问题。在结构化分析方法中导出的分析模型如图所示。



结构化分析方法分析模型的核心是数据字典，围绕这个核心，有三个层次的模型，分别是数据模型（ER 图）、功能模型（DFD 图）和行为模型（状态转换图）。

2.3.1.数据流图 DFD（重点★★★★★）

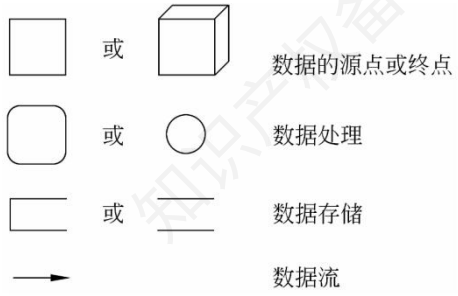
DFD 是结构化分析方法中的重要工具，是表达系统内数据的流动并通过数据流描述系统功能的一种方法。

2.3.1.1.DFD 的基本符号（重点★★★★★）

在 DFD 中，通常会出现 4 种基本符号，如下表所示（结合图片进行记忆）。这一块是看图的基础，必须掌握。

符号名称	表示内容	图形表示
数据流	具有名字和流向的数据	标有名字的箭头
加工（数据处理）	对数据流的变换	圆圈
数据存储	可访问的存储信息	直线段
外部实体（数据源及数据终点）	位于被建模的系统之外的信息生产者或消费者，表明数据处理过程的数据来源及数据去向	标有名字的方框

补充：数据存储要被外部实体使用必须要经过加工。有一年案例考察过这个知识点。

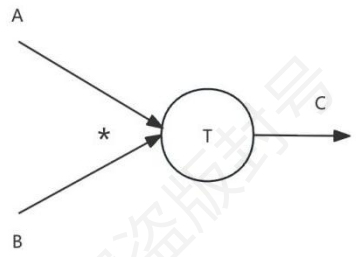
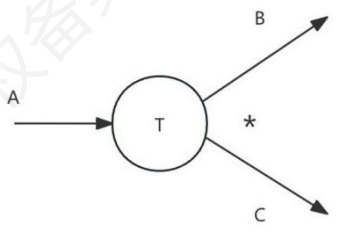


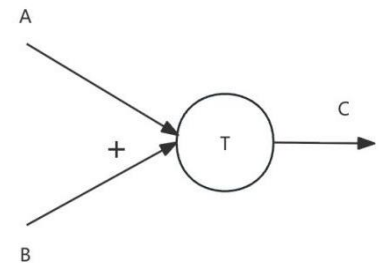
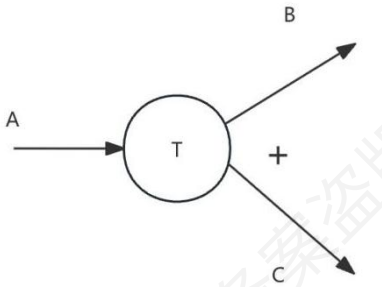
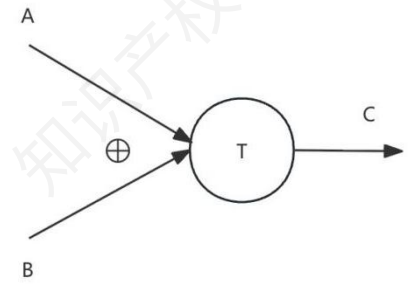
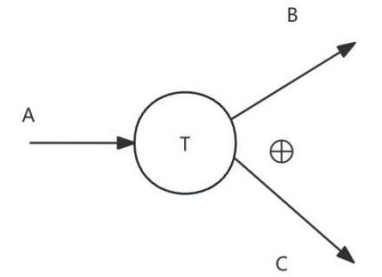
有的同学看到其他资料反馈图对不上，这里特此解释。数据流图模板有两种常见的记号集：Yourdon 和 Coad 方法以及 Gane 和 Sarson 方法。这两个系统都以创造它们的计算机科学家的名字命名。上图左侧的是 Gane 和 Sarson 方法，右侧是 Yourdon 和 Coad 方法。

扩充符号通常不属于 DFD 标准的 Gane & Sarson 或 Yourdon 标准，但在软考中考到了。

符号名称	表示内容	图形表示
与	表示数据流之间的 “与” 关系。输入流中，所有输入数据流都到达后才能进行加工处理；输出流中，加工结束会同时产生所有输出数据流。	*
或	表示数据流之间的 “或” 关系。输入流中，任何一个输入数据流到达就可以进行加工处理；输出流中，加工处理的结果至少会产生其中一个输出数据流。	+
异或	表示数据流之间的 “互斥” 关系。输入流中，只有当其中一个输入流到达时才能进行加工处理；输出流中，加工处理的结果仅产生这些输出数据流中的一个。	⊕

这些符号可以作用于输入和输出，如下图所示。

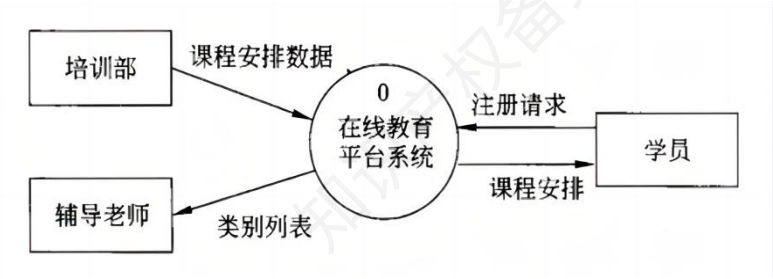
表示内容	图形表示
	数据 A 和 B 同时输入才能变换成数据 C
	数据 A 变换成 B 和 C

	数据 A 或 B，或 A 和 B 同时输入变换成 C
	数据 A 变换成 B 或 C，或 B 和 C
	只有数据 A 或只有数据 B（但不能 A、B 同时）输入时变换成 C
	数据 A 变换成 B 或 C，但不能变换成 B 和 C

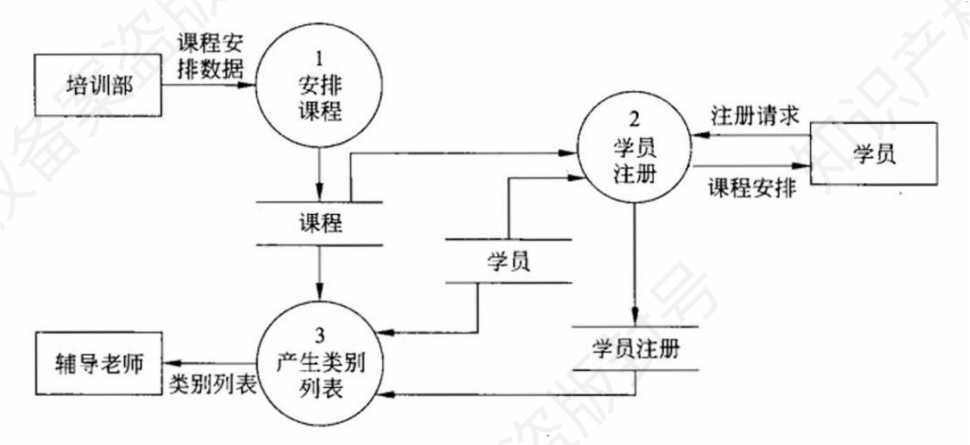
2.3.1.2.DFD 的层次（重点★★★★★）

结构化分析方法的思路是依赖于 DFD 进行自上而下的分析。DFD 可以表现系统的高层和低层概念，通过先绘制高层 DFD，再对其中的加工进行逐步分解，直至系统被清晰描述。这一节不需要背，你知道这个流程就行。

（1）顶层图。顶层图是描述系统最高层结构的 DFD，它的特点是将整个待开发的系统表示为一个加工，将所有的外部实体和进出系统的数据流都画在一张图中。



（2）逐层分解。当完成了顶层图的建模之后，就可以在此基础上进行进一步的分解。对上图进行分解，在对原有流程了解的基础上，可以得到下图。



2.3.1.3.数据流图的平衡原则（重点★★★★★）

重点关注，数据流图的平衡原则，如下所示，要掌握父子图的平衡和子图的平衡，结合案例真题能够找茬写出错误的理由。

（1）父图（上层数据流图）与子图（下层数据流图）平衡。个数一致：两层数据流图中的数据流个数一致。方向一致：两层数据流图中的数据流方向一致。

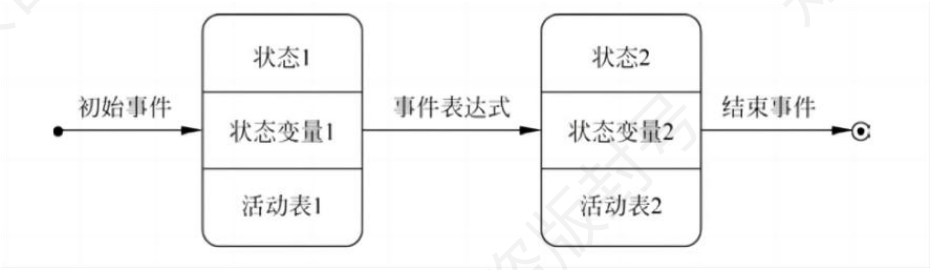
（2）子图内部的平衡表示如下。

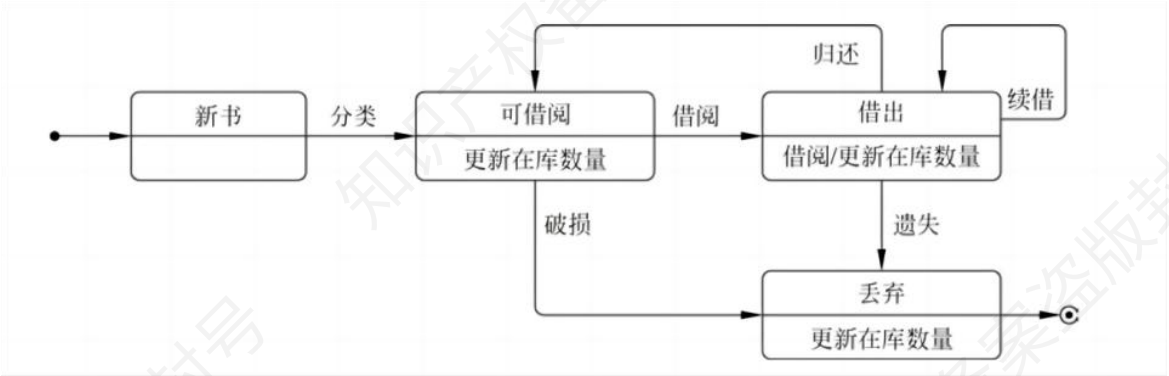
概念	描述
黑洞	加工只有输入没有输出
奇迹	加工只有输出没有输入
灰洞	加工中输入不足以产生输出（加工应该输出 A，实际给出 B，挂羊头卖狗肉的意思）
数据存储	正常情况下必须既有读的数据流，又有写的数据流；在某张子图中，可能只有读没有写，或者只有写没有读

2.3.2.状态转换图（次重点☆☆☆☆☆）

状态转换图是一种描述系统对内部或外部事件响应的行为模型。它描述系统状态、事件和事件引发系统在状态之间的转换。在状态转换图中定义的状态主要有：初态（即初始状态）、终态（即最终状态）和中间状态。初态用黑圆点表示，终态用黑圆点外加一个圆表示，状态图中的状态用圆角四边形表示，状态之间状态转换用带箭头的线表示。

状态转换图怎么画不重要不会考，因为这个东西 1970 年代发明的，没有统一标准，现在已经用 UML 状态图替代。





2.3.3.数据字典（非重点☆☆☆☆☆）

数据字典基于 DFD，对其中所有命名元素进行定义，赋予图形元素确切解释。DFD 与数据字典配合，能从图形和文字两方面完整描述系统逻辑模型。

2.4.面向对象分析方法/建模（超级重点★★★★★）

超级重点，选择题，案例都常考。案例部分会结合具体场景让你识别类，识别用例的关系等等。

2.4.1.分析/建模工具（重点★★★★★）

面向对象分析方法或者说面向对象建模方法实际上有的书分为对象模型（类图表示）、动态模型（状态图表示）、功能模型（用例图表示）。功能模型从用户和业务角度说明系统“要做什么”，描述系统对外提供的功能及其处理过程，常用数据流图或用例图表示；动态模型刻画系统或对象在时间维度上的行为变化，说明系统在什么状态下、受到何种事件触发会发生什么行为，通常通过状态图、顺序图等来描述；对象模型则从结构角度定义系统中有哪些对象、对象包含哪些属性以及对象之间的关系，是三种模型中最基础、最核心的模型。对象模型为功能模型和动态模型提供了结构基础，而功能模型和动态模型又从行为和时序角度对对象模型进行补充，三者相互映射、相互约束，共同构成对待开发系统的完整而统一的认识。这个某年考试考到过，需要重视。而教材呢就只提到了用例模型（用例图表示）和分析模型（用类图表示）。

2.4.2.用例模型（重点★★★★★）

按照书上的说法，构建用例模型一般需要经历 4 个阶段，分别是识别参与者、合并需求获得用例、细化用例描述和调整用例模型，其中前三个阶段是必需的。

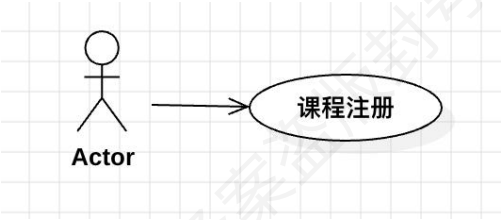
记忆口诀：人合细条。人合团结了就协调了。人（识别参与者）合（合并）细（细化）条（调整）。

2.4.2.1.用例图的元素（重点★★★★★）

这里我们重点要记忆的是这里的关联元素。其中关联关系一般是作用于参与者和用例之间的（横线符号），拓展、包含和继承是作用于用例和用例之间的。有一年真题提到通信关联也可以作用于用例之间，这个不太说得通。

结构元素	图形符号	关系元素	图形符号
用例(Use Case)	Use Case	关联(Association)	——
参与者(Actor)	Actor	扩展(Extend)	<<extend>>
系统边界(System Boundary)	System	包含(Include)	<<include>>
注释(Note)	Note	泛化(Generalization)	→
		注释链接(Note Link)	- - - - -

用例是一种描述系统需求的方法，使用例的方法来描述系统需求的过程就是用例建模。在用例图中，主要包括参与者、用例和通信关联（箭头）三种元素，如图所示。



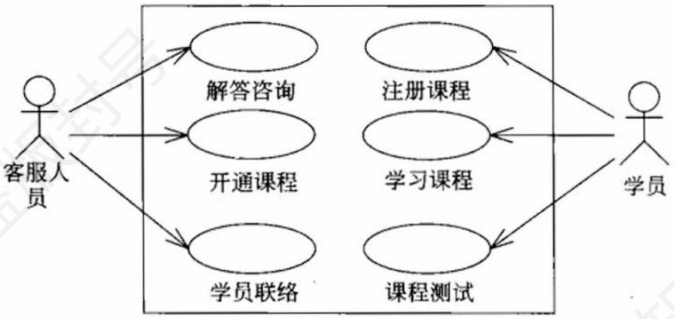
2.4.2.2.识别参与者（重点★★★★★）

参与者是与系统交互的所有事物,该角色不仅可以由人承担,还可以是其他系统和硬件设备(传感器)，甚至是系统时钟。案例真题让你分析题目中哪些属于参与者，这里要记住处理人，一些系统和设备都是。要注意的是，参与者一定在系统之外，不是系统的一部分。下面是三个具体的参与者识别的场景。另外有一年提到,参与者与参与者之间也可以发生继承关系,这个知识点了解即可。

参与者类型	描述	示例
其他系统	当系统需要与其他系统交互时，该其他系统为参与者	对某企业在线教育平台系统，该企业的 OA 系统是参与者
硬件设备	若系统需与硬件设备交互，此硬件设备为参与者	开发集成电路（IC）卡门禁系统时，IC 卡读写器是参与者
时钟	当系统需定时触发时，时钟作为参与者	开发在线测试系统 “定时交卷” 功能时，引入时钟作为参与者

2.4.2.3.合并需求获得用例（次重点★★★☆☆）

将识别到的参与者和合并生成的用例，通过用例图的形式整理出来，以获得用例模型的框架，如图所示。



2.4.2.4.细化用例描述（次重点★★★☆☆）

这一章节几乎没有考过，架构可以忽略，系分可以看看，熟悉一下用例描述包含哪些部分，案例可能会进行考察。

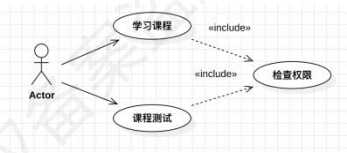
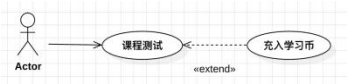
用例描述部分	内容说明
用例名称	应与用例图相符，并标注相应编号
简要说明	用简洁自然语言，描述用例为参与者传递的价值结果
事件流	参与者和系统为达成目标所发生的一系列活动
非功能需求	单列描述用例涉及的非功能需求，需保证可度量和可验证
前置条件和后置条件	前置条件为执行用例前系统必须存在的状态，不满足则用例无法启动；后置条件是用例执行完毕系统可能处于的一组状态
扩展点	若有扩展（或包含）用例，写出其名称及使用情况
优先级	表明用户对该用例的期望，确定开发先后顺序

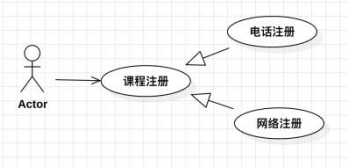
部分	内容说明
用例名称	UC01 用户登录系统
简要说明	用户通过输入账号和密码成功登录系统，获得系统授权的功能访问权限。
事件流	基本流程：

	1. 用户在登录页面输入账号和密码； 2. 系统校验账号和密码的有效性； 3. 系统返回校验结果； 4. 若信息正确，用户进入系统主界面。
备选流程：	a. 若用户输入信息有误，系统提示“用户名或密码错误”； b. 用户可重新输入账号和密码； c. 若连续输入错误 3 次，系统锁定账户，并提示联系管理员解锁。
非功能需求	登录响应时间 ⩽ 2 秒； 系统必须支持至少 1000 并发用户登录； 提示信息必须清晰、准确，符合可用性规范。
前置条件	用户已在系统中注册并存在有效账号。
后置条件	登录成功：用户处于系统主界面，获得相应操作权限； 登录失败：系统保持在登录页面，未进入系统。
扩展点	UC02：找回密码（若用户忘记密码，可选择该用例）。 UC03：多因素认证（系统可选扩展，若开启则在输入账号密码后要求输入动态验证码）。
优先级	高：系统核心功能，必须优先开发和验证。

2.4.2.5.调整用例模型（重点★★★★★）

这个章节是必须要掌握的，死记硬背也要背下来，包括箭头形状，箭头指向含义，谁指向谁，各个关系的内涵。用例之间的关系主要有包含、扩展和泛化，利用这些关系，把一些公共的信息抽取出来，以便于复用，使得用例模型更易于维护。

关系类型	描述	示例	构造型	箭头指向	图例
包含关系	从两个或多个用例提取公共行为，提取出的公共用例为抽象用例，原始用例为基本用例。	“学习课程”和“课程测试”都需检查学员权限，“检查权限”为抽象用例，“学习课程”和“课程测试”为基本用例。	<<include>>	指向抽象用例	
扩展关系	一个用例混合多种不同场景，可分为一个基本用例和多个扩展用例。	学员“课程测试”时，若测试次数超出限额需“充入学习币”，“课程测试”是基本用例，“充入学习币”是扩展用例。	<<extend>>	指向基本用例	

泛化关系	多个用例有类似结构和行为，将共性抽象为父用例，其他为子用例，子用例继承父用例所有结构、行为和关系。	学员课程注册可通过电话或网上注册，“注册课程” 是 “电话注册” 和 “网上注册” 的父用例。		指向父用例	

2.4.3.分析模型（类图）（超级重点★★★★★）

分析模型描述系统的基本逻辑结构，展示对象和类如何组成系统（静态模型），以及它们如何保持通信，实现系统行为（动态模型）。建立分析模型的过程大致包括定义概念类、确定类之间的关系、为类添加职责、建立交互图等，其中有学者将前三个步骤统称为 CRC（类——责任——协作者）建模。选择案例都会考。

2.4.3.1.为类添加职责（非重点☆☆☆☆☆）

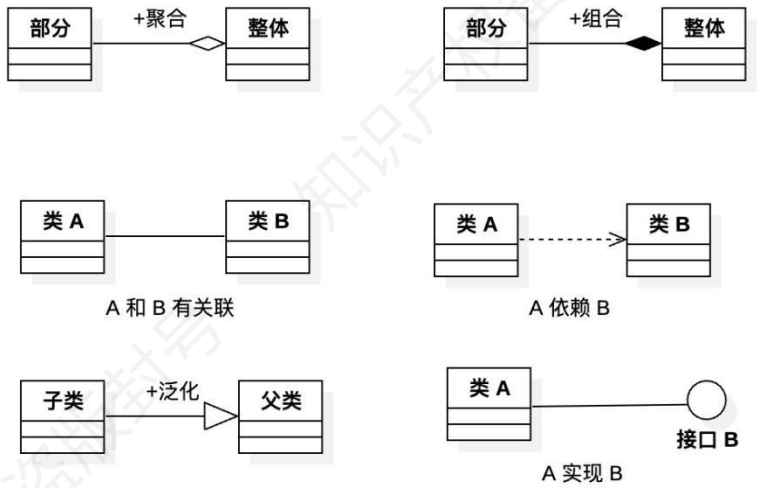
类的职责包括维护的知识（属性）和执行的行为（成员方法或责任）（不重要，白话就是给类添加属性和方法）。

2.4.3.2.建立交互图（非重点☆☆☆☆☆）

多个对象的行为通常采用对象交互来表示， UML 2.0 提供的交互图有顺序图、交互概览图、通信图和定时图。每种图出于不同视点对行为有不同的表现能力，其中最常用的是顺序图，几乎可以用在任何系统的场合。

2.4.3.3.类图（重点★★★★★）

类之间的主要关系有关联、依赖、泛化、聚合、组合和实现等六大关系，它们在 UML 中的表示方式如图所示。其中，聚合和组合关系都是关联关系的一种。这些关系的强弱顺序不同，排序结果为：泛化=实现>组合>聚合>关联>依赖。



记忆口诀：饭（泛）食（实现）组聚关羽（依）。干饭小组和关羽聚会。

类图实际上是 UML 里比较复杂的一个图，一般来说掌握这 6 个关系就已经非常不错。

关系类型	定义	举例	特点	记忆方法
关联关系	一种强语义联系的结构关系，表明两个事物间存在明确、稳定的语义联系	学生与课程，学生选修课程，两者存在明确关联	强语义、结构稳定	-
依赖关系	两个类 A 和 B，若 B 的变化可能引起 A 的变化，A 依赖于 B	教师授课使用教学工具，教师类不持有工具类实例，仅在方法中临时使用	B 变化影响 A，A 对 B 是临时使用关系	教师临时用工具授课，工具变化影响授课
泛化关系	描述一般事物与特殊种类的关系，即父类与子类的关系	经理是员工的子类，继承员工属性和方法并扩展特有功能	子类继承父类属性和方法，可扩展功能	父与子，子承父业
聚合关系	表示类之间整体与部分的关系，“部分” 可同时属于多个 “整体”，“部分” 与 “整体” 生命周期可不同	图书馆与书籍，一本书可被多个图书馆收藏，书的生命周期与图书馆不同	部分可多属，生命周期可不同	好聚好散，部分能单独存在
组合关系	同样表示类之间整体与部分的关系，“部分” 只能属于一个 “整体”，“部分” 与 “整体” 生命周期相同	汽车与发动机，发动机仅属于一辆汽车，发动机随汽车创建与消亡	部分唯一归属，生命周期相同	与聚合比，生命周期同步
实现关系	将说明和实践联系起来，接口说明行为，类实现接口操作	多个类实现 “打印” 接口，各自实现打印操作	接口定义行为，类负责实现	接口定行为，类来实现

2.5.建模语言（重点★★★★★）

2.5.1.统一建模语言 UML（重点★★★★★）

UML（统一建模语言）是一种用于描述、可视化和文档化软件系统的标准建模语言。从总体上来看，UML 的结构包括构造块、规则和公共机制三个部分。

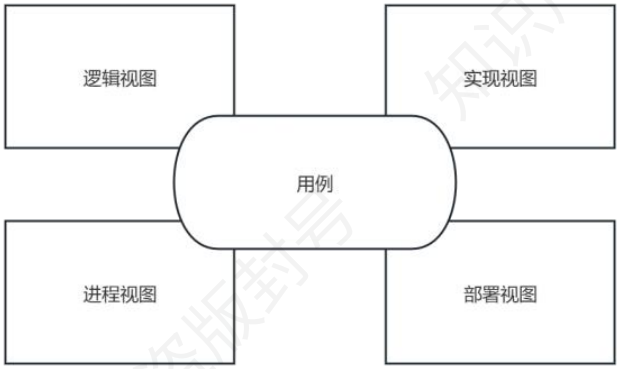
UML 中的公共机制是统一建模语言（UML）实现系统可视化建模与形式化描述的核心功能模块，其不仅限于图形语言标准，通过图形表示法和规格说明分别展现系统宏观结构与微观细节。

该机制包含四大要素：详述通过文字补充图形符号的语义信息；修饰利用附加符号（如访问权限标识）增强元素表现力；通用划分基于类/对象二分法抽象系统层次；扩展机制则通过构造型、标记值和约束支持建模元素的自定义扩展。四项机制共同保障了 UML 在标准化与灵活性之间的平衡。

在 UML 中，有两种类型的图，分别为结构图和行为图。结构图用来描述事物之间的关系包括类图、对象图、组件图和部署图。行为图用来描述参与者和用例之间的交互，或者描述参与者如何使用系统。行为图包括用例图、顺序图、活动图、状态图和通信图（实际上没啥意义，这些都是属于 UML 基本概念，但是选择题喜欢考察，这些内容要记忆）。

2.5.2.UML 4+1 视图（重点★★★★★）

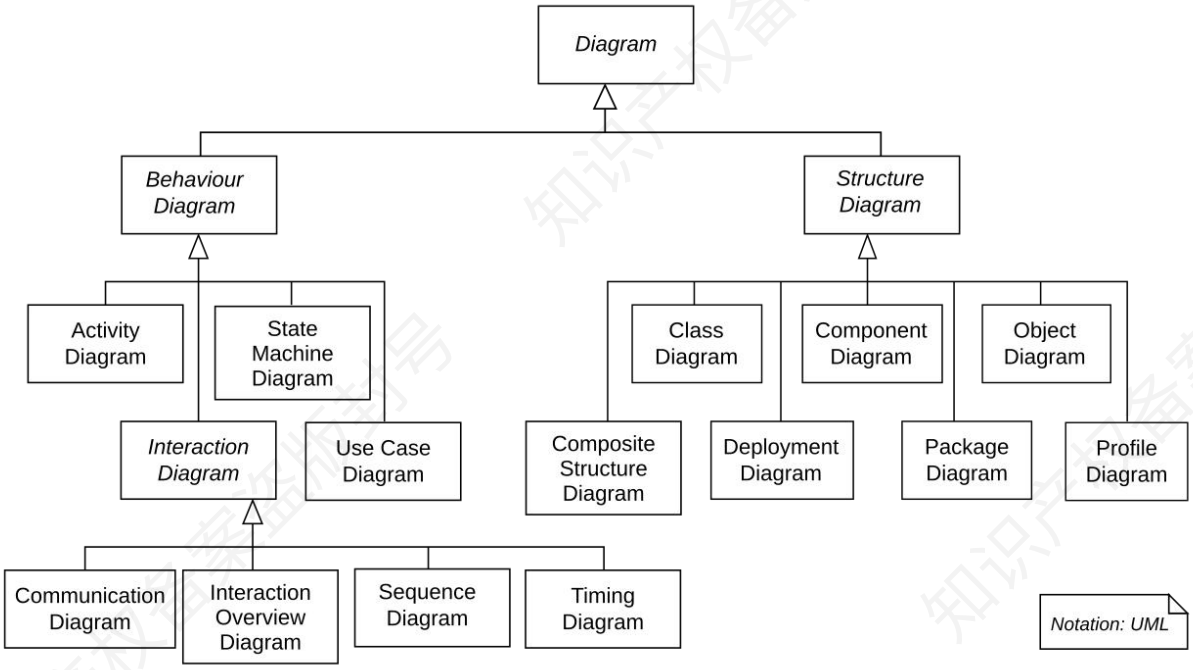
UML 4+1 视图 ＝RUP 4+1 视图。都是以用例为中心的。出自系统架构设计师第二版 182 页。



视图名称	主要描述	关注点	作用	关注的角色
逻辑视图	表示设计模型中在架构方面具有重要意义的部分，如类、子系统、包和用例实现的子集	系统的功能模块及其相互关系，以实现系统的业务目标	展示系统的主要功能组件及交互关系,为系统设计提供功能框架	最终用户
进程视图	可执行线程和进程作为活动类的建模，是逻辑视图的一次执行实例，描述并发与同步结构	系统运行时的进程和线程的组织、交互，以及并发和同步机制	体现系统的动态运行特性，有助于优化系统性能和可伸缩性	系统集成人员
实现视图	对组成基于系统的物理代码的文件和构件进行建模	软件构件在开发环境中的组织结构，包括代码的模块化、层次结构、软件包、类和接口的划分等	指导软件开发人员进行代码的组织和实现	程序员
部署视图	把构件部署到一组物理节点上，表示软件到硬件的映射和分布结构	系统的物理结构，包括硬件设备、网络设备的布局以及软件在硬件上的部署情况	明确软件系统在物理环境中的部署方式和拓扑结构	系统工程师
用例视图	通过用例和参与者来描述系统的功能需求	系统如何被外部使用，重点关注用户交互和系统为用户提供的价值	从用户角度描述系统功能，为系统的需求分析和功能验证提供依据	分析人员和测试人员

2.5.3.UML 的图（重点★★★★★）

UML 2.5 包括 14 种图，这里重点关注用例图，类图，活动图，状态图，顺序图，其他图了解即可，有的同学说最好有对应的图例，这个不太重要，光一个图没啥意义，展开又太多，所以凯恩这里没有添加，不同的图的描述列举如下。这里我们要记住结构图包含哪些，剩下的都是行为图。



分类	名称		描述
结构图	类图	Class Diagram	描述类、接口、属性、操作以及它们之间的关系，如继承、实现、关联和依赖
	对象图	Object Diagram	描述系统在某一时刻的对象实例及对象之间的链接关系，是类图的实例化
	包图	Package Diagram	描述模型中包的组织结构以及包之间的依赖关系
	组合结构图	Composite Structure Diagram	描述类或构件内部的组成结构，以及内部部件之间的协作关系
	构件图	Component Diagram	描述系统构件、接口及其依赖关系，关注系统的模块化结构
	部署图	Deployment Diagram	描述构件在硬件节点上的部署关系及节点之间的通信路径
	轮廓图	Profile Diagram	描述 UML 的扩展机制，如构造型、标记值和约束
行为图	用例图	Use Case Diagram	描述系统的功能需求，展示参与者与用例之间的关系
	活动图	Activity Diagram	描述业务流程或操作流程中的控制流与数据流
	状态机图	State Machine Diagram	描述对象在其生命周期内的状态变化及状态转换条件

行为图 （交互图）	顺序图/序列图	Sequence Diagram	描述对象之间按时间顺序发送消息的过程，强调消息的先后顺序
	通信图/协作图	Communication Diagram	描述对象之间的消息交互，强调对象之间的协作关系
	时间图	Timing Diagram	描述对象状态或消息随时间变化的精确时序关系
	交互概览图	Interaction Overview Diagram	从整体上描述多个交互图之间的执行流程与控制关系

这里唯一要强调的是用例图在 UML2.x 下是一种行为图。书本上没有出现外廓图只有制品图，实际上是 UML2.5 下轮廓图取代了制品图。

2.5.4.UML 的关系（重点★★★★★）

UML 事务的关系就 4 种，依赖，关联，泛化，实现，其他都是引申出来的。展开一下，用例图中的包含和扩展都属于依赖关系。类图中的组合和聚合属于关联关系。后面提到这些关系的时候，你要知道他们是从哪里引申出来的。

关系类型	含义	表示方式
依赖关系	一个元素的改动可能影响另一个使用它的元素，使用具有临时性、偶然性	带箭头的虚线，箭头指向被依赖元素
关联关系	对象间较稳定的结构联系，一个对象能知晓并与另一个对象交互，有单向、双向之分及不同多重性	实线，两端可带箭头表示方向，也可不标，关联线可标注名称、角色名和多重性
泛化关系	即继承关系，子类继承父类的属性和方法，可扩展或重写，体现 “is - a” 关系	带空心三角形箭头的实线，箭头指向父类
实现关系	描述类与接口的关系，类实现接口中定义的所有抽象方法	带空心三角形箭头的虚线，箭头指向接口

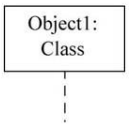
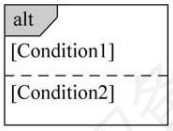
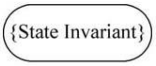
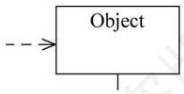
2.5.5.其他常考的 UML 图（超级重点★★★★★）

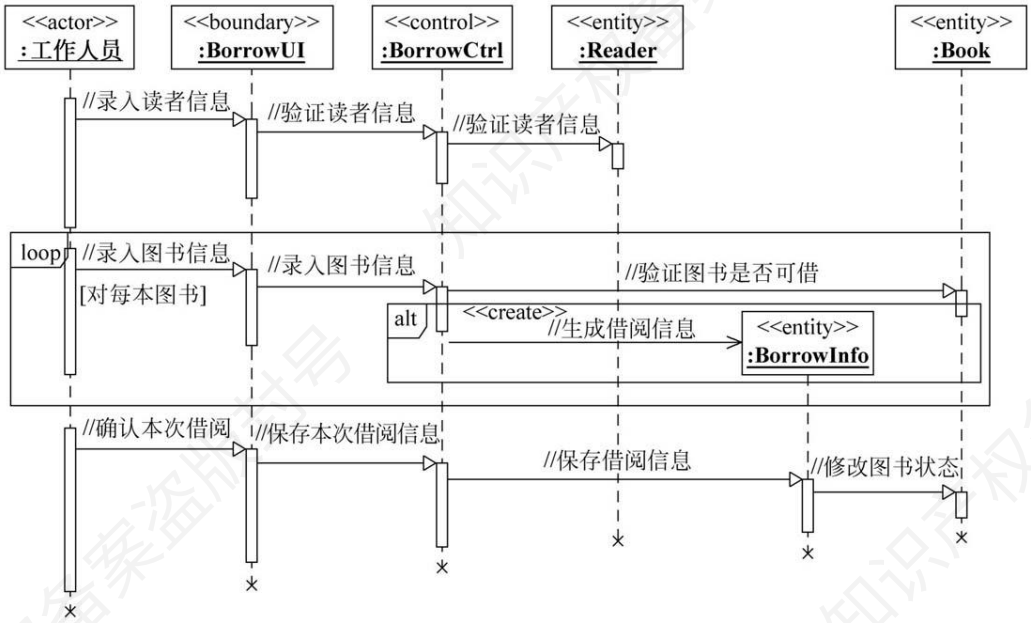
顺序图和活动图是案例常考的内容，放在这里作为补充。特别是它们各自对应的元素、图形符

号都要了解含义。在案例分析中还会展开讨论。

2.5.5.1.顺序图（重点★★★★★）

类别	详情
定义	顺序图是一种最常用的动态交互图，用于显示对象间的交互活动，关注对象之间消息传送的时间顺序
核心概念	对象、生命线、执行发生、消息，交互片段：UML 2 中的新增概念，用于封装交互图中的片段，并可对片段施加如选择、循环、并行等操作，使 UML 支持复杂的交互建模
推荐使用场合	用例分析、用例设计等场合

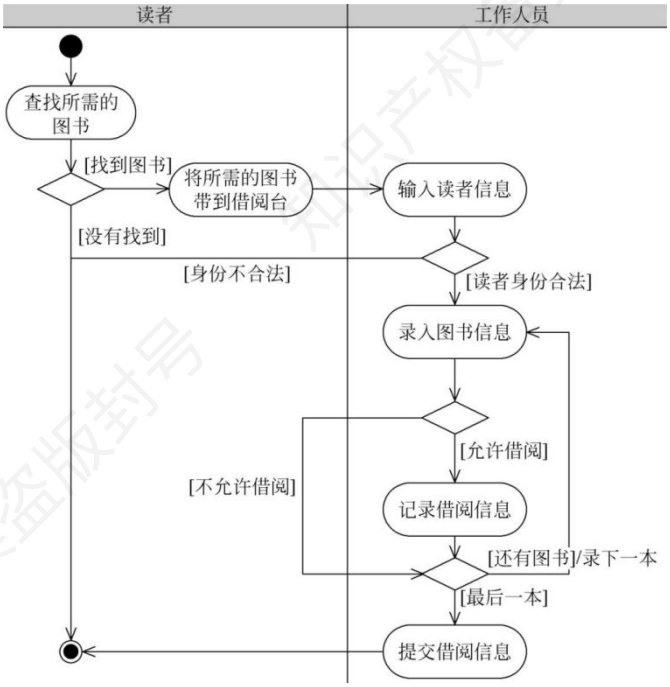
元　素	图形符号	元　素	图形符号
对象/生命线 (Object/Lifeline)		同步消息 (Synchronous Message)	
交互片段 (Interaction Frame)		异步消息 (Asynchronous Message)	
执行发生 (Execution Occurrence)		返回消息 (Return Message)	
状态不变式 (State Invariant)		创建消息 (Create Message)	



2.5.5.2.活动图（重点★★★★★）

类别	详情
定义	将业务流程或其他计算的结构展示为内部一步步的控制流和数据流，主要用于描述某一方法、机制或用例的内部行为
核心概念	活动、组合活动：表示某个内部的控制逻辑 对象、对象流：与活动相关的数据对象 转移、分支：控制活动之间的先后顺序 并发、同步：支持活动间的并发和同步 分区：描述活动的不同参与者
推荐使用场合	业务建模、需求、类设计等场合

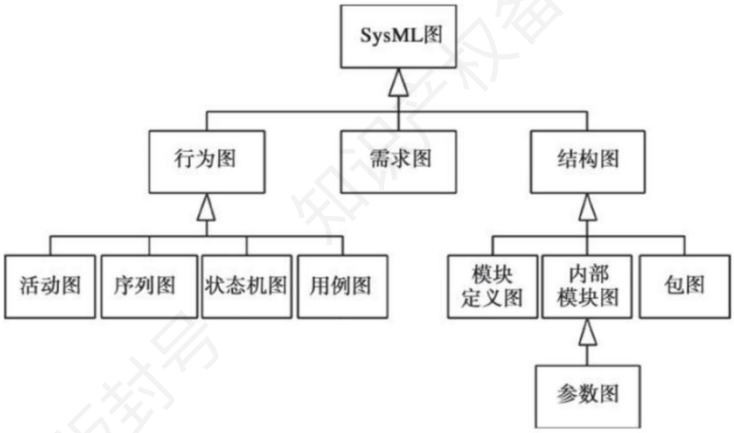
元　素	图形符号	元　素	图形符号
活动/动作(Activity/Action)	Activity	起点(InitialNode)	●
对象(Object)	Object	终点(FinalNode)	◉
发送事件(SendEvent)	Event	流结束(FlowFinal)	⊗
接收事件(AcceptEvent)	Event	分叉/合并(Fork/Join)	———
分区(Partition)		控制流(ControlFlow)	⟶
决策点(Decision)	◇	对象流(ObjectFlow) (在 UML 1.x 中为虚线)	⟶



2.5.6.系统建模语言（SysML）（次重点★★★☆☆）

虽然书上没有 SysML 的相关说明，但是因为系分和架构都考过，所以必须补充。SysML 并不是一种独立的语言。它是 UML 的一种形式或者扩展，而 UML 是特别为系统工程领域所创建的，它扩展了一些新的功能机制，如类图、活动图等，新增了 UML 中没有的图，如模块定义图、需求图、参数图。

特性	UML	SysML
局限性	聚焦单个软件，对多软件及中间件协同关注不足；功能描述不全，难描述非功能性需求	学习曲线陡峭，团队从 UML 转换需适应期；模型易复杂庞大；工具支持相对有限；在简单项目中易过度设计
优势	广泛应用，有大量成熟工具和丰富文档资源；学习资料多，易上手，适合初学者和小型软件项目；图形化表示直观，能清晰描述软件系统的结构、行为和关系	具备系统级视角，可阐述多组件协同；能全面分析需求，兼顾功能与非功能需求；有丰富交互描述元素



SysML 中需求图和用例图的区别在于，两者侧重点不一样。用例图是场景交互的描述，需求图侧重的是需求之间的关系。需求图资料不多，了解即可。

2.6.需求定义（非重点☆☆☆☆☆）

系统架构设计师在获取了用户的需求，并进行了详细分析之后，接下来的工作就需要把这些需求形成文档，作为系统后续开发的基础，这就是需求定义（或者说需求基线）。这里你要知道需求定义的产物是软件需求规格说明书（SRS）。

通常有两种需求定义的方法，分别是严格定义方法和原型方法（不做展开）。软件需求规格说明书（SRS） 是需求开发活动的产物，编制该文档的目的是使项目干系人与开发团队对系统的初始规定有一个共同的理解，使之成为整个开发工作的基础。

2.7.需求验证（非重点☆☆☆☆☆）

在系统分析阶段，检测 SRS 中的错误所采取的任何措施都将节省相当多的时间和资金。 因此，有必要对于 SRS 的正确性进行验证，以确保需求符合良好特征。需求验证也称为需求确认。

2.7.1.需求评审（非重点☆☆☆☆☆）

在软件开发的每个阶段结束前，都需要进行技术评审。所谓技术评审，是指对工作产品进行检查以发现产品中所存在的问题。SRS 的评审可以发现那些二义性的或不确定性的需求，为项目干

系人提供在需求问题上达成共识的方法。具体的评审类型如下所示。

三者最大的区别是评审是征求干系人专家意见（正式），检查是详细验证（正式+严格），走查是带领团队成员逐部分讲解（非正式）。

类型	定义
评审	一次正式的会议，在会议上向用户或其他项目干系人介绍一个或一组工作产品，以征求对方的意见和批准
检查	一种正式的评估方法，将由非制作者本人的个人或小组详细检查工作产品，以验证是否有错误、是否违反开发标准、是否存在其他问题
走查	一个评审过程，由某个开发人员领导一个或多个开发团队成员对他（或她）的工作产品进行检查，由其他成员针对技术、风格、可能的错误、是否违反开发标准和其他问题提出意见

2.7.2.需求测试（非重点☆☆☆☆☆）

软件测试应该从需求定义开始,如果在开发过程的早期就开始制订测试计划和进行测试用例的设计，就可以在发生错误时立即检测到并纠正它。实际上，需求开发阶段不可能有真正意义上的测试进行，因为还没有可执行的系统，需求测试仅仅是基于文本需求进行“概念”上的测试。

2.8.需求管理（重点★★★★★）

在前面提到的软件能力成熟度模型（CMM）中，需求管理是可重复级的一个关键过程域，其目标是为软件需求建立一个基线，供软件开发及其管理使用，使软件计划、产品和活动与软件需求保持一致。这里书本上要求掌握的是变更管理、风险管理和需求跟踪，有的书籍分的层次略有区别，如下图所示。



2.8.1.需求变更管理（非重点☆☆☆☆☆）

这个小节主要讲需求变更的流程，选择题偶尔出个选择，不重要。

2.8.1.1.需求基线（非重点☆☆☆☆☆）

需求开发的结果应该有项目视图和范围文档、用例文档和 SRS，以及相关的分析模型。

根据 IEEE 的定义，基线是指已经通过正式评审和批准的规约或产品，它可以作为进一步开发的基础，并且只能通过正式的变更控制系统进行变化。在软件工程范围内，基线是软件开发中的里程碑，其标志是有一个或多个软件配置项的交付，且已经经过正式技术评审而获得认可。

2.8.1.2.需求变更（非重点☆☆☆☆☆）

在需求变更之前尽量减少变更，以将需求变更带来的风险降到最低。关于需求变更两种说法，看到都是正确的：

说法 1：变更管理的流程，包括变更申请、变更评估、通告评估结果、变更实施、变更验证与确认、变更发布。

说法 2：识别问题、问题分析和变更描述、变更分析和成本计算、变更实现。

2.8.2.需求风险管理（次重点★★★☆☆）

风险管理历年很少考到，凯恩认为这次有可能放在案例或者论文中考察。架构同学可以忽略，系分同学要看一下。特别是与需求有关的风险需要了解。

2.8.2.1.带有风险的做法（次重点★★★☆☆）

带有风险的做法列举如下，这些内容系分同学了解即可，可能案例里让你默写：

（1） 无足够用户参与。（2） 忽略了用户分类。（3） 用户需求的不断增加。（4） 模棱两可的需求。（5） 不必要的特性。（6） 过于精简的 SRS。（7）不准确地估算。

其中 12 是用户类的，345 是需求类的，6 是文档类的，7 是费用工期估算类的。

2.8.2.2.与需求有关的风险（次重点★★★☆☆）

根据业内人士的经验，与需求有关的主要风险及其应对措施如表所示。从获取、分析、定义、验证和管理都要有风险应对措施。了解即可。

阶段	主要风险	风险应对措施
需求获取	在项目早期写一份项目视图与范围将业务需求涵盖在内，并将其作为新的需求及修改需求的指导	记录参与的每个项目中实际需求开发的工作量，这样就能知道所花的时间是否合适，并改进将来项目的工作计划
	忽略市场对产品的反馈信息	强调市场调查研究，建立原型，并运用客户核心小组来获得产品的反馈信息
	没有非功能需求	编写非功能需求文档和验收标准，作为可接受的标准
	客户反对产品需求	确定出主要的客户，并采用产品代表的方法来确保客户代表的积极参与，并确保在需求决定权上有正确的人选
	期望需求	尽量识别并记录用户的期望，提出大量的问题来提示用户，以充分表达他们的想法和建议
	把已有的产品作为需求基线	将在逆向工程中收集的需求编写成文档，并让用户评审以确保其正确性
	给出期望的解决办法	从用户描述的解决办法中提炼出其本质需求

需求分析	划分需求优先级	评估每项新需求的优先级，并与已有的工作对比，以做出相应的决策
	带来技术困难的特性	分析每项需求的可行性，以确定是否能按计划实现
	不熟悉的技术、工具 / 平台	明确那些高风险的需求，并留出充裕时间进行学习、实验和测试使用
	系统分析师和用户对需求的不同理解	使用高水平的系统分析师；使用模型和原型，使一些模糊的需求变得清晰
需求定义	时间压力对待确定因素的影响	记录解决每项待确定因素的负责人的名字、如何解决的，以及解决的截止日期
	SRS 的完整性和正确性	以用户的任务为中心，采用用例技术获取需求；根据场景编写需求测试用例，建立原型；让用户代表对 SRS 和分析模型进行正式评审
	具有二义性的术语	建立一本术语和数据字典，用于定义所有的业务和技术词汇
	需求说明中包括了设计	仔细评审 SRS，以确保它是在强调 “做什么”，而不是 “怎么做”
需求验证	未经验证的需求评审	从用户代表方获得参与需求正式评审的承诺，并尽早通过非正式评审
	审查的有效性	对参与需求评审的所有人员进行培训，以使评审工作更加有效
需求管理	需求变更	将项目视图与范围文档作为变更的参照；用户积极参与需求获取过程；将那些易于变更的需求用多种方案实现，并在设计时注意其可修改性
	需求变更过程	建立规范的变更控制流程，并严格执行
	未实现的需求	使用需求跟踪能力矩阵或相关工具
	项目范围蔓延	在项目早期编制视图与范围文档，并得到用户确认；采用迭代式开发方法

2.8.3.需求跟踪（次重点★★★☆☆）

什么是需求跟踪了解即可，不太会直接考定义。需求跟踪是将单个需求和其他系统元素之间的依赖关系和逻辑联系建立跟踪，这些元素包括各种类型的需求、业务规则、系统架构和构件、源代码、测试用例，以及帮助文件等。

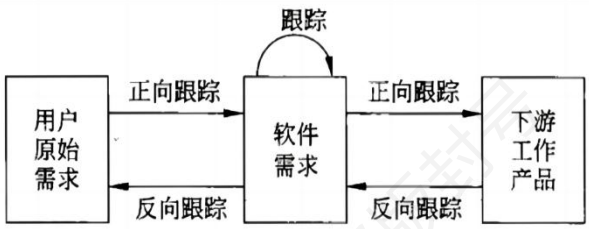
2.8.3.1.需求跟踪的内容（次重点★★★☆☆）

根据国家标准 GB/T 8567—2006， SRS 中的每个软件配置项的需求到其涉及的系统（或子系

统）需求都要具有双向可追踪性。所谓双向跟踪，包括正向跟踪和反向跟踪。这两个是最重要的。跟踪的目的就是确保所有的需求都是被定义并且实现的。

跟踪类型	概念定义	例子
正向跟踪	检查 SRS 中的每个需求是否都能在后继工作成果中找到对应点	检查设计文档，查看商品详情页的页面设计是否包含加入购物车按钮，并且按钮的交互逻辑是否符合需求描述，确保从 SRS 到设计文档有对应点。
反向跟踪	检查设计文档、代码、测试用例等工作成果是否都能在 SRS 中找到出处	对于设计文档中商品详情页的设计，也能在 SRS 中找到对商品详情页功能及相关元素（如加入购物车按钮）的需求描述，知道设计是基于 SRS 的哪些需求。

实际上的需求跟踪工作流如下图所示。



2.8.3.2.需求跟踪的目的（非重点☆☆☆☆☆）

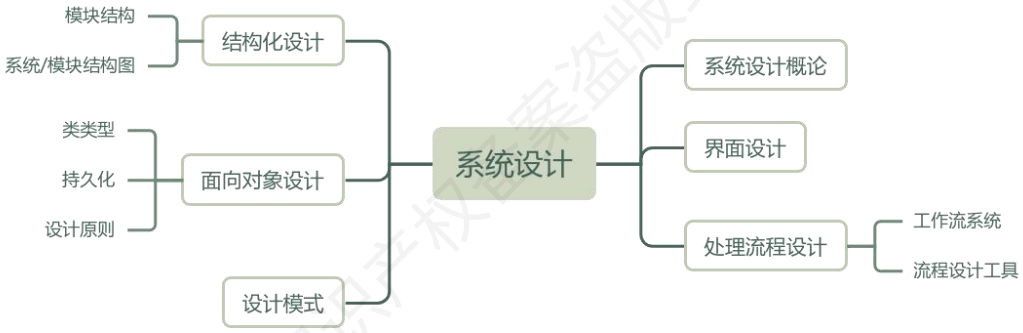
在项目实践中，使用需求跟踪能力，可以获得如下好处，不重要，没有考过，了解即可。

方面	描述
审核	跟踪能力信息帮助开发人员审核并确保所有需求都被正确应用
变更影响分析	增、删、改需求时，跟踪能力信息确保不忽略每个受影响的系统元素
维护	可靠的跟踪能力信息使维护时能正确完整实施变更，提高生产率
项目跟踪	认真记录跟踪能力数据，可获得计划功能当前实现状态的记录
再工程	列出遗留系统中要替换的功能，记录其在新系统中的需求及在软件构件中的位置
重复利用	跟踪能力信息帮助开发人员在新系统中对相同功能利用现有系统相关资源，如功能设计、需求、代码和测试等
减小风险	需求联系文档化减少因项目团队关键成员离职带来的风险
测试	测试模块、需求和代码段间的联系链在测试出错时指出最可能有问题的代码段

3.系统设计（超级重点★★★★★）

系统设计这里的内容架构和系分都是重点中的重点，属于不看必挂系列。这里主要涉及结构化设计和面向对象设计的基本知识。照道理这一块应该写得很深，很系统，但是遗憾的是，书本这块说得语焉不详，故凯恩这里尽可能保留书上一些概念，并以《大话软件工程》的部分概念作为补充，尽可能帮大家梳理清楚脉络。

此章节从考察形态上来看，选择、案例部分多考察面向对象设计和结构化设计的相关概念（必须非常熟悉，下划线标出都要背出），处理流程设计考察的相对偏少。论文部分也是高频考点，以最新的红宝书论文押题为准。思维导图如下。



3.1.系统设计在软件工程中的位置（非重点★☆☆☆☆）

首先还是要明确系统设计的定义，这里经过我的调研实际上也有狭义和广义之分。狭义的系统设计单纯是指概要设计/整体设计/蓝图设计。而软考教材上的定义，是广义的系统设计的定义。它不单单包括概要设计，还包括具体构建系统的详细设计方案和相关模型。这个我们需要了解，以免你在看到其他图书的时候产生疑惑。

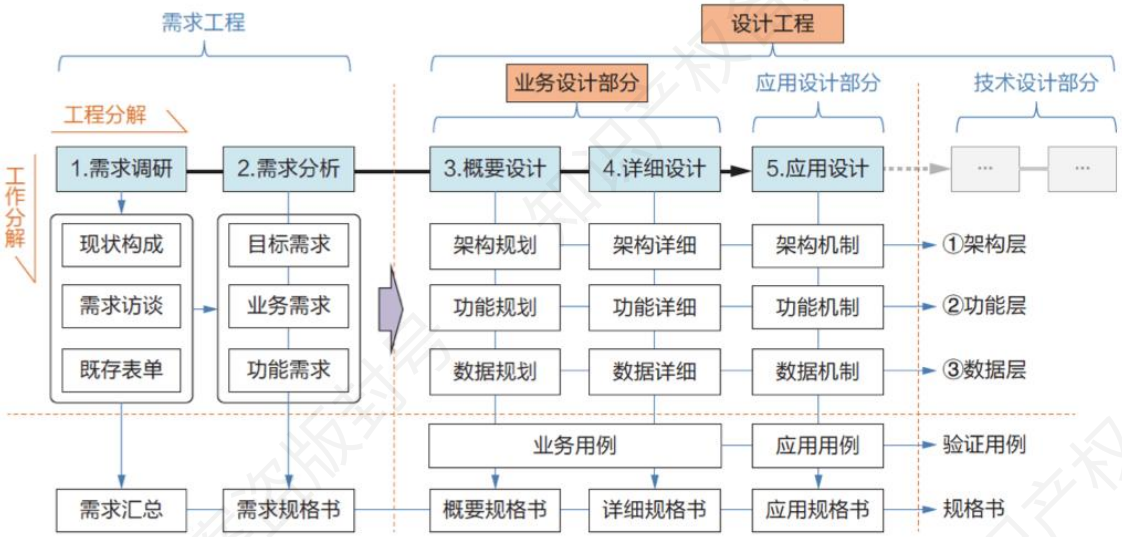


图8-1　设计工程在软件工程结构中的位置

从这张图里（出自大话软件工程）我们可以看到，设计工程是指在整个软件开发过程中，通过对需求工程的结果进行分析和设计，制定出系统的整体设计方案，包括概要设计、详细设计、应用设计等多个部分。其中，概要设计是对系统进行高层次的规划和设计，详细设计则是对系统中各个模块的具体实现方式进行确定；应用设计则主要是针对特定业务场景进行设计。并且概要设计、详细设计、应用设计都可以纵向分解成架构、功能和数据层，这个是符合直觉的。

书上有提到的就是概要设计和详细设计。概要设计是指在整个软件开发过程中，对系统进行高层次的规划和设计，确定系统的基本结构、组成要素以及它们之间的关系。它主要关注的是系统的大体框架和基本功能，着重解决系统应该如何组织、如何实现等问题。

详细设计（也称为模块设计）是指在概要设计的基础上，进一步深入到系统细节层面，确定系统中各个模块的具体实现方式、接口定义、算法实现等内容。它主要关注的是系统内部的细节实现，着重解决系统应该如何完成某个具体任务、如何处理某个具体的数据等问题。

3.2.系统设计概述（次重点★★★☆☆）

系统设计按照对象不同可以分为以下 7 类。这个了解即可，只会考选择题。

设计类型	主要任务 / 目的

网络设计	根据系统要求选择网络结构，安排设备分布、布线部署，划定节点权限，选择系统及管理软件。
代码设计（重点）	包括面向对象设计和结构化设计。实现代码的唯一化、规范化、系统化。
输入设计	确保输入数据的完整性、正确性、一致性。
输出设计	确保输出数据的完整性、正确性、一致性。
处理流程设计（重点）	确定各系统模块的内部结构（局部数据组织、控制流、加工过程与细节）。
数据存储设计	选择存储方式、介质、组织方式，估算容量，满足业务与管理需求。数据库设计（概念设计、逻辑设计、物理设计）
用户界面设计	优化系统与用户的交互体验，引导用户操作。黄金三原则：置于用户控制之下、减轻用户记忆负担、保持界面一致性。
安全性和可靠性设计	确保系统安全、有效运行，对运行环境和数据处理进行控制。

3.3.处理流程设计概述（次重点★★★☆☆）

处理流程设计的任务是设计出系统所有模块以及它们之间的相互关系，并具体设计出每个模块内部的功能和处理过程，为开发人员提供详细的技术资料。这是整个章节中较为重要的内容，喜欢出选择题。

3.3.1.流程设计（次重点★★★☆☆）

流程设计包含 4 个核心概念，如下表所示。重点要看的是这里的流程的内容，下一小节展开说。

类别	定义	例子
流程	ISO 9000 定义：一组将输入转化为输出的相互关联或相互作用的活动。	在线教育平台 “开通课程” 流程：输入为注册用户名，活动含合法性判断等，输出为开通通知，用户为付费注册用户。

工作流	WFMC 定义：可部分 / 完全自动执行的业务过程，基于规则在执行者间传递文档、任务等。	课程申请单流转：员工填写申请单→主管审批→培训部门备案→财务核算，各环节按规则自动传递。
活动及其所有者	活动：流程基本要素，改变数据 / 状态，推动流程（人或系统完成）；所有者：有权结束活动并推动流程的参与者。	审批活动：输入为课程申请单，处理含合规性判断，输出为审批结果；所有者为部门主管，可将申请单转下一环节或退回。
工作项	流程实例中参与者需执行的具体工作。	某员工创建的 “软考课程申请单”，在系统中以唯一编号标识，是主管审批环节的具体待办工作。

3.3.1.1.什么是流程（次重点★★★☆☆）

ISO 9000 定义业务流程为一组将输入转化为输出的相互关联或相互作用的活动。一般来说，流程包括 6 个基本要素，分别是输入资源、活动、活动的相互作用（结构）、输出结果、用户和价值。

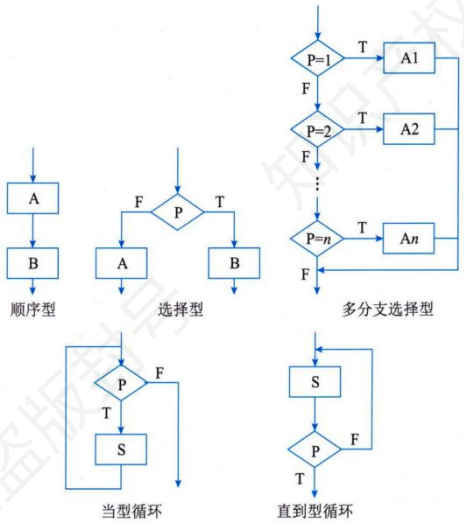
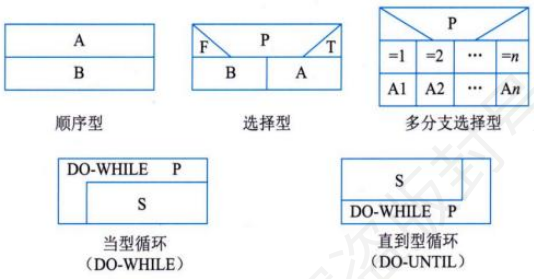
例如，在线教育平台系统中的“开通课程”流程，其有 6 个要素。如表所示。这个点系统分析师有出案例的可能。比如告诉你某个流程让你根据 6 个基本要素将其分解。

要素名称	要素含义
输入资源	需要开通课程的注册用户名
活动	开通课程的业务逻辑（例如，用户名合法性判断、时间判断和费用计算等）
活动的相互作用	开通课程与其他活动（例如，在线测试等）流程的相互关系
输出结果	开通课程成功后获取的短消息通知和电子邮件通知
用户	已交纳课程学习费用的注册用户
价值	用户可通过该流程实现学习课程的功能

记忆口诀：输入输出活动（相互作用）用户价值。

3.3.2.流程设计工具（次重点★★★☆☆）

在处理流程设计过程中，为了清晰表达过程规则说明，常用的流程设计工具分为三类：图形工具、表格工具和语言工具。这里的内容目前还是在选择题中考察概念为主，不会让你去作图。

大类	工具名称	描述	图例
图形工具	程序流程图 (PFD)	用图框表示各种操作，独立于任何程序设计语言，直观且易于掌握，但符号不规范、随意转移控制的问题需要改进。	 图 13-2 程序流程图的 5 种基本控制结构
	N-S 图（盒图）	以方框代替传统的 PFD，包括顺序型、选择型、WHILE 循环型、UNTIL 循环型和多分支选择型控制结构，具有强烈的结构化特征。	 图 13-4 N-S 图的 5 种基本控制结构

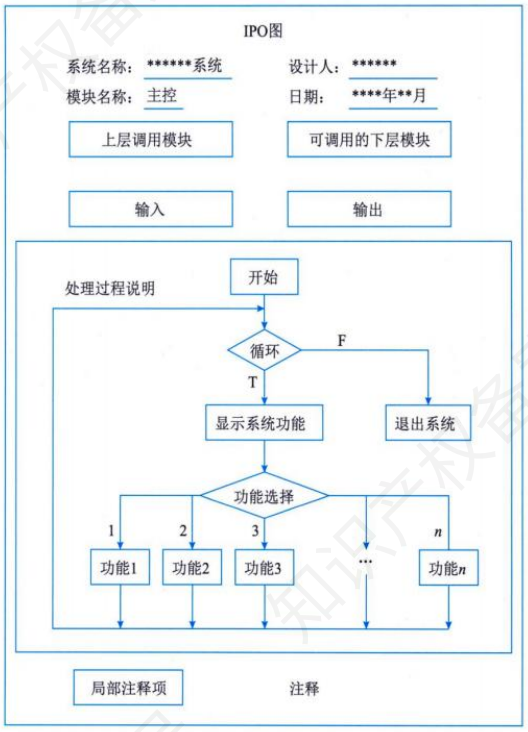
<table>
<tr><td rowspan="3"></td><td>IPO 图</td><td>描述模块的输入、输出和数据加工，结构清晰，常用于系统设计文档。</td><td>

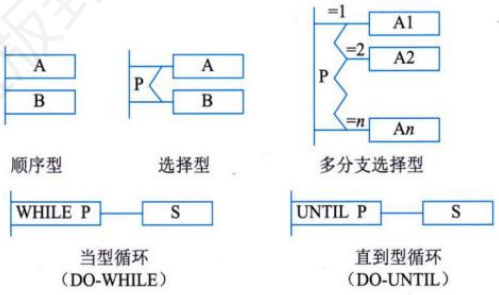
图 13-3 IPO 基本结构图</td></tr>
<tr><td>问题分析图 (PAD)</td><td>由日立公司提出，支持结构化程序设计，包含五种基本控制结构，执行顺序明确，适合嵌套和层次关系的表示。</td><td>

图 13-5 PAD 的 5 种基本控制结构</td></tr>
<tr><td>判定树</td><td>用树形结构表示逻辑判断，条件和处理流程直观，适合复杂条件判断。</td><td>

图 13-6 判定树描述的某公司折扣策略</td></tr>
<tr><td>表格工具</td><td>判定表</td><td>结构清晰、简明，用表格形式表示逻辑判断问题，条件和行动一目了然，适合多个互相联系的条件和多种结果的描述。</td><td>表 13-2 判定表描述的某公司折扣策略
<table>
<tr><th>不同条件组合
条件和行动</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th></tr>
<tr><td>C1：每年交易额在 50 000 元以上</td><td>T</td><td>T</td><td>T</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>C2：最近 3 个月无欠款</td><td>T</td><td>T</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td></tr>
<tr><td>C3：与本公司交易 20 年及以上</td><td>T</td><td>F</td><td>T</td><td>F</td><td>T</td><td>F</td><td>T</td><td>F</td></tr>
<tr><td>A1：折扣率 15%</td><td>Y</td><td>Y</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>A2：折扣率 10%</td><td></td><td></td><td>Y</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>A3：折扣率 5%</td><td></td><td></td><td></td><td>Y</td><td></td><td></td><td></td><td></td></tr>
<tr><td>A4：无折扣率</td><td></td><td></td><td></td><td></td><td>Y</td><td>Y</td><td>Y</td><td>Y</td></tr>
</table></td></tr>
<tr><td>语言工具</td><td>过程设计语言 (PDL)</td><td>也称伪代码，混合自然语言词汇和结构化程序设计语言语法，用于描述处理过程，灵活</td><td></td></tr>
</table>

		且有助于逐步求精和详细设计。	

下面是一道真题，因为程序流程图（PFD）主要用于描述程序的逻辑流程，包括各个步骤的执行顺序、判断条件等，而不是用于描述系统中每个模块的输入、输出和数据加工，该项说法错误。



3.3.3.工作流管理系统（次重点★★★☆☆）

扯开说一句，书本在流程设计中提到工作流管理系统（Workflow Management System，WfMS），主要是因为它是将流程从“纸面上的设计”落到“可执行、可监控的系统化管理”中的关键纽带。可以理解为把流程程序化系统化。这个章节爱出选择题，主要是考察概念（系分特别爱考），真题考过的概念要掌握。

工作流管理系统（WFMS）通过软件定义、创建工作流并管理其执行。它运行在一个或多个工作流引擎上，这些引擎解释对过程的定义与工作流的参与者相互作用，并根据需要调用其他 IT 工具或应用。

WFMS 基本功能包括对工作流建模（定义活动和规则，对应现实业务处理过程）、工作流执行（创建和执行实际工作流）、业务过程管理和分析（监控管理执行中的业务情况）。

记忆口诀：简直过分。简（建摸）直（执行）过（过程）分（分析）。

工作流参考模型（WRM） 包含 6 个基本模块，分别是工作流执行服务、工作流引擎、流程定义工具、客户端应用、调用应用和管理监控工具。工作流参考模型（WRM） 为工作流管理系统（WFMS）关键模块提供了功能描述，并描述了关键模块之间的交互，而且这个描述是独立于特定产品或技术的实现的。

记忆口诀：指引丁克调管。指（执行）引丁（定义）克（客户端）调（调用）管（管理）。

3.4.面向对象设计（重点★★★★★）

面向对象设计和下面的结构化设计都属于软件设计。这个部分主要考察面向对象设计的概念，划线部分可能会在选择中考察。

面向对象设计（OOD）是面向对象分析（OOA） 方法的延续，其基本思想包括抽象、封装和可扩展性，其中可扩展性主要通过继承和多态来实现。在面向对象设计中，数据结构和在数据结构上定义的操作算法封装在一个对象之中。

3.4.1.设计类分类（重点★★★★★）

这是面向对象的设计中的重点，也是整个章节的重点。一般会在案例中让你判定什么是实体类，什么是边界类，什么是控制类。具体的判定方法如下表所示。

维度	实体类	控制类	边界类
定义	映射需求中的每个实体，保存需永久存储在存储体中的信息	用于控制用例工作，对用例特有的控制行为建模	封装用例内外流动的信息或数据流，处于系统与外界交界处
命名规则	采用业务领域术语，一般为名词	由动宾结构短语转化而来的名词	与系统接口紧密相关，无固定规则但能体现外部交互功能
作用	代表业务领域实际对象，对用户有意义，呈现业务模型重要元素	协调用例执行时不同对象间的交互，确保用例按逻辑推进	实现系统与外界交互，隔离外部环境变更对系统内部的影响

生命周期	通常具有永久性，只要业务存在其信息就需持续保存	系统执行用例时产生，用例执行完毕后消亡	随系统运行一直存在，在系统与外部交互时发挥作用
属性与方法	一定有属性，操作依业务需求而定，不一定存在	通常没有属性，但一定有方法	可以既有属性也有方法
常见示例	在线教育平台的学员类、课程类	如“身份验证”对应的“身份验证器”	窗口、通信协议、打印机接口、报表等

一般判断的时候，我们就先看是否是（窗口、通信协议、打印机接口、报表等），这些都是边界类，剩下的用命名规则判断，名词是实体类，动名词就是控制类。

3.4.2.面向对象设计原则（重点★★★★★）

这块知识以往考过论文，案例也有可能会考 SOLID 相关概念。

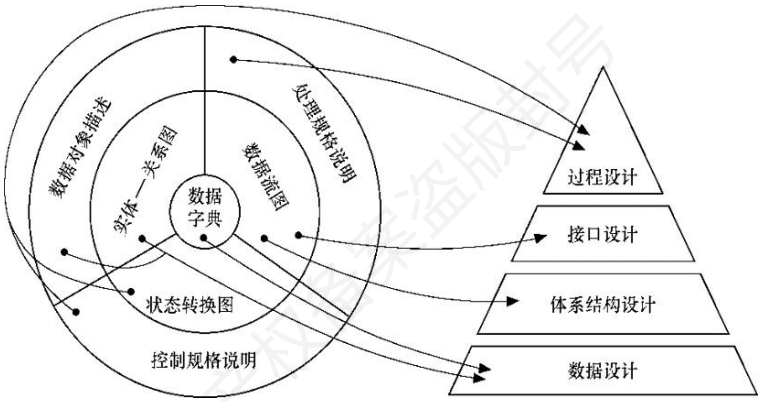
在面向对象设计中，可维护性的复用是以设计原则为基础的。常用的面向对象设计原则包括单一职责原则（S）、开闭原则、里氏替换原则、依赖倒置原则、组合/聚合复用原则、接口隔离原则和最少知识原则等。每个原则具体是什么意思需要记忆。

设计原则	详细描述
单一职责原则（S）	单一职责原则要求类应该只有一个设计目的
开闭原则（O）	可扩展系统并提供新行为，通过添加抽象层实现，是其他原则的基础。对扩展开放对修改封闭。
里氏替换原则（L）	软件实体使用基类对象时适用于子类对象，设计时应将变化类设计为抽象类或接口。
接口隔离原则（I）	分逻辑和狭义两种理解，前者将接口视为角色，后者要求接口提供客户端需要的行为，将大接口方法分至小接口。
依赖倒置原则（D）	抽象不依赖于细节，针对接口编程，是实现开闭原则的主要机制，与各种技术和框架相关。
组合/聚合复用原则	通过组合或聚合关系复用已有对象，比继承更灵活，耦合度低。
最少知识原则（迪米特法则）	软件实体应尽量少与其他实体相互作用，分狭义和广义原则，狭义降低类耦合但可能影响通信效率，广义主要控制信息相关方面，利于子系统解耦和复用。

3.5.结构化设计（重点★★★★★）

结构化设计这里比较难记的是耦合和内聚的顺序。凯恩会给你记忆口诀一定要去参透。考试选择遇到的时候套口诀可以直接秒杀的，那种感觉真的很美妙。

结构化设计（Structured Design，以下简称 SD） 是一种面向数据流的方法，它以软件需求规格说明（SRS），系统分析（SA）阶段所产生的数据流图和数据字典等文档为基础，是一个自顶向下、逐步求精和模块化的过程（书本上这里又引入概要设计和详细设计，这个会令人迷惑，所以这里省去，凯恩在系统设计开篇已经把这两个概念梳理清楚了）。下图是结构化分析的若干工具和结构化设计的映射关系，有个概念就行，不需要死记硬背。



软件设计模型/结构化设计——包括四个既独立又相互联系的活动——数据设计、软件结构设计、人机界面设计（接口设计）和过程设计。这个概念喜欢在选择题考察，希望大家记住。

3.5.1.模块结构（重点★★★★★）

选择和案例会对模块划分原则进行考察。记住模块大小适中，模块扇入扇出合理，深度和宽度适当，信息隐蔽即可。在模块的分解中应尽量减少模块的耦合，力求增加模块的内聚，遵循 “高内聚、低耦合”的设计原则。

模块划分原则	具体说明

模块大小适中	系统分解时考虑模块规模，过大模块可能分解不充分，功能不单一；过小模块增加系统复杂度，调用频繁，独立性降低。一般模块实现代码 1 - 2 页纸内或 50 - 200 行，易于实现和维护
模块扇入扇出合理	扇出指模块直接调用下级模块个数，扇出大复杂度高，需控制协调下级模块，过大因缺中间层次可增加控制模块，过小可分解下级模块或合并到上级模块；扇入指直接调用该模块的上级模块个数，扇入大复用程度高。系统平均扇入和扇出系数为 3 或 4，不超 7，否则易出错
深度和宽度适当	深度指模块层数，过多考虑模块是否简单可合并；宽度指同一层次模块总数最大值，宽度大系统复杂，模块产出对宽度影响大
信息隐蔽	采用封装技术隐藏模块实现细节，使接口简单，模块设计成 “黑盒”，仅通过接口交互，相对独立，易实现、理解和维护；抽象原则抽取事物基本特性和行为，忽略细节，分层次抽象控制开发复杂性，包括过程抽象、数据抽象和控制抽象

3.5.2.耦合（重点★★★★★）

这一小节，比较喜欢出选择题，考察的也是你的记忆力，熟读下方口诀，选择题就是 1 分。模块的耦合类型通常分为 7 种，你只要记住非直接耦合是最低的耦合，内容耦合是最高的耦合即可。

耦合表示模块之间联系的程度。紧密耦合表示模块之间联系非常强，松散耦合表示模块之间联系比较弱，非耦合则表示模块之间无任何联系，是完全独立的。模块的耦合类型通常分为 7 种，根据耦合度从低到高排序如表所示

耦合类型	**描 述**
非直接耦合（低）	两个模块之间没有直接关系，它们之间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	一组模块借助参数表传递简单数据值
标记耦合	一组模块通过参数表传递记录信息（是指数据结构）
控制耦合	模块之间传递的信息中包含用于控制模块内部逻辑的信息
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	多个模块都访问同一个公共数据环境，公共的数据环境可以是全局数据结构、共享的通信区、内存的公共覆盖区等
内容耦合（高）	一个模块直接访问另一个模块的内部数据：一个模块不通过正常入口转到另一个模块的内部：两个模块有一部分程序代码重叠：一个模块有多个入口

记忆口诀：非鼠标控制外公嘞。非（非直接）要用鼠（数据）标（标记）控（控制）外（外部）公（公共）嘞（内容）。

3.5.3.内聚（重点★★★★★）

这一小节，比较喜欢出选择题，考察的也是你的记忆力。模块的内聚类型也分为 7 种，特别注意功能内聚和偶然内聚。耦合这一块书上写的不多，但是这一块其实对于你在实际开发中组织模块分别代码好恶有着非常大的好处。

内聚表示模块内部各成分之间的联系程度，是从功能角度来度量模块内的联系，一个好的内聚模块应当恰好做目标单一的一件事情。模块的内聚类型通常也可以分为 7 种，根据内聚度从高到低的排序如表所示。这里需要注意的是顺序和过程最大的区别是顺序内聚的模块各个成分和同一个功能密切相关，并且一个成分的输出作为另一个成分的输入，存在数据传递和依赖关系；而过程内聚中构件或操作之间即使没有数据传递也可组合在一起，不一定存在数据上的依赖。

内聚类型	**描 述**
功能内聚（高）	完成一个单一功能，各个部分协同工作，缺一不可
顺序内聚	处理元素相关，而且必须顺序执行（存在数据传递和依赖关系）
通信内聚	所有处理元素集中在一个数据结构的区域上
过程内聚	处理元素相关，而且必须按特定的次序执行（不存在数据传递和依赖关系）
瞬时内聚（时间内聚）	所包含的任务必须在同一时间间隔内执行
逻辑内聚	完成逻辑上相关的一组任务
偶然内聚（巧合内聚）（低）	完成一组没有关系或松散关系的任务

记忆口诀：公孙铜锅涮罗欧。公（功能）孙（顺序）铜（通信）锅（过程）涮（瞬时）罗（逻辑）欧（偶然）。

3.6.设计模式（次重点★★★☆☆）

此章节 90%的概率只会在选择题中出现，小概率在案例和论文出现。不需要花太多时间。难度跨度很大，最简单的就是设计模式的分类以及辨别，稍难一点的是给你场景，让你判断用哪种设计模式实现最佳，或者是给你一个类图，让你判别是哪种设计模式。凯恩的建议是，假如你不是做后端开发的，不吃这口饭的，最基础的辨别和分类知道就行。

设计模式是前人经验的总结，它使人们可以方便地复用成功的设计和架构。设计模式这块展开能写一本书，凯恩也很为难，为此我把《深入理解设计模式》这本书放在会员网盘里，想要进一步了解的同学可以看看。

模式大类	模式名称	具体解释（了解即可）
创建模式	工厂方法	定义一个接口用于创建对象，但由子类决定实例化哪个类。
	抽象工厂	提供一个接口以创建相关或依赖对象的家族，而无需明确指定具体类。
	生成器	将一个复杂对象的构建与其表示分离，使同样的构建过程可以创建不同的表示。
	原型	通过复制现有对象来创建新对象，而不是从头开始实例化。
	单例	确保一个类只有一个实例，并提供全局访问点
结构模式	适配器	将一个类的接口转换为客户希望的另一个接口，使原本接口不兼容的类可以一起工作。
	桥接	将抽象部分与它的实现部分分离，使它们都可以独立变化
	组合	将对象组合成树形结构以表示“部分-整体”的层次结构，使客户可以统一处理单个对象和组合对象
	装饰	动态地给对象添加职责，而不改变其接口
	外观	为子系统中的一组接口提供一个统一的高层接口，使子系统更容易使用。
	享元	通过共享技术来有效支持大量细粒度对象的复用。
	代理	为另一个对象提供一个代理或占位符以控制对它的访问。

行为模式	责任链	使多个对象都有机会处理请求，从而避免请求的发送者和接收者之间的耦合。
	命令	将请求封装成对象，以便使用不同的请求、队列或日志来参数化其他对象。
	迭代器	提供一种方法顺序访问一个聚合对象中的各个元素，而不暴露其内部表示。
	中介者	用一个中介对象来封装一系列对象交互，使对象不需要显式地相互引用，降低它们的耦合度。
	备忘录	在不破坏封装的前提下，捕获并外部化对象的内部状态，以便以后恢复到该状态。
	观察者	定义对象间一种一对多的依赖关系，使得每当一个对象改变状态，所有依赖它的对象都会得到通知并自动更新。
	状态	允许对象在内部状态改变时改变其行为，对象看起来好像修改了其类。
	策略	定义一系列算法，将每一个算法封装起来，并使它们可以互换。
	模板方法	在一个方法中定义一个算法的骨架，而将一些步骤延迟到子类中，使子类可以不改变算法结构即可重定义该算法的某些步骤。
	访问者	表示一个作用于某对象结构中的各元素的操作，使你可以在不改变各元素类的前提下定义作用于这些元素的新操作。
	解释器	给定一个语言，定义其文法的一种表示，并定义一个解释器，用于处理该语言中的句子。

为了方便记忆，给大家编一些打油诗，记住了这个打油诗，你分类的最基础的选择题就能做。

（结构）试桥组装外享代
（行为）观察命令责任链，迭代模板状态变。中介策略备忘录，解释访问都包含。
（创建）工厂抽象建造者，单例原型巧构成。

4.系统架构设计（超级重点★★★★★）

此章节属于重点中的重点，属于不看必挂的章节。选择题，案例，论文都会进行重点考察。凯恩这里会把质量属性，质量效用树，评估都放在这里来方便一把梭。选择题部分每年至少 8-10 分，常考架构风格，ABSD，ATAM，DSSA 相关概念，质量属性，架构评估，必须熟记，背出。案例部分主要考察质量属性识别，软件架构风格之间的对比。几乎一年一题。考到就是送分。论文部分，架构系分都有考场。这里架构系分内容略有差别，在红宝书部分我会兼顾两者的一些差异，把它们

融合在一起。

4.1.基本定义（重点★★★★★）

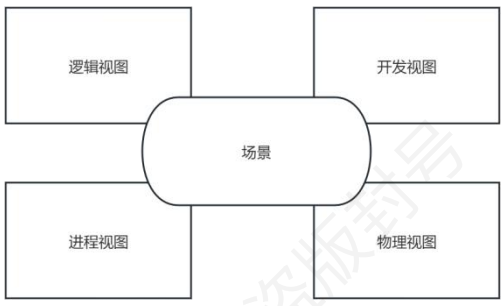
信息系统架构是关于软件系统的结构、行为和属性的高级抽象。在描述阶段,其对象是直接构成系统的抽象组件以及各个组件之间的连接规则,特别是相对细致地描述组件之间的通信。

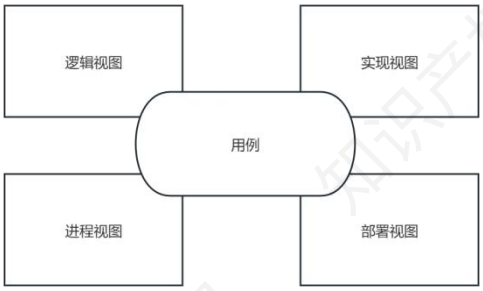
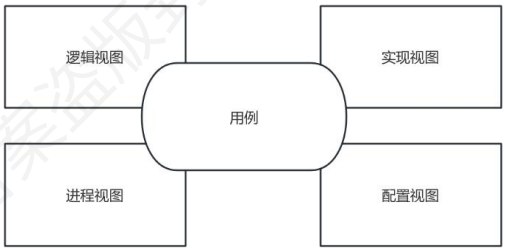
在实现阶段,这些抽象组件被细化为实际的组件,比如具体类或者对象。软件系统架构不仅指定了软件系统的组织结构和拓扑结构,而且表示了系统需求和构成组件之间的对应关系,包括设计决策的基本方法和基本原理。

软件体系结构的设计通常考虑到设计金字塔中的两个层次——数据设计和体系结构设计。数据设计体现传统系统中体系结构的数据构件和面向对象系统中类的定义（封装了属性和操作），体系结构设计则主要关注软件构件的结构、属性和交互作用。

4.2.软考中的 4+1 视图（重点★★★★★）

Kruchten 在 1995 年提出了一个“4+1”的视图模型。“4+1”视图模型从 5 个不同的视角来描述软件架构,每个视图只关心系统的一个侧面 5 个视图结合在一起才能反映软件架构的全部内容。Kruchten 在 1995 年提出的“4+1”视图模型与 RUP（Rational Unified Process）中的“4+1”模型本质上是同一个概念，它们的核心思想一致。如下表所示。这里我又放了 ABSD（基于架构的软件开发）中提到的 4+1 视图。这个也算一个特例。

视图模型	图例	记忆口诀
架构“4+1”的视图模型		罗凯进五常。罗凯（走进联合国）五常会议中。罗（逻辑）凯（开发）进（进程）五（物理）常（场景）。

RUP 和 UML 的“4+1”视图模型		落实进步用。这个东西是落实进步用的。落（逻辑）实（实现）进（进程）部（部署）用（用例）。
ABSD 中的 4+1 视图		逻实进配。

4.3.软件架构设计与生命周期（非重点☆☆☆☆☆）

这里有六大阶段，需求分析阶段，设计阶段，实现阶段，构件组装阶段，部署阶段和后开发阶段，书上写得晦涩难懂，看凯恩整理的了解一下就行！

阶段	主要研究内容	具体说明
需求分析阶段	从需求模型转换到架构模型及保证可追踪性	研究如何从如用例图的需求模型转换为如类图的架构模型，将架构概念贯穿生命周期，方便各阶段参与者交流并维护可追踪性
设计阶段	架构模型描述相关内容	基本概念涉及构件、连接器；使用架构描述语言（ADL）；采用多视图表示方法，如经典 4 + 1 视图模型、Hofmeister 模型（概念视图、模块视图、执行视图和代码视图）、Views and Beyond 模型（模块视图、构件和连接子视图、分配视图）从不同角度描述架构
实现阶段	缩小架构设计与具体实现的差距	在架构模型引入编程概念、运用模型转换技术、基于中间件平台进行构件组装等，还研究基于架构的测试、配置管理等技术

构件组装阶段	依据架构模型进行构件组装及处理相关问题	根据架构模型中的连接器组装构件实现系统，研究支持连接器实现以及检测和消除架构失配（构件与实际环境假设不符）问题。 （1）由构件引起的失配，包括由于系统对构件基础设施、构件控制模型和构件数据模型的假设存在冲突引起的失配。 （2）由连接子引起的失配，包括由于系统对构件交互协议、连接子数据模型的假设存在冲突引起的失配。 （3）由于系统成分对全局体系结构的假设存在冲突引起的失配等。 要解决失配问题，首先需要能够检测出失配问题，并在此基础上通过适当的手段消除检测出的失配问题。
部署阶段	基于架构模型描述软硬件部署模型及分析质量属性	研究怎样基于架构模型来描述软硬件部署模型，并对部署方案的质量属性展开分析
后开发阶段	动态软件架构及遗留系统架构相关研究	包括动态软件架构（运行时体系结构可变）的研究，以及从遗留系统中恢复 / 重建架构的方法研究

有心的同学可能发现实现阶段有构件组装，这和下一步的构件组装阶段有什么关系。这里书上说的很明确，主要是构件粒度不一样。实现阶段是封装代码细节，组装成大粒度的构件，方便构件组装阶段来构建系统。

4.4.基于架构的软件开发方法 ABSD（次重点★★★☆☆）

这个方法很少人用，主流还是敏捷。但是出于某些原因，此方法发明人和软考有千丝万缕的联系（系分教材作者），所以凯恩建议掌握，划线的部分都要背，选择题会考。

4.4.1.基本概念（超级重点★★★★★）

这一大段概念架构考生几乎全部都要背诵（全文），下划线部分必考选择题，选择常考 4-5 分。

基于体系结构的软件设计（Architecture-Based Software Design，ABSD） 方法。 ABSD 方法是由体系结构驱动的，即指由构成体系结构的商业、质量和功能需求的组合驱动的。

ABSD 方法有 3 个基础。第 1 个基础是功能的分解。在功能分解中，ABSD 方法使用已有的

基于模块的内聚和耦合技术。第 2 个基础是通过选择体系结构风格来实现质量和商业需求。第 3 个基础是软件模板的使用，软件模板利用了一些软件系统的结构。

ABSD 方法是递归的，且迭代的每一个步骤都是清晰定义的，有助于降低体系结构设计的随意性。

ABSD 方法是一个自顶向下，递归细化的方法，软件系统的体系结构通过该方法得到细化，直到能产生软件构件和类。在最顶层，系统被分解为若干概念子系统和一个或若干个软件模板。在第 2 层，概念子系统又被分解成概念构件和一个或若干个附加软件模板。

视图视角：考虑体系结构时，要从不同的视角来观察对架构的描述，展示功能组织的静态视角能判断质量特性，展示并发行为的动态视角能判断系统行为特性。选择的特定视角或视图（如逻辑视图、进程视图、实现视图和配置视图。这个又是一个新的说法，书本没有展开解释只能当做特例记忆）可以全方位地考虑体系结构设计。

这里可以发现 ABSD 用的视图非常类似 RUP / UML 4+1 视图（逻辑视图、进程视图、开发视图和部署视图，然后以用例为中心）。这个书本没有给出解释，当作特例进行记忆。

使用逻辑视图来记录设计元素的功能和概念接口,设计元素的功能定义了它本身在系统中的角色，这些角色包括功能、性能等。

用例和质量场景：用例用来捕获功能需求。在使用用例捕获功能需求的同时，人们通过定义特定场景来捕获质量需求，并称这些场景为质量场景。我们质量场景捕获变更、性能、可靠性和交互性，分别称之为变更场景、性能场景、可靠性场景和交互性场景。

质量场景必须包括预期的和非预期的场景。例如，一个预期的性能场景是估计每年用户数量增加 10%的影响，一个非预期的场景是估计每年用户数量增加 100%的影响。非预期场景在决定设计的边界条件时很有用。

4.4.2.基于体系结构的开发模型 ABSD（重点★★★★★）

ABSD 模型把整个基于体系结构的软件过程划分为体系结构需求、设计、文档化、复审、实现和演化 6 个子过程。重点关注体系结构需求和体系结构文档化。具体描述如下表所示。

阶段	主要内容	具体说明
体系结构需求	需求相关概念及来源、需求过程、需求评审	需求指用户对目标软件在功能等方面的期望，受技术环境和设计师经验影响。来源为系统质量目标、商业目标及开发人员商业目标。需求过程获取用户需求并标识构件，可利用需求库提高效率。需求评审由不同代表组成小组审查需求及相关构件
体系结构设计	设计过程	体系结构需求激发和调整设计决策，通过不同视图表达质量目标信息，设计是迭代过程
体系结构文档化	文档作用、输出结果及要求	文档是系统演化各阶段人员通信媒介和设计验证等基础。主要输出体系结构规格说明和测试质量设计说明书。文档要完整高质量，从使用者角度编写，分发所有相关开发人员并保证最新。（有一年真题说不用最新，这里还是以书本概念为准）
体系结构复审	复审安排、方式及目的	主版本软件体系结构分析后安排外部人员（用户代表和领域专家）参加复审。常搭建最小化可运行系统评估测试架构，目的是标识潜在风险，发现设计缺陷和错误
体系结构实现	实现过程	以复审后的文档化体系结构说明书为基础，按设计结构，利用组装支持工具组装构件实现体，完成系统连接合成，测试包括构件功能及整体功能性能测试
体系结构的演化	演化原因及步骤	构件开发或软件运行移植时需求变动，需修改体系结构。主要步骤有需求变化归类、制订演化计划、修改 / 增加 / 删除构件、更新构件相互作用、构件组装与测试、技术评审

记忆口诀：许设温敷实验。许博士设计了一个温敷的实验。许（需求）设（设计）温（文档）敷（复审）实（实现）验（演化）。

4.5.软件架构风格（超级重点★★★★★）

以往每年几乎都会出现，23 年下半年破天荒一题没考，意味着命题组风格进行了切换，但是这一章的内容不可不准备，仍然是重点。

软件体系结构风格是描述某一特定应用领域中系统组织方式的惯用模式。体系结构风格定义一

个系统家族，即一个体系结构定义一个词汇表和一组约束。词汇表中包含一些构件和连接件类型，而这组约束指出系统是如何将这些构件和连接件组合起来的。体系结构风格反映了领域中众多系统所共有的结构和语义特性，并指导如何将各个模块和子系统有效地组织成一个完整的系统。

实际上考试常考的给定场景让你判断用哪种架构风格处理。这里凯恩把书上的内容进行了梳理，看下面这个大表。这里的概念作为架构考生几乎全部要熟练记忆，系分考生要求低一点，真题会判断就行。

一级风格	概念	二级风格	概念	应用
数据流体系结构风格	数据流体系结构没有概念上的程序计数器：指令的可执行性和执行仅基于指令输入参数的可用性来确定，因此，指令执行的顺序是不可预测的，即行为是不确定的。	批处理体系结构风格	每个处理步骤是一个单独的程序，每一步必须在前一步结束后才能开始，并且数据必须是完整的，以整体的方式传递。它的基本构件是独立的应用程序，连接件是某种类型的媒介。	经典数据处理、程序开发、Windows 下的 BAT 程序、传统的编译器
		管道-过滤器体系结构风格	把系统分解为几个处理步骤，这些步骤之间通过数据流连接，一个步骤的输出是另一个步骤的输入。每个处理步骤由一个过滤器实现，处理步骤之间的数据传输由管道负责。每个处理步骤（过滤器）都有一组输入和输出，过滤器从管道中读取输入的数据流，经过内部处理，然后产生输出数据流并写入管道中。基本构件是过滤器，连接件是数据流传输管道。可以并行。	Uunix Shell 编写的程序
调用/返回体系结构风格	调用/返回风格是指在系统中采用了调用与返回机制。是一种分治策略，将一个复杂的大系统分解为若干子系统，以便降低复杂度，并	主程序/子程序风格	主程序/子程序风格一般采用单线程控制，把问题划分为若干处理步骤，构件即为主程序和子程序。子程序通常可合成为模块。过程调用作为交互机制，即充当连接件。调用关系具有层次性，其语义逻辑表现为子程序的正确性取决于它调用的子程序的正确性（这是递归描述）。	早期结构化程序设计

	且增加可修改性。	面向对象体系结构风格	面向对象系统风格建立在数据抽象和面向对象的基础上，数据的表示方法和它们的相应操作封装在一个抽象数据类型或对象中。这种风格的构件是对象。	Java/C# 应用程序
		层次型体系结构风格	层次系统组成一个层次结构，每一层为上层提供服务，并作为下层的客户。这样的系统中构件在层上实现了虚拟机。连接件由通过层间交互协议来定义，拓扑约束包括对相邻层间交互的约束。每一层最多只影响两层，同时只要给相邻层提供相同的接口，允许每层用不同的方法实现，为软件重用提供了强大的支持。	Web 系统、数据库应用
		客户端/服务器体系结构风格	客户端/服务器体系结构风格是基于资源不对等，且为实现共享而提出的。两层 C/S 体系结构有 3 个主要组成部分：数据库服务器、客户应用程序和网络。三层 C/S 结构增加了一个应用服务器。整个应用逻辑驻留在应用服务器上。应用功能分为表示层、功能层和数据层三层。	Web 系统、数据库应用
以数据为中心的体系结构风格	以数据为中心	仓库体系结构风格	仓库是存储和维护数据的中心场所。两种不同的构件：中央数据结构说明当前数据的状态以及一组对中央数据进行操作的独立构件，仓库与独立构件间的相互作用在系统中会有大的变化。连接件即为仓库与独立构件之间的交互。	现代集成开发环境（IDE）、企业级数据管理系统
		黑板体系结构风格	黑板系统是一种问题求解模型。它将问题的解空间组织成一个或多个应用相关的分级结构。分级结构的每一层信息由一个唯一的词汇来描述，它代表了问题的部分解。	信号处理领域，如语音识别和模式识别。另一应用是松耦合代理数据共享存取。

虚拟机体系结构风格	虚拟机体系结构风格的基本思想是人为构建一个运行环境，在这个环境之上，可以解析与运行自定义的一些语言，这样来增加架构的灵活性。	解释器体系结构风格	一个解释器包括（1）完成解释工作的解释引擎（2）一个包含将被解释的代码的存储区（3）一个记录解释引擎当前工作状态的数据结构（4）一个记录源代码被解释执行进度的数据结构。解释器缺点是执行效率较低。	在线游戏系统支持用户自行定义游戏对象的属性、行为和对象之间的交互关系。 业务灵活组合那就是选择解释器风格。
		规则系统体系结构风格	基于规则的系统包括规则集、规则解释器、规则/数据选择器及工作内存。	专家系统/决策支持系统，是指采用计算机技术和决策理论建立的、能够为决策者提供决策信息和决策方法的信息系统。
独立构件体系结构风格	独立构件风格主要强调系统中的每个构件都是相对独立的个体，它们之间不直接通信，以 降低耦合度，提升灵活性。	进程通信体系结构风格	在进程通信结构体系结构风格中，构件是独立的过程，连接件是消息传递。这种风格的特点是构件通常是命名过程，消息传递的方式可以是点到点、异步或同步方式及远程过程调用等。	分布式系统、微服务架构
		事件系统体系结构风格	基于事件的隐式调用风格的思想是构件不直接调用一个过程，而是触发或广播一个或多个事件。系统中的其他构件中的过程在一个或多个事件中注册，当一个事件被触发，系统自动调用在这个事件中注册的所有过程。	在编程环境中用于集成各种工具，在数据库管理系统中确保数据的一致性约束，在用户界面系统中管理数据，以及在编辑器中支持语法检查。
C2	C2 体系结构风格可以概括为通过连接件绑定在一起按照一组规则运作的并行构件网络。	（1）构件和连接件都有顶部和底部。 （2）构件的顶部连接到连接件的底部，构件的底部连接到连接件的顶部，构件之间不允许直接连接。 （3）连接件可以连接任意数量的构件和其他连接件。 （4）当两个连接件直接相连时，必须一个的底部连接到另一个的顶部。		
闭环控制架构 - 过程控制	软件与硬件之间可以粗略地表示为一 个反馈循环，这个反馈循环通过接受一定的输入，确定一系列的输出，最终使环	空调控温，定速巡航。		

	境达到一个新的状态。	

批处理和管道过滤器的区别：批处理前后构件不一定有关联，并且是作为整体传递，即必须前一个行完才能执行下一个。管道-过滤器是前一个输出作为后一个输入，前面执行完后可以开始下一步的执行。

4.6.软件架构复用（重点★★★★★）

软件架构复用（Software Architecture Reuse）是指在多个软件系统或项目中重复使用已有的架构设计、组件、模式、技术或构件，以提高开发效率、降低成本、提高质量和可维护性。它强调将已有的架构经验和设计方案有效地应用到新的项目中，避免从零开始重复开发。

这一块老教材没有，是架构新教材新增内容。重点关注划线的概念，会出选择和论文。

4.6.1.软件产品线（次重点★★★☆☆）

软件产品线是指一组软件密集型系统，它们共享一个公共的、可管理的特性集，满足某个特定市场或任务的具体需要，是以规定的方式用公共的核心资产集成开发出来的。即围绕核心资产库进行管理、复用、集成新的系统。核心资产库包括软件架构及其可剪裁的元素，还包括设计方案及其文档、用户手册、项目管理的历史记录（如预算和进度）、软件测试计划和测试用例。

软件复用是系统化的软件开发过程：开发一组基本的软件构造模块，以覆盖不同的需求/体系结构之间的相似性，从而提高系统开发的效率、质量和性能。

软件架构复用的类型包括机会复用和系统复用。机会复用是指开发过程中，只要发现有可复用的资产，就对其进行复用。系统复用是指在开发之前，就要进行规划，以决定哪些需要复用。

4.6.2.软件架构复用的基本过程（次重点★★★☆☆）

复用的基本过程主要包括 3 个阶段：首先构造/获取可复用的软件资产，其次管理这些资产，

最后针对特定的需求，从这些资产中选择可复用的部分，以开发满足需求的应用系统。每个阶段做什么要有大致的概念。

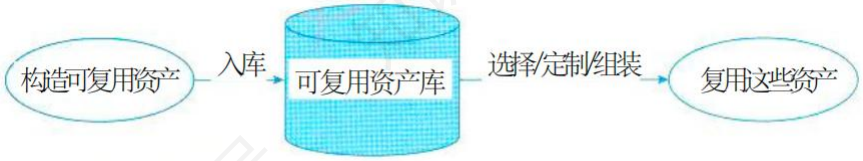


图7-18 软件架构复用的基本过程

阶段	主要内容	具体说明
获取可复用的软件资产	明确可复用软件资产的特性	这些资产须具备可靠性、广泛适用性、易理解性以及易修改性
管理可复用资产	强调构件库对软件复用的支持及功能	构件库是支持软件复用的关键设施，其主要功能涵盖构件存储、管理、检索，库的浏览与维护等，助力使用者高效、精准地找到所需可复用构件。具体包括构件分类，即按特定方式组织大量构件；构件检索，即依据给定查询需求快速准确找到相关构件
使用可复用资产	阐述可复用资产的使用流程	在最后阶段获取可复用资产，通过修改、扩展、配置等方式定制这些资产，最终将其组装与集成，形成最终系统

4.7.特定领域软件体系结构 DSSA（次重点★★★☆☆）

"特定领域软件体系结构"（Domain-Specific Software Architecture，简称 DSSA）是指专门为解决某一特定领域的问题而设计的软件体系结构。DSSA 通过对特定领域共性需求与变化点的系统抽象，将架构经验上升为可复用的领域架构资产，从而在保证系统质量属性的同时，显著效率、可扩展性、可维护性和适应性。DSSA 中领域可以分为垂直域和水平域。

类别	定义	适用范围
垂直域	定义特定系统族，包含族内多个系统，成果是该领域中可作为系统可行解决方案的通用软件体系结构	只能应用于成熟、稳定的领域

水平域	定义多个系统和多个系统族中功能区域的共有部分，在子系统级涵盖多个系统族的特定部分功能	不局限于特定成熟领域，只要存在多个系统或系统族有部分功能共性即可

DSSA 包含（三种）基本活动，（四种）参与人员，5 个阶段。下表梳理的内容，特别是基本活动和参与人员需要熟悉。

大类	定义	描述
基本活动	领域分析	目标是获得领域模型，描述领域中系统的共同需求。
	领域设计	目标是获得 DSSA
	领域实现	目标是依据领域模型和 DSSA 开发和组织可重用信息。
参与人员	领域专家	包括该领域系统的资深用户、从事需求分析、设计、实现及项目管理的软件工程师
	领域分析人员	由具备知识工程背景的资深系统分析员担任
	领域设计人员	由资深软件设计人员担任
	领域实现人员	由资深程序设计人员担任
建立过程	定义领域范围	重点确定感兴趣领域的范围及过程终止时间
	定义领域特定的元素	目标是编制领域词典和术语同义词词典，在前一阶段的高层模型基础上，增加更多细节，识别应用间的共性和差异性
	定义领域特定的设计和实现需求约束	目标是描述解空间中的差异特性，需识别约束，记录约束对设计实现决策的影响，讨论处理这些问题的方法
	定义领域模型和体系结构	目标是产生一般体系结构模型
	产生、搜集可重用的产品单元	目标是为领域特定软件架构增加可重用构件，使之可用于在该领域产生新应用

4.8.系统质量属性与架构评估（重点★★★★★）

这里两类质量属性注意区分，选择题都有考察，一类叫作软件系统质量属性，描述系统的，另一类叫作架构质量属性，你要注意区分。软件质量属性是架构新教材第二版新增内容，你要注意区分。

4.8.1.软件系统质量属性（次重点★★★☆☆）

软件系统质量属性是指影响软件系统性能和可靠性的因素,用来衡量和评估软件系统在不同方面的表现。质量属性通常是在设计和开发过程中考虑的非功能性要求(Non-functional Requirements),例如系统的响应时间、可扩展性、可维护性等。基于软件系统的生命周期，可以将软件系统的质量属性分为开发期质量属性和运行期质量属性 2 个部分。了解即可，不重要。

开发期质量属性（6 个）	易理解性、可扩展性、可重用性、可测试性、可维护性、可移植性
运行期质量属性（7 个）	性能、安全性、可伸缩性、互操作性、可靠性、可用性、鲁棒性。

4.8.2.面向架构评估的质量属性（超级重点★★★★★）

架构设计会直接影响到软件系统如何满足这些质量属性，因此，在进行架构评估时，需要重点关注这些属性如何通过架构设计进行优化和保证。和软件系统质量属性的不同点在于，面向架构评估的质量属性的特点是，它强调如何通过架构设计实现和优化质量属性。

4.8.2.1.九大必考质量属性（超级重点★★★★★）

为了评价一个软件系统，特别是软件系统的架构，需要进行架构评估。在架构评估过程中，评估人员所关注的是系统的质量属性。评估方法所普遍关注的质量属性有以下 9 种。也是案例选择常考的、让你辨析的内容。

属性	描述	对策
性能	指系统的响应能力,处理事务所需时间或单位时间内处理事务数量。	优先级队列、增加计算资源、减少计算开销、引入并发机制、采用资源调度

可靠性	指软件在应用错误、意外错误使用情况下维持功能特性的基本能力，通过平均失效时间来衡量。	冗余设计（硬件冗余、数据冗余）、错误检测与恢复机制（如奇偶校验、循环冗余校验）、异常处理（捕获并处理程序运行中的异常）、备份与恢复策略（定期备份数据，在故障时可恢复）、负载均衡（避免单个组件过载）
可用性	指系统正常运行的时间比例，表现为两次故障间时长或故障时恢复正常速度。	心跳机制（检测系统组件的运行状态）、冗余系统（如双机热备、集群）、快速恢复机制（使用备用设备或系统）、内置监控器监控与预警（实时监控系统状态，提前发现潜在问题）、故障转移（自动将工作负载转移到备用组件）。心跳是检测系统组件的在线状态的，内置监视器更多的是预警
安全性	指向合法用户提供服务同时阻止非授权访问或拒绝服务的能力，包括机密性、完整性、不可否认性、可控性等。	用户认证（如用户名密码认证、多因素认证）、授权管理（基于角色的访问控制）、数据加密（对敏感数据进行加密存储和传输）、防火墙（防止外部非法访问）、入侵检测与防范系统（检测并阻止网络攻击）、审计与日志记录（记录系统操作，便于事后审计）
可修改性	指快速、高性价比地变更系统的能力，包括可维护性、可扩展性、结构重组、可移植性。	模块化设计（降低模块间耦合度）、接口标准化（便于替换和扩展组件）、抽象与封装（隐藏实现细节，提高可维护性）、配置管理（使用配置文件管理系统参数）、版本控制（管理代码和文档的变更）
互操作性	软件不是独立存在的，需要与其他系统或环境交互。为了支持互操作性，软件架构必须为外部功能和数据结构提供精心设计的入口。不同编程语言编写的软件系统之间的交互作用体现了互操作性问题。	使用标准协议和接口（如 RESTful API、SOAP）、数据格式转换、中间件、服务注册与发现、跨语言开发工具
可移植性	软件或硬件适应不同运行环境的能力。通过抽象层（如虚拟机）、环境无关代码设计、减少平台依赖，使系统在迁移至新操作系统（如从 Windows 到 Linux）、硬件平台或云环境时，无需重大修改即可正常运行，如容器化应用跨云部署。	遵循相关标准协议并依赖标准 API，避免特定平台专有 API 以确保兼容性和减少平台依赖；抽象底层差异，通过封装底层代码和借助中间件或虚拟机隔离软件与底层的差异；进行兼容性测试，利用多种硬件平台、操作系统、浏览器等进行多平台测试。

可测试性	指软件能够被有效测试的难易程度，即通过设计和开发手段，使软件易于发现缺陷、验证功能正确性和性能指标达标性的特性。通常体现在测试用例的设计、执行效率，以及缺陷定位的便捷性上。	设计分层与模块化：将系统拆分为独立模块，减少模块间耦合度，便于针对单个模块编写测试用例，降低测试复杂度；提供测试接口：在代码中预留专用的测试接口，允许测试人员直接访问内部状态或调用特定功能，辅助进行白盒测试；
易用性	指系统让用户“好学、好找、好操作、少犯错”的程度，包括学习成本、操作效率、可理解性与可访问性。	统一交互与视觉规范；信息架构清晰；可发现性与即时反馈；降低认知负荷；快捷操作；多语言与本地化；响应式与无障碍；新手引导与内嵌帮助；可配置主题/字号与个性化偏好保存等。

Q1:可靠性和可用性的区别是否需要辨析。实际上架构第二版上这两个不一样的概念，但是真题实际上也没出现过两者的辨析。同时根据国际标准 ISO/IEC 25010 的解读，可用性实际上是可靠性的一个子集。其中可用性的定义包含了故障恢复速度的含义，可靠性的度量指标平均失效时间实际上也是这个意思（失效时间短也是恢复速度快）。近十年之内两者都没出现过辨析，以真题为例：

网络失效后，系统需要在 2 分钟内发现错误并启用备用系统；

主站点断电后，需要在 3 秒内将访问请求重定向到备用站点；

网络失效后，系统需要在 2 分钟内发现并启用备用网络系统；

系统主站点断电后，需要在 3 秒内将请求重定向到备用站点；

能够连续运行的时间不小于 240 小时，意外退出后能够在 10 秒之内自动重启。

网络失效后，系统需要在 10 秒内发现错误并启用备用系统；

系统主站点断电后，必须在 3 秒内将访问请求重定向到备用站点；

系统主站点断电后，应在 5 秒内将请求重定向到备用站点；

当系统发生网络失效后，需要在 15 秒内发现错误并启用备用网络；

系统主站点断电后，应在 3s 内将请求重定向到备用站点;

系统宕机后，需要在 15s 内发现错误并启用备用系统;

平台支持分布式部署，当主站点断电后，应在 20 秒内将请求重定向到备用站点；

平台主站点宕机后，需要在 15 秒内发现错误并启用备用系统；

这些都是归结到可用性上。所以这个不纠结。写可用性可靠性都没问题。默认可用性。

4.8.3.质量属性场景描述（超级重点★★★★★）

这一章节架构同学必学必会，案例题就捶你概念，让你默写。为了精确描述软件系统的质量属性，通常采用质量属性场景 (Quality Attribute Scenario) 作为描述质量属性的手段。质量属性场景是一个具体的质量属性需求，是利益相关者与系统的交互的简短陈述。

目的	引入质量属性场景的目的是精确描述软件系统的质量属性。
概念	质量属性场景是对利益相关者与系统交互的简短陈述，描述了一个具体的质量属性需求
6 个组成部分	刺激源、刺激、环境、制品、响应、响应度量
6 类质量属性	可用性、可修改性、性能、可测试性、易用性和安全性

组成部分	定义
刺激源	生成刺激的实体
刺激	到达系统时需考虑的条件
环境	刺激发生的条件
制品	被激励的对象
响应	激励到达后采取的行动
响应度量	对响应进行度量的方式

记忆口诀：原词换纸想读。原来的歌词换个纸我就想读了。原（刺激源）词（刺激）换（环境）纸（制品）想（响应）读（度量）。

上面的定义一般不会直接考察，会结合具体的场景让你写出对应的组成部分是什么，凯恩整理如下表所示，考架构的同学必学必会，最低要求是看熟悉。可用性、可修改性、性能和安全性是必

须要掌握的。

质量属性	刺激源	刺激	环境	制品	响应	响应度量
可用性	系统内部、外部	疏忽、错误、崩溃、时间	正常操作、降级模式	系统处理器、通信信道、持久存储器、进程	检测事件，记录并通知相关方，禁止错误或故障源	可用时间/可用性；允许不可用时长；故障修复时间（MTTR）；降级可运行时长
可修改性	最终用户、开发人员、系统管理员	希望增加、删除、修改、改变功能、质量属性、容量等	系统设计时、编译时、构建时、运行时	系统用户界面、平台、环境或与目标系统交互的系统	查找需修改位置，修改且不影响其他功能，测试并部署修改	修改成本/工时；影响范围（受影响元素数）；对其他功能/质量属性影响程度
性能	用户请求，其他系统触发等	定期事件到达、随机事件到达、偶然事件到达	正常模式、超载模式	系统	处理刺激、改变服务级别	响应/等待时间；吞吐量；期限/截止时间 SLA 达成率/超时率；丢失率
可测试性	开发人员、增量开发人员、系统验证人员、客户验收测试人员、系统用户	已完成的分析、架构、设计、类和子系统集成；所交付的系统	设计时、开发时、编译时、部署时	设计、代码段、完整地应用	提供对状态值的访问，提供所计算的值，准备测试环境	覆盖率；缺陷暴露率；测试执行时间；环境准备时间；依赖链长度
易用性	最终用户	想要学习系统特性、有效使用系统、使错误的影响最低、适配系统、对系统满意	系统运行时或配置时	系统	提供支持学习、有效使用、降低错误影响、适配系统、使客户满意的响应	任务完成时间；错误数；成功率；满意度

安全性	正确识别、非正确识别身份未知的来自内部/外部的个人或系统；经过了授权/未授权它访问了有限的资源/大量资源	试图显示数据，改变/删除数据，访问系统服务，降低系统服务的可用性	在线或离线、联网或断网、连接有防火墙或者直接连到了网络	系统服务、系统中的数据	对用户身份进行认证；隐藏用户的身份；阻止/允许访问；授予/收回许可；记录访问；存储数据；识别高需求；通知并限制服务可用性	绕过成本（时间/资源）；攻击检测率；溯源成功率；DDoS 下可用性；恢复时间/影响范围

4.8.4.系统架构评估（超级重点★★★★★）

系统架构评估是在对架构分析、评估的基础上，对架构策略的选取进行决策。系统架构评估的方法通常可以分为 3 类：基于调查问卷或检查表的方式、基于场景的方式和基于度量的方式。软考只考基于场景的方式和基于度量的方式。

4.8.4.1.系统架构评估中的重要概念（超级重点★★★★★）

这里特别是敏感点和权衡点的辨析是系分架构选择最爱出的方向，看凯恩举的例子就能明白。案例中呢有时候会出现风险点和非风险点（基本不考），也需要注意。这些都是语文题，注意这里的句式。敏感点，风险点，权衡点，说多了可能觉得咬文嚼字，最简单的判定是根据句式，如下表所示。

概念	定义	举例
敏感点	一个或多个构件（和/或构件之间的关系）的特性	对查询请求处理时间的要求将影响系统的数据传输协议和处理过程的设计

权衡点	影响多个质量属性的特性，是多个敏感点的综合	还是上述在线文件存储系统，加密级别的选择就是权衡点。高加密级别（如 AES - 256）能大幅提升安全性，但会增加数据加密和解密的计算量，降低系统性能；低加密级别（如 DES）安全性稍弱，但对性能影响较小。它影响了安全性和性能多个质量属性。
风险承担者 / 利益相关人	系统架构涉及到的利益相关者，会试图影响架构决策以实现自己目标	企业 ERP 系统中的财务部门、销售部门
场景	从风险承担者的角度对于系统的交互的简短描述，架构评估中一般采用刺激、环境和响应三方面来描述	客户在公共 Wi - Fi 环境下尝试登录银行网上银行系统，系统需 10 秒内完成身份验证并显示账户信息
风险点	可能导致系统架构出现问题或无法满足质量属性要求的特性、决策或情况。这些因素一旦发生，会给系统带来负面的影响，如降低性能、损害安全性等	如果“养护报告生成”业务逻辑的描述尚未达成共识，可能导致部分业务功能模块规则的矛盾，影响系统的可修改性（如果...，可能...，影响）
非风险点	不会对系统架构的质量属性造成负面影响，或者对系统架构的影响在可接受范围内的特性、决策或情况	选择一款成熟且持续更新的消息队列中间件（如 RabbitMQ ），其性能、稳定性都经过市场验证，在当前系统架构下使用它不会给系统带来明显风险

4.8.5.系统架构评估方法（重点★★★★★）

系统架构评估方法是用于评价系统架构优劣、合理性及是否满足需求等的一系列技术和手段。软考里只有 SAAM、ATAM、CBAM 和其他。选择主要考概念。

4.8.5.1.基于场景的架构分析方法 SAAM 方法（次重点★★★☆☆）

SAAM 的目标是通过场景验证架构假设和原则，评估架构固有风险，比较不同架构方案。SAAM 采用场景技术进行评估，将质量属性具体化为场景描述。SAAM 关注的主要质量属性是可修改性，但也可扩展到其他属性。SAAM 协调不同利益相关者的关注点，达成架构共识。SAAM 针对最终架构而非详细设计进行评估。SAAM 的评估过程包括：场景开发、架构描述、单场景评估、场景

交互评估和总体评估。

4.8.5.2.架构权衡分析方法 ATAM 方法（重点★★★★★）

ATAM 在 SAAM 的基础上发展起来的，主要针对性能、可用性、安全性和可修改性，在系统开发之前，对这些质量属性进行评价和折中。

ATAM 活动过程。ATAM 包括 4 个主要阶段：场景和需求收集、架构视图和场景实现、属性模型构造和分析、折中。整个过程是迭代式的。真题考到过 ATAM 关注的是什么。这里的定论是需求，因为你后面的架构视图还是质量属性评估，无一例外都要解决具体的用户需求问题，这是出发点也是终点。

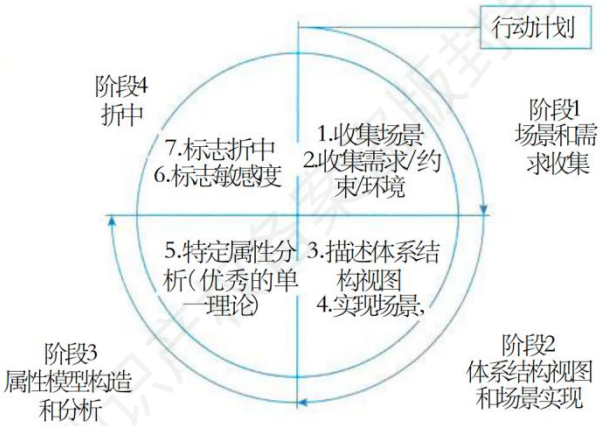


图 8-2 ATAM 分析评估过程

上面的图例是 ATAM 的分析过程，按照大阶段来说分为四步。

阶段	名称	内容	示例（社交软件）
阶段 1	场景和需求收集	收集场景以及约束等需求信息	开发社交软件时，收集用户对消息收发及时性、隐私保护等场景需求
阶段 2	体系结构视图和场景实现	描述体系结构图，实现场景	画出社交软件模块架构图，规划消息模块高并发下消息收发场景的实现
阶段 3	属性模型构造和分析	基于优秀单一理论进行特定属性分析，构造属性模型并分析	构建性能模型分析社交软件大量用户同时在线时的响应时间

阶段 4	折中	标志折中情况，识别敏感度	社交软件中加强隐私保护影响消息收发速度为折中，加密算法等为敏感点

ATAM 方法的架构评估实践分为四个阶段：演示、调查和分析、测试和报告。有心的读者会发现这里的四个阶段和上面的四个阶段不一样。凯恩的理解是上图的行为可以看作的是调查分析阶段的展开。即一个完整的 ATAM 实践活动是演示、调查和分析（场景和需求收集、架构视图和场景实现、属性模型构造和分析、权衡）、测试和报告，如下表所示。

阶段	主要内容
演示阶段	评估团队介绍 ATAM 过程、业务目标和要评估的体系结构
调查和分析阶段	深入探讨架构方法，生成质量属性效用树，分析架构方法
测试阶段	通过头脑风暴和优先场景理解质量属性要求，分析架构方法
报告阶段	ATAM 团队向利益相关者呈现评估结果、效用树、场景、分析问题、风险和非风险以及架构方法的发现，为架构改进提供指导

ATAM 方法采用效用树 （Utility tree）这一工具来对质量属性进行分类和优先级排序。效用树的结构包括：树根—质量属性—属性分类—质量属性场景（叶子节点）。需要记忆。

需要注意的是，ATAM 主要关注 4 类质量属性：性能、安全性、可修改性和可用性，这是因为这 4 个质量属性是利益相关者最为关心的。

得到初始的效用树后，需要修剪这棵树，保留重要场景（通常不超过 50 个），再对场景按重要性给定优先级（用 H/M/L 的形式），再按场景实现的难易度来确定优先级（用 H/M/L 的形式），这样对所选定的每个场景就有一个优先级对（重要度、难易度），如（H、L）表示该场景重要且易实现。做真题的时候你会知道，是通过填空形式来进行考察的。

4.8.5.3.CBAM 方法（次重点★★★☆☆）

CBAM 的核心思想是：架构策略会影响系统的质量属性。质量属性的变化会为系统的相关方

带来一定收益（效用）。CBAM 协助相关方根据投资回报率（ROI）来选择最优架构策略。

4.8.5.4.其他方法（非重点☆☆☆☆☆）

了解即可不做要求，假如出论文题目放弃，不好写。

方法名称	主要特点
SAEM 方法	兼顾软件架构的产品属性与过程属性，从内外部质量属性评估，构建通用评估框架
SAABNet 方法	借助专家知识，利用贝叶斯信念网络实现定性评估
SACMM 方法	聚焦架构修改，以图内核定义准则度量架构变化
SASAM 方法	通过对比预期与实际架构进行静态评估
ALRRA 方法	综合复杂度因素与失效后果严重性评估可靠性风险
AHP 方法	采用多准则决策流程解决评估冲突并排名
COSMIC+UML 方法	基于面向对象系统，关联 COSMIC 与 UML 组件图度量可维护性

5.软件测试（次重点★★★☆☆）

本章主要在选择题中进行考察，案例几乎不考察。测试书本上提到了二十多种测试方法，都要掌握完全不可能，我们要掌握的就是下表凯恩列出的内容，标记颜色的要重点看。

5.1.测试分类方法（重点★★★★★）

测试也是水很深的内容，从不同的角度可以提出不同的测试概念，如下表所示，架构的内容没把功能测试和性能测试展开，系分讲的很清楚，见后面两小节的补充。

分类	类别	详情
按被测程序是否可见	黑盒测试	又叫功能测试等，着重软件功能性需求，将程序看作黑盒，不考虑内部结构与特性，从用户角度基于输入输出关系测试。能发现功能、数据结构等错误，用例设计方法有等价类划分法等多种。在软件测试各阶段尤其系统和确认测试中作用重要

分类	白盒测试	又称结构测试等，需全面了解程序内部逻辑结构，对所有逻辑路径测试。覆盖标准有逻辑覆盖、循环覆盖和基本路径测试等这个后面提到的排序一定要看。遵循特定原则，“三步法” 结合多种测试方式，主要用于军工、航天等对可靠性要求高的软件领域
	灰盒测试	介于白盒与黑盒测试之间，多用于集成测试，既关注输出输入正确性，也关注程序内部情况。通过表征性现象等判断内部运行状态，由特定方法和工具组成，结合了白盒和黑盒测试要素
按是否需要执行被测程序分类	静态测试	不运行被测程序，通过分析源程序语法、结构、过程、接口等检查正确性。包括代码检查、静态结构分析、代码质量度量等，可人工或借助软件工具自动进行，能发现如不匹配参数等问题，具有发现缺陷早等优点，但耗时长、对技术能力要求高
	动态测试	通过运行被测程序，检查运行结果与预期结果差异，并分析运行效率、正确性和健壮性等性能。由构造测试用例、执行程序、分析程序输出结果三部分组成，确保软件安装后正常运行
按测试对象划分	功能测试	根据产品特性、操作描述和用户场景测试产品功能，确保程序按预期方式运行。通过需求分析、编写测试案例、准备数据、执行测试、对比结果并报告等 6 个步骤完成
	性能测试	测试软件在不同情况下的响应时间，检查是否满足性能需求。包括负载测试、压力测试、并发测试、容量测试、可靠性测试等，可通过自动化测试工具模拟多种负载条件进行
	安全测试	验证应用程序安全等级，识别潜在安全性缺陷，分应用程序本身安全性和数据安全两个层面。对安全性要求高的软件需专门测试，主要目的是查找安全隐患，检查防范能力，包括渗透测试等
	兼容性测试	测试软件之间、软件与硬件之间能否协调工作，避免相互影响。核心内容包括软件在不同操作系统平台及版本、自身前后兼容、与其他软件兼容以及数据兼容性等
	界面测试	主要测试用户界面功能模块布局、整体风格、控件位置、操作便捷性、导航、页面元素可用性、文字正确性等，检查界面是否美观
	易用性测试	又称用户体验测试，是交互的适应性、功能性和有效性的集中体现。通过裁剪出易用性检查清单、项目组成员达成一致、按清单排查并管理、评估并记录结果等流程进行
	稳定性测试	通过长时间运行软件，观察软件和系统在多次测试、开启关闭、业务切换、长时间开启不操作以及日常用户操作等情况下是否正常运行，检查内存占用等方面是否出现问题
按测试阶段划分	单元测试	对软件中最小可测试单元检查验证，单元含义因编程语言或软件类型而异。以白盒技术为主，黑盒技术为辅，一般由开发人员在组长监督下完成，使用数据多为非真实数据。测试阶段在编码前后，测试内容包括模块接口、局部数据
	集成测试	将软件集成起来后进行测试，又叫子系统测试等，主要针对软件高层设计，以模块和子系统为单位。包含模块内、子系统内、系统集成等层次。测试阶段在单元测试之后，测试内容包括模块间数据传输等方面
	系统测试	对整个系统进行测试，将硬件、软件、操作人员看作整体，检验是否符合系统说明书。可发现系统分析和设计错误，包括恢复测试、安全测试、压力测试等。测试阶段在集成测试之后，测试方法为黑盒测试，测试内容涵盖功能、性能

特别是白盒测试部分需要掌握白盒测试法的覆盖标准需要重点记忆，如下表所示。语句覆盖最弱，路径覆盖最强需要记忆。

具体覆盖标准	标准描述	发现错误能力

语句覆盖	每条语句至少执行一次	1（最弱）
判定覆盖	每个判定的每个分支至少执行一次	2.5
条件覆盖	每个判定的每个条件应取到各种可能的值，但是不考虑组合。	2.5
判定条件覆盖	同时满足判定覆盖和条件覆盖	4
条件组合覆盖	每个判定中各条件的每一种组合至少出现一次	5
路径覆盖	每一个判断的所有可能结果都出现过、每一个判断中所有条件的所有可能结果都出现过、每一个进入点及结束点都执行过、判断中每一个条件都可以独立影响判断的结果	6（最强）

记忆口诀：路条组判｜条条判语。（从高到底）。路（路径）条组（条件组合）判｜条（判定条件）条（条件）判（判定）语。

这里其实一定要举个例子才说的清楚。比如有下面这一串代码：

```
if (A && B) {
    执行优惠;
}
```

语句覆盖最弱，它只要求每条语句至少执行一次。比如只测 A=true，B=true，优惠语句执行了，就算满足语句覆盖。

判定覆盖看的是整个判断结果。也就是说，A && B 这个判断，结果为 true 的情况要测到，结果为 false 的情况也要测到。

条件覆盖看的是判断里面的每个小条件。比如 A 要取过 true 和 false，B 也要取过 true 和 false。但条件覆盖有一个典型问题：它不一定能保证整个判断结果 true 和 false 都出现过。比如 A=true，B=false 和 A=false，B=true，A、B 都取过真假值，但 A && B 的整体结果都是 false，所以 true 分支没有被覆盖。

判定条件覆盖就是同时满足判定覆盖和条件覆盖。它既要求整个判断结果有 true、有 false，也要求每个小条件都取过 true 和 false。比如：

```
用例 1：A = true，  B = true    结果 true
```

用例 2：A = false，B = false　结果 false

A 有 true、有 false，B 有 true、有 false，整体判断也有 true、有 false，所以满足判定条件覆盖。

条件组合覆盖更严格，它要求所有条件的真假组合都出现一次。比如 A && B 有两个条件，就要覆盖 true true、true false、false true、false false 四种组合。条件越多，组合越多，所以这种覆盖发现错误能力更强，但测试成本也更高。

前面我们都是围绕条件来讨论的，而路径覆盖关注程序从入口到出口有哪些执行路线。它要求从入口到出口的各种执行路线都测到。比如

```
if (A && B) {
    执行优惠;
}else {
    S2;
}
```

从路径角度看，其实只有两条路线：

路线 1：if 判断为 true，执行 S1

路线 2：if 判断为 false，执行 S2

所以路径覆盖可以用两个用例做到：

用例 1：A = true，　B = false　　整体结果 true

用例 2：A = false，B = false　　整体结果 false

这两个用例已经把 true 分支和 false 分支都走到了。但是条件组合覆盖要求 A、B 的四种组合都出现。

考试里通常把路径覆盖认为是发现错误能力最强的覆盖标准。上面的例子我们可以看到，但严格来说，路径覆盖和条件组合覆盖关注对象不同，不要简单理解成路径覆盖一定包含条件组合覆盖。

条件和判定一般是没法比，主要是条件覆盖没考虑组合。所以条件判定得到的逻辑结果，不一定能覆盖所有的判定分支，这个记住即可。

5.2.功能测试（次重点★★★☆☆）

功能测试又可以继续细分，如下表所示，这些内容有个印象即可，选择会考。

类别	详情
链接测试	检查链接是否指向正确资源、所链接资源是否存在、有无孤立页面，常用 Xenu Link Sleuth 等工具自动化测试
表单测试	校验用户提交信息正确性与有效性、默认值是否正确，确保表单元素标识准确、输入检查与异常处理得当、默认项合理，以及浏览器功能对表单数据无不良影响
数据校验	依据业务规则验证用户输入的正确性与合法性，确保校验功能正常运作
Cookies 测试	检测 Cookies 是否正常工作、保存时间是否符合预期、刷新页面的影响，以及 Cookies 中信息加密和统计次数功能是否正常，可借助 BrowserHistoryView 等工具
数据库测试	测试数据完整性，防止错误结果保存或数据失效，同时确保数据提取和操作指令正确，常使用 DBunit 等工具

5.3.性能测试（次重点★★★☆☆）

性能测试又可以继续细分，真题已经考过辨析，如下表所示。

类别	详情
负载测试	在系统性能指标已有一项或几项达到瓶颈的情况下，让系统继续运行一段时间，测试系统运行过程中各项性能指标；是在已经达到最大负载（压力）时进行的测试
压力测试	不断给被测系统加压（负载），测试系统的各项性能指标，直到系统的某些性能指标（如 CPU 使用率达到 100% 等）达到瓶颈为止，这个不断施加压力的过程称为压力测试
并发测试	负载测试和压力测试的结合过程，即逐渐增加负载，直到系统的瓶颈或者不能接受的性能点，通过综合分析交易执行指标和资源监控指标来确定系统并发性能的过程
容量测试	预先分析出反映软件系统应用特征的某项指标的极限值（如最大并发用户数、数据库记录数等），测试系统在其极限值状态下没有出现任何软件故障或还能保持主要功能正常运行

可靠性测试	为了满足软件的可靠性要求而进行的一系列设计、分析、测试等工作；是在软件生存周期的系统测试阶段提高软件可靠性水平的有效途径，能最有效地发现对可靠性影响大的缺陷，也是评估软件可靠性水平、验证软件产品是否达到软件可靠性要求的重要且有效的途径
强度测试	强度测试主要用于检查程序对异常情况的抵抗能力,检查系统在极限状态下运行时性能下降的幅度是否在允许的范围内。强度测试可以测试系统在超负荷情况下的表现,也可以测试系统在最差工作环境下的工作能力,或者验证系统在标准工作压力下的各种资源的最下限指标。
配置测试	配置测试旨在通过测试找到系统资源的最优分配原则，为系统调优提供依据。Web 工程中，硬件、操作系统、浏览器等客户端配置具有可变性和不稳定性，且用户配置可能因系统升级、网络更换等发生规律改变，易引发微妙且重要的错误，导致不同用户使用 Web 应用的体验和交互方式差异显著。配置测试并非验证所有客户端配置，而是选取一组典型的客户端与服务器端配置开展测试，保障用户在各类配置下的体验一致，并分离特殊配置相关错误，其通常与其他性能测试同步进行。
速度测试	速度测试包含网络连接速度测试和业务处理速度测试，其中网络连接速度受上网方式和带宽影响，需测试以避免响应时间过长导致用户流失，业务处理速度测试主要针对关键业务手工测速，通过多次测试取平均值并与测试工具指标对比分析。

真题考察过强度测试和压力测试的辨析，这里主要掌握，强度测试主要侧重于制造一个极限状态，这个极限状态包括超负荷，也包括环境。而压力测试是慢慢加压，渐增加负载，是一个过程性。

5.4.安全测试（次重点★★★☆☆）

安全测试也是额外的内容，安全测试各个类型比较好辨析，作为选择题不会很难。

测试类型	测试内容
数据加密测试	验证关键数据是否加密存储 / 传输，加密算法是否符合安全标准（如 AES-256）。
用户身份验证测试	检查无效登录、密码大小写敏感、登录次数限制、未验证直接访问等问题。
日志文件测试	确认日志记录事务处理、错误信息、IP 地址、用户名等，确保可追溯性。
Session 测试	验证会话超时自动退出、回退按钮失效、会话数据安全存储等机制。
备份与恢复测试	测试备份策略（全量 / 增量 / 差量）、恢复时间目标（RTO）和恢复点目标（RPO）。
访问控制策略测试	检查管理员权限隔离、访问控制文档完整性、权限代码逻辑严谨性。

安全漏洞测试	扫描跨站脚本（XSS）、SQL 注入、命令注入等漏洞，使用工具如 Burp Suite。
TCP 端口测试	关闭非必要端口，仅保留业务必需端口（如 80/443），防止未授权访问。
服务器端脚本漏洞检查	检测未授权脚本上传 / 修改、目录遍历、文件包含等服务器端安全隐患。
防火墙测试	验证防火墙规则是否拦截非法请求、限制流量、防御 DDoS 攻击等配置有效性。

5.5.软件测试模型 W/H/X 模型（次重点★★★☆☆）

2023 年系统架构设计师已经考到过，凯恩建议大家了解一下。主要关注每种测试的特点。这三种方法本质上讨论的都是测试如何来做。图不重要直接忽略。

模型	定义	测试和开发的关系
W 模型	在 V 模型基础上发展而来，测试与开发同步	同步
H 模型	测试活动独立，贯穿整个软件生命周期	独立
X 模型	针对单独程序片段进行编码和测试，频繁交接集成	交错

6.系统运行与维护（次重点★★★☆☆）

信息系统在完成规划、分析、设计、开发、测试、交付后就给用户使用，系统便进入运行和维护阶段。系统运行和维护是一项长期工作，从大多数信息系统的实际情况看，系统运行与维护阶段占整个系统生命周期的比重为 60%～80%。这一章节虽然是系分专属，但是里面的运维指标其实架构选择题也在拼命考，所以架构红宝书仍然保留。另外软件维护相关也是论文常考的内容。

6.1.运维技术指标（次重点★★★☆☆）

系统开发交付使用后便进入运行和维护阶段，此过程中因系统自身异常或外部环境等因素，易

出现问题致业务间断、不可用或失效。为此需制定应对策略降低异常概率，提升用户体验，可通过量化指标衡量系统运行维护能力，提高管理水平。这些指标其实和可靠性那节凯恩总结的一样。这里再重放一下，这里多了计算公式。【四个指标的含义需要掌握】。

指标名称	含义	计算方式
平均故障修复时间（MTTR）	系统可修复故障的平均维修时间，从系统故障发生到恢复正常状态所花费时间的平均值，含故障通知、排查诊断、修复测试等时间	MTTR = 给定时间段内消耗在系统修复上的总时间 / 维修次数
平均故障间隔时间（MTBF）	系统两次故障之间的平均运行时间，从系统开始正常运行或故障恢复后到下次故障发生的累计运行时间相对故障次数的平均值	MTBF = 多次故障之间系统总运行时间 / 故障总数
平均无故障时间（MTTF）	可更换或不可修理系统或部件提供正常功能的平均持续时间，多个同类系统或部件从开始正常运行到出现故障的持续时间相对同种系统或部件数量的平均值	MTTF = 给定时间周期内系统可正常运行总时间 / 故障总数
平均应答时间（MTTA）	运维团队响应业务部门投诉、业务中断或异常事件所花费的平均时间	MTTA = 给定时间周期内系统出现告警到告警确认之间累计的总时间 / 事件总数

6.2.系统运行管理（非重点☆☆☆☆☆）

系统运行管理是对信息系统运行管控，记录状态，修改扩展，为管理和决策支持。主要包括日常运行管理、系统运行情况记录、对系统运行情况的检查与评价。下表内容看一眼即可不重要。

管理类别	具体内容	详细说明
系统用户管理	用户管理功能	用户账号管理、权限管理、企业外部用户管理和用户安全审计
	统一用户管理	好处：用户使用方便，安全控制加强，减轻管理人员负担提高效率
	身份认证方法	用户名 / 密码（简单常用但安全性差）、IC 卡认证（存在安全隐患）、动态密码（“一次一密码”，时间或次数同步问题）、USB Key 认证（软硬件结合解决安全与易用矛盾）
	用户安全审计	收集、保护和分析数据形成报告，内容包括系统常规情况、用户登录时段及失败记录等
网络资源管理	网络资源管理功能	性能管理（确保通信能力并优化资源使用）、故障管理（检测、定位和排除故障）、配置管理（掌握网络状态）、计费管理（记录资源使用情况）、安全管理（约束控制资源和信息访问）
	网络资源管理系统	管理网络资源数据，支持设计与分配，提供多种基础管理及系统运行管理功能

软件资源管理	软件构件管理	软件构件是可独立配置且复用的单元，用构件库管理，依赖软件平台支持
	软件分发管理	支持工具自动完成软件部署，可实现软件部署、安全补丁分发、远程管理和控制等任务
	文档管理	软件文档指记录的数据和媒体，编制在软件开发中地位重要工作量大

6.3.系统故障管理（非重点☆☆☆☆☆）

系统故障管理是为了尽可能快恢复系统运行，减少对业务运营不利影响，确保服务质量和可用性，包括 5 个基本活动故障监视、故障调查、故障排查、恢复处理、故障收尾。下表内容看一眼即可不重要。

管理环节	具体内容	详细说明
故障监视	设置待监视项目	考虑故障影响范围、紧迫性，重点监视影响大的故障类别，借助工具并合理投入资源；严格管理故障接触人员，制定职责和操作手册，将其及其活动作为监视对象
	监视内容和方法	重点监视人员、操作规范性、系统硬件和软件；对自然灾害进行风险分析并采取容灾防范措施。硬件设备监视借助管理监控工具；软件监视包括性能、缺陷和变更需求，分别通过工具、测试工程师或用户发现；监视各类与系统接触人员行为
故障调查	收集故障信息	收集方式分为自动收集（通过系统监控工具或日志）和人工收集（用户或 IT 部门发现），必要时进行故障隔离保留现场信息
	确定故障位置	硬件设备故障定位相对简单，计算机系统有诊断测试手段，外围及网络设备采用特定检测方式；软件和数据故障定位复杂，需软件调试定位错误代码行
	调查故障原因	初步支持未解决故障时进行。服务台报告故障，故障管理人员判断是否与已有故障相同或相似，创建或更新故障记录。给故障编号并记录分析信息，判断严重程度决定处理流程。故障原因分为 7 类：计划内维护、应用性故障、人为操作、系统软件、系统硬件、相关设备、灾难和灾害
故障排查和恢复处理	硬件设备故障恢复	主机故障启用系统备份，重要系统采用冗余结构；线路或网络连接问题利用备用电路或改变通信路径；其他相关设备故障分析原因，联系供应商维修、调换或更新

管理环节	具体内容	详细说明
	数据库故障恢复	分为事务、系统、介质故障，利用数据库后备副本和日志文件恢复到故障前一致性状态
	应用软件故障恢复	通过软件调试找出错误代码修改并测试，可采用软件容错技术提高可靠性
故障收尾	确认与更新	与用户确认故障是否成功解决，更新故障信息和记录

6.4.软件系统维护（次重点★★★☆☆）

软件维护是指在软件交付使用之后，直至软件被淘汰的整个时期内，为了改正错误或满足新的需求而修改软件的活动。软件的维护活动基于“软件是可维护的”这一基本前提。这一章我们只关注软件维护的分类和手段如下表总结所示。其中可维护性论文也考过。

维护分类	维护原因	维护定义
改正性维护	识别和纠正软件错误、改正软件性能缺陷、排除实施中的误用	进行诊断和改正错误的过程（这就是我们常说的“修 Bug”，它发生在软件上线运行后，用户或测试人员发现了错误或故障，开发人员进行修复以保证系统正常运行）。
适应性维护	使用过程中，外部环境（新硬、软件配置）或数据环境（数据库、数据格式等）发生变化	为使软件适应这些变化而修改软件的过程（这里的“环境变化”可以包括操作系统升级、硬件更新、业务规则改变等。例如将原本在 Windows Server 2012 上运行的系统迁移到 Windows Server 2022，就可能需要适应性修改。）
完善性维护	软件使用过程中，用户提出新的功能与性能要求	为满足这些要求，修改或再开发软件以扩充功能、增强性能、改进效率、提高可维护性的维护活动（该类维护可能包括用户体验优化、功能增强、新增模块、性能调优、代码重构等，不是为了解决故障，也不是因外部环境变化而改动，而是为提升质量和用户满意度）。
预防性维护	预先提高软件的可维护性、可靠性等，为日后改进软件打基础	采用先进软件工程方法对需维护的软件或部分进行重新设计、编码和测试，即“把今天的方法学用于昨天的系统以满足明天的需要”

这里比较难分辨的是适应性维护和完善性维护的对比。这里关键是变化来源的判断。假如题干提到外部环境的改变导致的维护工作，那么就是适应性维护。假如是用户想要更多/更好导致的维护工作，那么就是完善性维护。这个架构 25 年 11 月的题目中也有出现。

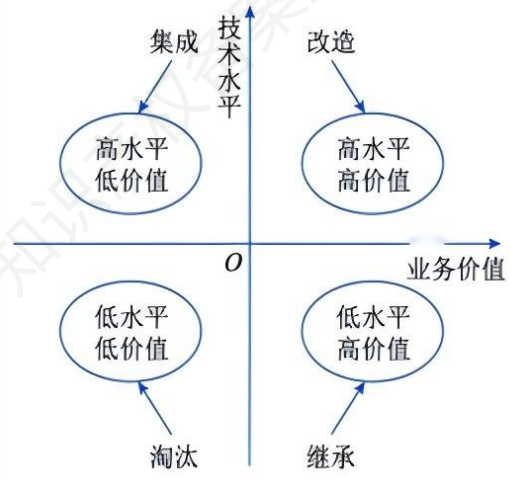
如何提高软件可维护性，要从软件工程的角度出发，使软件开发过程规范，产生文档丰富维护资源，增加软件可维护性。【论文会用到】。

阶段	具体方法	说明
需求分析阶段	明确未来改进和可能修改部分；讨论软件跨平台可移植性并形成解决方案	为后续开发和维护明确方向，提前考虑软件在不同平台的适应性
设计阶段	遵循“高内聚，低耦合”原则；对可能修改处采用灵活可扩充设计；考虑跨平台可移植性设计；加大可复用构件设计力度	提高软件模块独立性，便于修改和扩充，增强软件的通用性和可复用性
编码阶段	采用科学代码规范；强化注释力度，保证注释质量；加大可复用构件使用力度	便于维护人员理解代码逻辑，提高代码复用率，减少重复开发工作
测试阶段	做好测试以减少维护工作量；完善测试相关文档（测试计划、用例、报告等）	保证软件质量，为维护后的回归测试提供基础，确保维护质量
维护阶段	进行严格的配置管理，维护后同步更新系统和用户文档；大维护交付前做好用户培训	保证系统一致性，避免因文档不同步导致的问题，使用户能正确使用软件，减少因使用问题导致的误解

6.5.遗留系统处置（次重点★★★☆☆）

这一章节爱考选择和论文，概念需要理解。遗留系统（Legacy System）是指任何基本上不能进行修改和演化以满足新的变化了的业务需求的信息系统。在遗留系统处置上，对遗留系统的整体评价，以及演化方式的选择是极为重要的环节。软考重点就考遗留系统演化，结合四象限评价办法有不同的演化策略。【选择论文爱考】。

象限	技术与业务价值	策略名称	策略详情
第一象限	高水平、高价值区：技术含量高，有生命力，业务价值高，能满足企业业务运作和决策支持需求	改造策略	包括系统功能增强（在原有系统基础上增加新应用要求，不改变遗留系统本身）和数据模型改造（将旧数据模型转化为新数据模型）
第二象限	高水平、低价值区：技术含量高，但业务价值低，完成局部业务管理，形成信息孤岛	集成策略	对遗留系统进行集成
第三象限	低水平、低价值区：技术含量低，业务价值低	淘汰策略	全面重新开发新系统代替遗留系统。适用情况：企业业务根本变化，遗留系统不适应企业运作；维护人员和文档丢失；开发新系统比改造旧系统成本更合算。可借鉴遗留系统功能设计新系统，降低开发风险
第四象限	低水平、高价值区：技术含量低，不能满足功能或性能要求，但商业价值高，企业业务依赖该系统	继承策略	开发新系统时完全兼容遗留系统的功能模型和数据模型。新老系统并行运行一段时间后逐渐切换到新系统



按照逆时针的顺序记忆。

记忆口诀：该记陶吉。该记住陶**喆**（陶吉吉）。该（改造）记（集成）陶（淘汰）吉（继承）。

还要记忆 X、Y 轴的关系。水平是业务，垂直是技术。沿着箭头方向从低到高。

6.6.新旧系统转换（次重点★★★☆☆）

当新系统开发完毕投入运行，要取代现有系统时，就要进行系统转换。系统转换是指运用某种

方式，由现有系统的工作方式向新系统工作方式的转换过程，也是系统设备、数据、人员等的转换过程。这一章节主要出选择题，要掌握三种转换策略。

转换策略	策略定义	适用场景
直接转换策略	在原有系统停止运行的某一时刻，新系统立即投入运行，中间无过渡阶段	新系统不太复杂或现有系统完全不能使用
并行转换策略	新系统和现有系统并行工作一段时间，试运行后新系统正式替换现有系统	较大的信息系统，或处理过程复杂、数据重要的系统
分段转换策略	直接转换和并行转换方式的结合，分期分批逐步转换	较大的系统；现有系统稳定能适应业务发展；新旧系统转换风险大

7.系统可靠性（次重点★★★☆☆）

本章内容属于次重点内容，主要在选择题中考察。从选择角度来看，主要考察 4 个可靠性指标，以及串并联可靠性分析和可靠性评价。案例考察得很少，可以忽略。论文部分重点考察点就是可靠性评价和设计方法，这个题目很难写，理论为主。

软件可靠性 (Software Reliability) 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。

7.1.软件可靠性设计（次重点★★★☆☆）

不重要，但是论文爱考，假如考到设计，下面都是可以展开讨论的内容。选择题很少考。

技术分类	具体技术	原理及特点
容错设计技术	恢复块设计	将程序划分为恢复块，每个恢复块含多个功能相同但设计不同的程序块，检测到故障时切换备用程序块

	N 版本程序设计	设计多个不同软件版本，对同样输入采用多数表决方式，各版本使用不同算法、编程语言、测试方法等确保相互独立
	冗余设计	在主系统外设计备用软件模块，主模块故障时切换，可提高可靠性但增加额外开销
检错技术	检错技术	在软件中设置检测点和检测机制，通过监测返回结果、运行时间等方式检测软件运行状态，发现故障后停止软件并报警
降低复杂度设计	降低复杂度设计	软件复杂度包括模块内部复杂性和模块间关联复杂性，过高复杂度是软件缺陷根源，通过简化软件结构、优化数据流等方式降低复杂度以提高可靠性
系统配置技术	双机热备	采用主备服务器模式，通过 “心跳” 检测切换
	服务器集群	多台服务器组成单一系统，某台服务器故障时其他服务器可接管

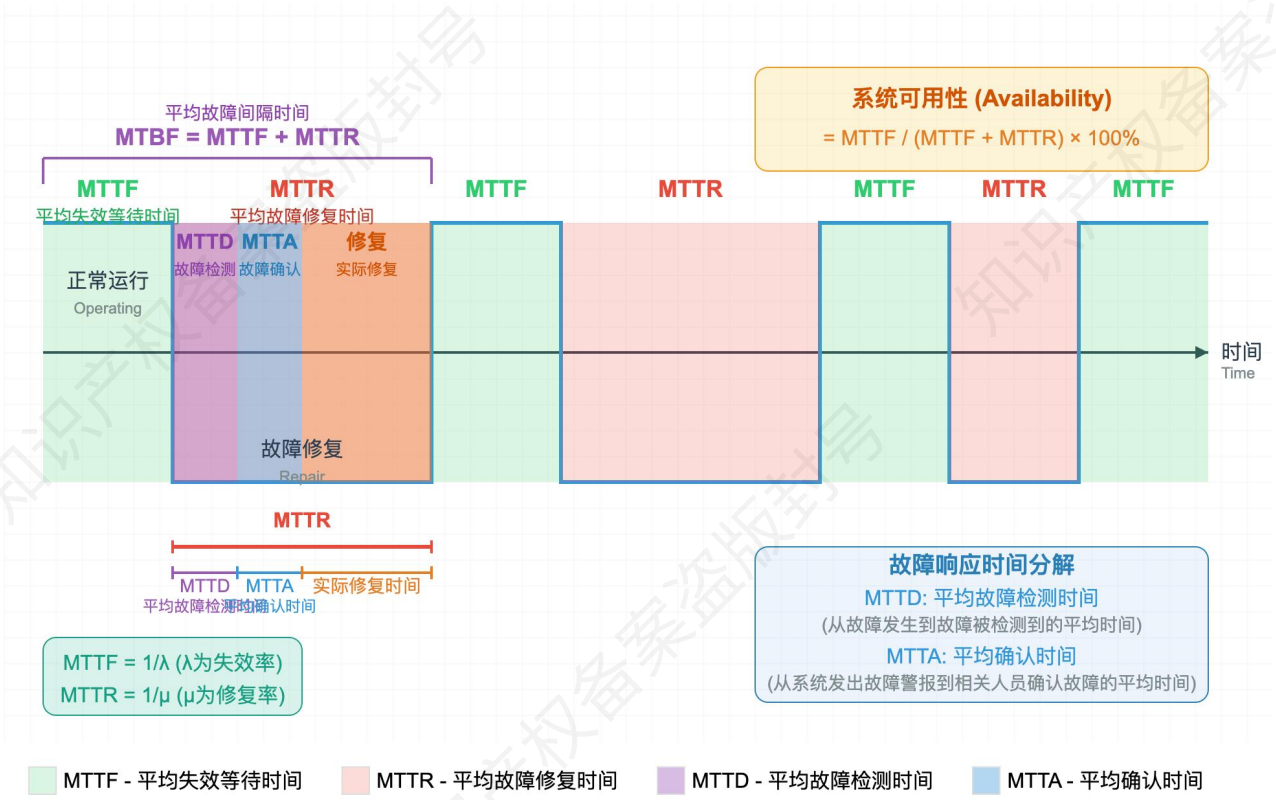
7.2.软件可靠性的定量描述（重点★★★★★）

可靠性这是架构最爱考的一个概念，下面的图熟悉起来就没问题。现在喜欢考偏门的比如 MTTA，MTTD，你要知道它们分别代表什么，建议记忆后缀，比如 A 是确认（Acknowledge）的英文首字母，F 是失败（Failure）的英文首字母等。

指标	英文全称（记忆末尾单词意思）	定义
MTBF	Mean Time Between Failures（平均故障间隔时间）	平均故障间隔时间，是指系统在两次相邻故障之间的平均工作时间（见下图），用于衡量系统的可靠性和稳定性，这里包括了故障检测时间和故障修复时间。
MTTR	Mean Time to Repair/Recover（平均修复时间）	平均修复时间，是指系统从发生故障到恢复正常运行所需要的平均时间
MTTA	Mean Time to Acknowledge（平均响应确认时间）	平均确认时间，是指从系统发出故障警报到相关人员确认故障的平均时间

MTTF	Mean Time to Failure（平均失效时间）	平均故障前时间，是指系统或者产品从开始使用到出现故障（不包括检测和修复）的平均时间
MTTD	Mean Time to Detect（平均发现时间）	即平均故障检测时间，是指从故障发生到故障被检测到的平均时间

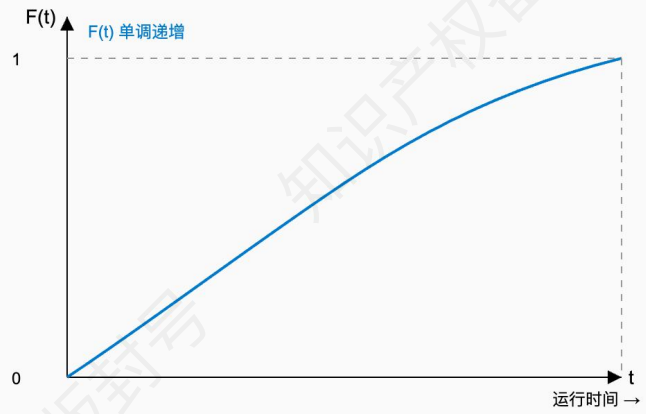
结合下图可以更好地理解上述概念。



在实际应用中，一般 MTTR，MTTD 很小，所以通常认为 MTBF 约等于 MTTF（选择题考过）。

另外这一章节还要关注：

失效概率：软件在运行过程中，其失效概率可以用函数 F(t)来描述。该函数在初始运行时刻失效概率为 0，随着运行时间的增加而不断增大，并在运行时间趋于无穷时接近 1，体现了软件缺陷不可避免的客观规律。为便于分析，通常将 F(t) 视为关于时间的连续、可导函数。



软件失效概率函数 F(t) 随运行时间变化示意图

可靠度：人们用来表示可靠性最为直接的方式就是可靠度，根据可靠性的定义，可靠度就是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。则可靠度的公式为 R(t)=1-F(t) 同样，我们知道 $R(0)=1$，$R(+\infty)=0$ 。

规定时间：指软件运行过程中，中央处理器（CPU）执行程序指令所用的时间总和。使用执行时间来度量软件的可靠性最为准确，效果也最好。

7.3.影响软件可靠性的 5 大因素（非重点☆☆☆☆☆）

不重要，纯概念，可能考选择，你如让你选下面哪些是影响可靠性的主要因素。

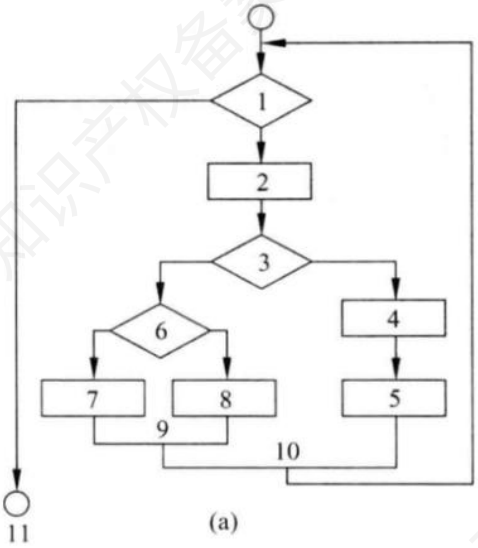
影响因素	含义说明	对可靠性的影响
运行剖面（环境）	软件运行的外部条件与使用场景，如运行平台、硬件环境、用户习惯、输入数据特征等	环境变化可能引入新的故障模式，环境越复杂，潜在故障概率越高
软件规模	软件的代码行数、模块数量、功能范围等规模指标	规模越大，缺陷潜在数越多，可靠性一般降低，除非质量控制严格
软件内部结构	模块间耦合度、内聚性、复杂度、层次结构等	结构合理、模块独立性高的系统更容易测试和维护，可靠性更高
软件的开发方法和开发环境	开发过程是否规范化，工具链是否先进，如自动化测试、CI/CD、静态分析等	规范方法和先进环境可减少缺陷、提升开发质量和可靠性
软件的可靠性投入	在测试、验证、评审、容错、监控等方面的资源投入（时间、人力、资金）	投入足够可显著减少潜在缺陷；投入不足会降低可靠性

记忆口诀：还闺内发钱。还闺蜜的钱。还（环境）闺（规模）内（内部）发（方法）钱（投入）。

7.4.程序复杂程度的定量度量 McCabe 方法（次重点★★★☆☆）

定量度量程序复杂程度的方法很有价值:把程序的复杂程度乘以适当常数即可估算出软件中错误的数量以及软件开发需要用的工作量,定量度量的结果可以用来比较两个不同的设计或两个不同算法的优劣。这个部分只会考选择题。

McCabe 方法是一种用来衡量程序控制结构复杂程度的度量方法,其计算结果称为环形复杂度。该方法关注的核心问题是：程序在运行过程中，可能存在多少条不同的执行路径。为了直观描述程序的控制流，McCabe 引入了流图这一表示方式。流图可以看作是对程序流程图的简化形式，它只描述控制流程本身，而不关心具体的数据处理过程，也不体现判断条件的具体内容。如下图所示。



(a)

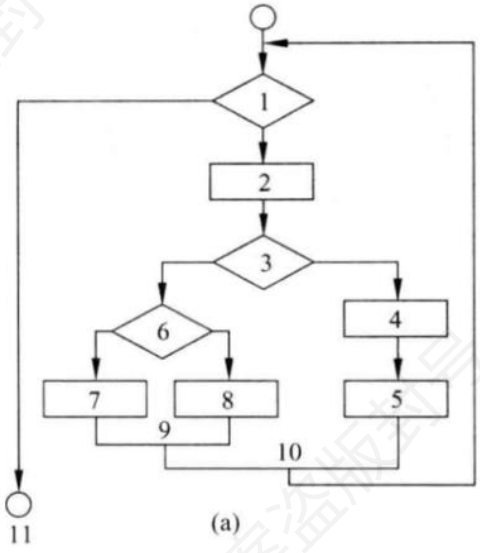
在流图中，结点用圆表示，一个结点可以对应一条语句，也可以表示多条顺序执行的语句；边用箭头表示，反映程序执行时控制从一个结点转移到另一个结点的过程。由结点和边围成的封闭区域称为区域，在计算环形复杂度时，不仅要统计图中封闭的区域，还要把图外部的区域一并计算在内，注意观察下面的例子。基于流图中结点、边和区域之间的关系，就可以用下述 3 种方法中的任何一种来计算环形复杂度。

（1）流图中线性无关的区域数等于环形复杂度（被控制流图划分成的区域数）。

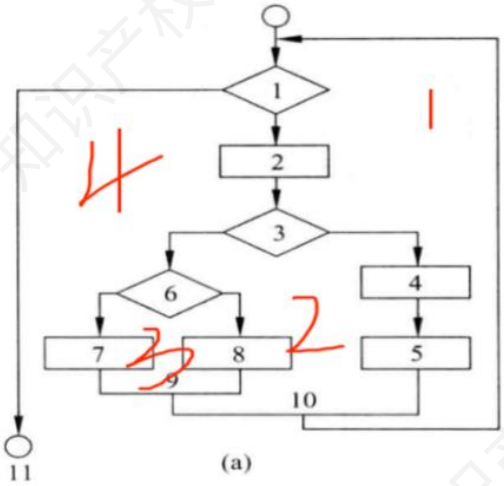
（2）流图 G 的环形复杂度 V(G)=E—N+2，其中，E 是流图中边的条数，N 是结点数。

（3）流图 G 的环形复杂度 V(G)=P+1，其中，P 是流图中判定结点的数目。

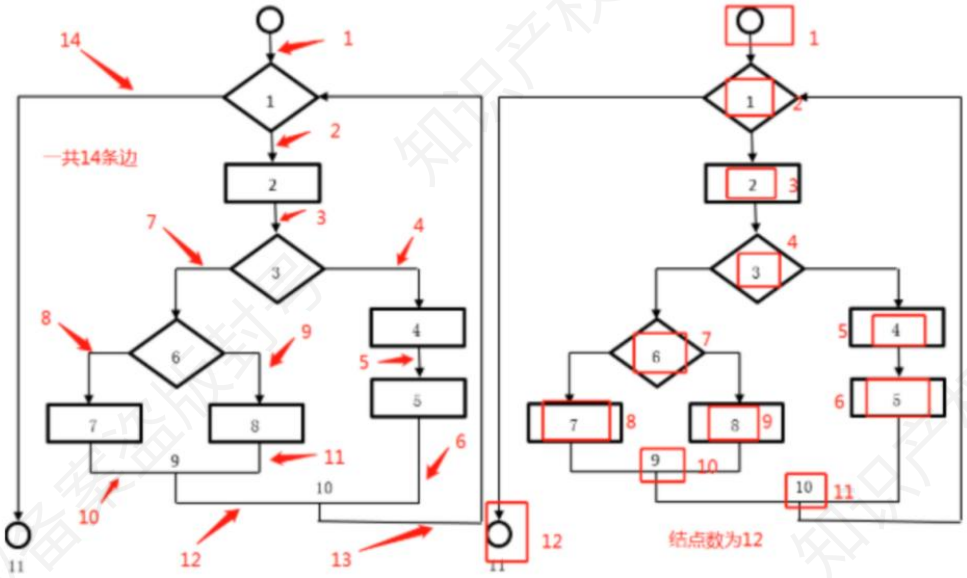
如下所示的流图（简化的流程图）的复杂度，判断方法有三种。



方法 1：你可以数出 4 个线性无关区域（被控制流图划分成的区域数）



方法 2：边数 14，节点数 12，所以根据公式 14-12+2=4



方法 3：判定结点是指具有判断条件的结点，通常是菱形框表示的结点。在该流图中，判定结点有节点 1、节点 3、节点 6，共 P＝3 个。将 P＝3 代入公式 V(G)=P＋1，可得：V(G)=3＋1＝4

7.5.串并联系统（重点★★★★★）

在讨论系统可靠性时，串联系统和并联系统 是两种基本的配置方式，它们对系统的可靠性有着不同的影响。当然真题中出现过串并联混合系统，做到真题的时候就会明白。

7.5.1.串联系统（重点★★★★★）

在串联系统中，所有组件依次连接，即系统的功能依赖于各个组件的正常运行。只有在所有组件都正常工作的情况下，整个系统才会正常运行。串联系统的可靠性是各个组件可靠性的乘积。若任何一个组件失效，整个系统将失败。串联系统可靠性公式如下所示，其中 Rn 是组件可靠性：

$$R_{串联} = R_1 \times R_2 \times \cdots \times R_n$$

7.5.2.并联系统（重点★★★★★）

并联系统中，多个组件提供冗余，以保证即使其中某个组件出现故障，系统仍能够继续工作。并联系统的功能通常是将多个组件连接，以共同实现目标。并联系统的可靠性是 1 减去所有组件都失效的概率。即系统的可靠性为，其中 Rn 是组件可靠性：

$$R_{并联} = 1 - (1 - R_1)(1 - R_2)\cdots(1 - R_n)$$

8.项目管理（次重点★★★☆☆）

选择题每年 1-2 题，概念以及计算都会出现。案例中基本不出现。不会在论文中出现。这里主要关注进度管理的关键路径，自由时差计算。高项考过的同学做这里的题目会觉得非常简单。架构同学简单读一下即可。

8.1.范围管理（次重点★★★☆☆）

范围管理就是要确定项目的边界，也就是说，要确定哪些工作是项目应该做的，哪些工作不应该包括在项目中。这个过程用于确保项目干系人对作为项目结果的产品（或服务），以及开发这些产品所确定的过程有一个共同的理解。对项目范围的管理，是通过 5 个管理过程来实现的（这里有的同学提到 PMBOK 即项目管理知识体系中的范围管理定义有 6 个，多了一个收集需求。这个新老教材都是没有的，大家注意一下）。

过程名称	描述	输入
编制范围管理计划	对定义、确认和控制项目范围的过程进行描述	-
定义范围	详细描述产品与项目范围，编制项目范围说明书作为项目决策基础	项目章程、项目范围管理计划、组织过程资产、批准的变更申请
创建工作分解结构（WBS）	将整个项目工作分解为较小、易管理的组成部分，形成自上而下的分解结构，持续	-

	分解至可管理的工作包，其总和即项目所有工作范围	
确认范围	正式验收已完成的可交付成果	-
范围控制	监督项目和产品的范围状态，管理范围基准变更	-

8.2.进度管理（重点★★★★★）

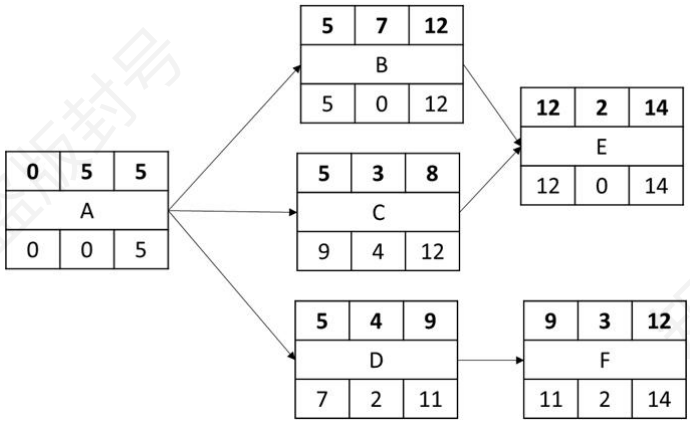
进度管理就是采用科学的方法，确定进度目标，编制进度计划和资源供应计划，进行进度控制，在与质量、成本目标协调的基础上，实现工期目标。下表的内容了解即可。

过程名称	描述	主要作用
活动定义	确定完成项目各项可交付成果而需开展的具体活动	将项目工作包进一步分解为具体活动，为后续进度管理提供基础
活动排序	识别和记录各项活动之间的先后关系和逻辑关系	明确活动开展顺序，构建项目进度逻辑框架
活动资源估算	估算完成各项活动所需要的资源类型和数量	为活动提供资源保障，合理分配资源
活动历时估算	估算完成各项活动所需要的具体时间	为项目进度计划制定提供时间参数
进度计划编制	分析活动顺序、活动持续时间、资源要求和进度制约因素，制订项目进度计划	确定项目活动的执行时间安排，指导项目按计划推进
进度控制	根据进度计划开展项目活动，如果发现偏差，则分析原因或进行调整	确保项目按既定进度计划执行，及时纠正偏差

在现代管理中，人们常用有向图来描述和分析一项工程的计划和实施过程，一项工程常被分为多个小的子工程，这些子工程被称为活动。在有向图中，若以顶点表示活动，弧表示活动之间的先后关系，这样的图简称为 AOV (Activity On Vertex) 网（单代号网络图）。若以顶点表示事件，弧表示活动，称为 AOE (Activity On Edge) 网（双代号网络图）。在软考中看到 PERT 图就想到网络图。

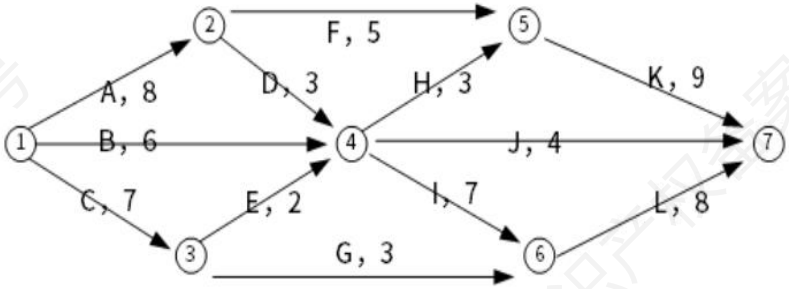
8.2.1.单代号网络（重点★★★★★）

单代号网络图（7 格图）是以节点来代表工作，节点编号表示工作，箭线表示工作之间的逻辑关系。看下图，因为每个工作只用一个节点表示，所以称为“单代号”。



8.2.2.双代号网络图（重点★★★★★）

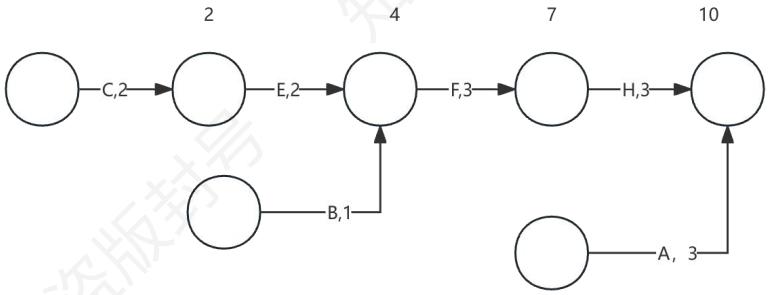
双代号网络图则是以箭线及其两端的节点编号来表示工作。比如，箭线“①-②”代表“从 A 地点运输材料到 B 地点”的工作，其中“①”和“②”是节点编号。由于工作是通过两个节点（编号）来表示的，故而叫作“双代号”。架构系分进度计算相关真题因为他让你求取的往往是最短路径，此时用双代号比较容易。



例：某项目有 8 个作业 A-H，每个作业的紧前作业、所需天数和所需人数见下表。由于整个项目团队总共只有 9 人，各个作业都必须连续进行，中途不能停止，因此需要适当安排施工方案，使该项目能尽快在（10 天）内完工。

本题考察的是项目进度网络计划与资源受限调度。个人建议架构系分中几乎所有进度管理的题

目都用双代号来做，计算起来比较方便。如下图所示。可以看到此时关键路径为 C-E-F-H，仅考虑进度分析不考虑人员安排，项目工期为 10 天。



8.2.3.三点估算法（重点★★★★★）

（1）乐观时间：在任何事情都很顺利的情况下，完成某项工作的时间。

（2）最可能时间：在正常情况下，完成某项工作的时间。

（3）悲观时间：在最不利的情况下，完成某项工作的时间

那么期望时间 =（乐观时间+4*最可能时间+悲观时间）/6

真题计算最基本的工期，总时差一定要学会计算（基本要求）。不会的看我 B 站视频（芝士架构凯恩，进度管理）。

8.3.成本管理（次重点★★★☆☆）

项目成本管理是在整个项目的实施过程中,为确保项目在批准的预算条件下尽可能保质按期完成，而对所需的各个过程进行管理与控制。项目成本管理包括成本估算、成本预算和成本控制三个过程。

过程名称	定义	主要方法 / 内容
成本估算	对完成项目所需成本的估计和计划，是项目计划中重要、关键且敏感的部分	基本估算方法分为自顶向下的估算、自底向上的估算和差别估算法

成本预算	把估算的总成本分配到项目的各工作包，建立成本基准计划以衡量项目绩效	将总成本分配至各工作包，形成成本基准计划
成本控制	保证各项工作在各自的预算范围内进行	

8.4.配置管理（次重点★★★☆☆）

配置是在技术文档中明确说明并最终组成软件产品的功能或物理属性，包括即将受控的所有产品特性，其内容及相关文档、软件版本、变更文档、软件运行的支持数据，以及其他一切保证软件一致性的组成要素。包括如下表的过程。

配置管理过程内容	描述
编制软件配置管理计划	项目启动时，项目经理制订整个项目开发计划，配置管理计划是其一部分，是软件开发工作基础
配置标识	确定进入配置管理的内容形成配置项，明确配置项命名方式及描述信息，是配置管理基础与前提
变更管理和配置控制	对配置项变更加以控制和管理，防止复杂无形的软件在多次变更下失控混乱
配置状态说明（配置状态报告）	有效地记录、报告管理配置所需信息，及时准确给出配置项当前状况，供相关人员了解以加强管理
配置审核	验证配置项对配置标识的一致性，确保软件配置管理有效，杜绝混乱现象

8.5.风险管理（次重点★★★☆☆）

风险管理就是要对项目风险进行认真的分析和科学的管理，这样能够避开不利条件、少受损失、取得预期的结果并实现项目目标，能够争取避免风险发生或尽量减小风险发生后的影响。但是，完全避开或消除风险，或者只享受权益而不承担风险是不可能的。风险管理包括六大过程如下所示，了解即可。凯恩认为风险管理计划编制、风险识别选择题考察概率较高，重点关注。

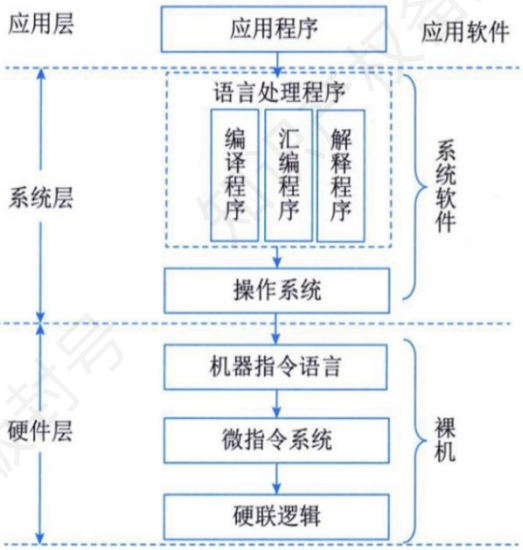
风险管理过程	具体内容
风险管理计划编制	描述如何安排与实施项目风险管理，是项目开发计划的从属计划，内容包括角色与职责、预算、风险类别、风险概率和影响的定义、汇报格式、风险跟踪等；很多项目还有应急计划（风险发生时采取的预先确定措施）和应急储备（项目范围或质量变更时用于减少成本或进度风险的资金）
风险识别	采用系统化方法识别项目中已知和可预测的风险，是反复过程，项目团队应参与；包括确定风险来源、产生条件，描述风险特征，确定影响整个项目的风险事件；在项目生命周期自始至终定期进行，分为收集资料、估计项目风险形势、识别潜在风险三步
风险定性分析	对已识别风险进行优先级排序，包括风险可能性与影响分析（确定风险发生可能性及对项目时间、成本、范围等各方面影响）、确定风险优先级（反映风险对项目综合影响）、确定风险类型
风险定量分析	在不确定情况下进行决策的量化方法，采用灵敏度分析、期望货币价值分析、决策树分析、蒙特卡罗模拟等技术；蒙特卡罗方法将多元函数取值范围问题分解为参数概率问题，用统计方法处理得到综合概率，分析取值范围可能性，关键是找随机数，适用于复杂模型，步骤包括选取变量、分析概率分布、选取样本、模拟价值结果、分析结果
风险应对计划编制	风险应对策略分防范策略（风险发生前采取措施防范）和响应策略（风险发生后采取措施降低损失）；消极风险防范策略常用，目的是降低风险发生概率或减轻损失，如避免、转移、减轻策略；正向风险应对策略有开拓、分享和强大（提高）；风险防范策略需体现在项目计划中，风险响应策略是事件触发，风险发生后执行
风险监控	跟踪已识别风险，监测残余风险和识别新风险，保证风险计划执行，评价计划对减轻风险的有效性

9.计算机组成原理（次重点★★★☆☆）

本章只会在选择题中着重考察，每年会出 3-4 分的题目，但是难度不大。从 2023 年开始架构选择题不太考这里的相关内容。但是以防万一，这里我也补充比较典型的计算题的求解思路，一看就会。

9.1.计算机系统层次结构（次重点★★★☆☆）

计算机系统层次结构研究的是：用“分层 + 抽象”的方式，把硬件到软件的复杂系统组织起来，每一层只关心“该关心的事”。掌握计算机基本分层就可以，选择题考到就是送分。



层级（从低到高）	描述	语言
硬联逻辑级	计算机的内核，由门、触发器等逻辑电路组成	/
微程序级	机器语言是微指令集，微程序一般直接由硬件执行	微指令集
传统机器级	机器语言是该机的指令集，程序可由微程序解释	指令集
操作系统级	一方面直接管理传统机器中的软硬件资源，另一方面是传统机器的延伸	/
汇编语言级	机器语言是汇编语言，由汇编程序完成翻译工作	汇编语言
高级语言级	机器语言是各种高级语言，通过编译程序完成翻译	各种高级语言
应用语言级	为使计算机满足某种用途专门设计，机器语言是各种面向问题的应用语言	应用语言

9.1.1.计算机系统结构的分类（次重点★★★☆☆）

Flynn 在 1966 年提出基于指令流—数据流的计算机系统结构分类方法，旨在从执行模型角度刻画计算机系统的并行性特征，为不同计算机体系结构提供一种统一、抽象且可比较的分类框架。Flynn 把计算机系统分成以下 4 类：

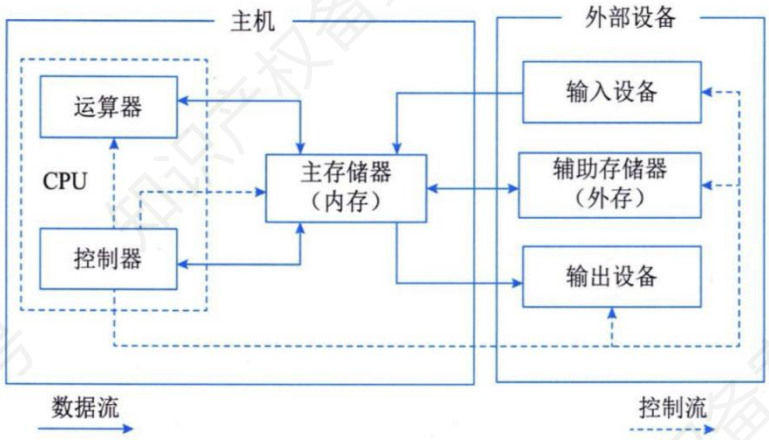
分类	描述	典型代表或相关说明
单指令流单数据流（SISD）	指令部件对单条指令译码和对单个操作部件分配数据	传统单处理器计算机

单指令流多数据流（SIMD）	并行处理机包括多个重复的处理单元，由单一指令部件控制，按照同一指令流的要求为它们分配各自所需的不同数据	并行处理机（矩阵处理机）
多指令流单数据流（MISD）	有 n 个处理单元对同一数据流按不同指令处理，少见，流水线有时被视为 MISD	流水线有时被视为该类型
多指令流多数据流（MIMD）	能实现作业、任务、指令等各级全面并行的多机系统。	高性能服务器、超级计算机

按照软考的常规要求。这里的关键是要记住 SS（SISD）-单指令流单数据流-对应传统单处理器计算机，MM（MIMD）-多指令流多数据流-对应高性能服务器，其他不重要。

9.1.2.冯诺依曼机（次重点★★★☆☆）

冯·诺依曼机的核心思想是“存储程序”概念，即把程序和数据一起存放在同一个存储器中，由控制器自动读取并执行。这是现代计算机体系结构的奠基。冯·诺依曼机的概念模型如下图所示。



这里有两个流向。**数据流：**输入设备或外存把数据送入主存储器，CPU 从主存中取数进行运算，结果再写回主存，最后通过输出设备输出。**控制流：**由控制器发出，指挥运算器、存储器、输入输出设备协同工作。其他组件作用如下表所示。

组成部分	主要功能	示例
运算器	执行算术运算和逻辑运算	CPU 内部运算单元

控制器	解释指令，控制运算器、存储器和输入输出设备的工作；与运算器一起构成 CPU	CPU 内部控制逻辑
主存储器（内存）	存放程序指令和数据，CPU 可直接访问	RAM
输入设备	把外部信息送入计算机	键盘、鼠标、扫描仪
输出设备	将计算结果输出给外部	显示器、打印机
辅助存储器（外存）	容量大，用于长期保存数据	磁盘、固态硬盘、光盘

9.2.数据表示与编码（次重点★★★☆☆）

计算机内部采用二进制形式表示和处理各种数据。数值数据进入计算机后，必须按照一定规则转换为二进制编码，才能被存储、传送和运算。数据表示与编码主要研究整数、小数以及特殊数值在计算机中的表示方法，是计算机组成原理中的基础内容。系分和架构偶尔出现一次。

9.2.1.原码、反码、补码（次重点★★★☆☆）

9.2.1.1.机器数与真值（次重点★★★☆☆）

在计算机中，数值以二进制编码形式存放。计算机内部存储的二进制编码称为机器数，机器数实际表示的数值称为真值。

例如，采用 8 位二进制表示有符号整数时，最高位通常作为符号位：0 表示正数，1 表示负数。若机器数为：10000101，则其真值取决于采用哪一种编码方式。若按照原码解释，该机器数表示 -5；若按照补码解释，该机器数表示 -123。怎么算的不要着急，这里要说明的是同一个二进制在不同的编码规则下，表示的意义不同。

9.2.1.2.原码表示法（次重点★★★☆☆）

原码是一种最直观的有符号数表示方法。其规则是：最高位为符号位，其余各位表示数值的绝对值。对于 8 位有符号数：

+5 的原码：00000101，-5 的原码：10000101

原码的优点是表示形式直观，便于人工阅读。正数和负数只在符号位上不同，数值部分保持一致。但原码存在两个明显缺点。

第一，零有两种表示形式：+0 的原码：00000000，-0 的原码：10000000

第二，原码不便于直接进行加减运算。计算机若采用原码进行运算，需要额外判断操作数符号，并根据符号和绝对值大小决定执行加法还是减法，硬件实现较复杂。

因此，原码主要用于说明有符号数表示思想，在计算机实际整数运算中并不常作为最终表示形式。

9.2.1.3.反码表示法（次重点★★★☆☆）

反码是在原码基础上形成的一种编码方法。其规则如下：

正数的反码与原码相同；负数的反码是在其原码基础上，符号位保持不变，数值位逐位取反。以 8 位表示 -5 为例：-5 的原码：10000101，-5 的反码：11111010

其中，符号位仍为 1，后 7 位由 0000101 取反为 1111010。

因此：+5 的反码：00000101，-5 的反码：11111010

反码在运算处理上比原码有所改进，但仍然存在两个零的问题：+0 的反码：00000000，-0 的反码：11111111。由于零的表示不唯一，反码也不适合作为现代计算机中有符号整数的主要表示方法。

9.2.1.4.补码表示法（次重点★★★☆☆）

补码规则如下：正数的补码与原码相同；负数的补码是在其反码基础上加 1。也可以表述为：负数补码 = 符号位不变，数值位取反，再加 1。

以 8 位表示 -5 为例：-5 的原码：10000101，-5 的反码：11111010，-5 的补码：11111011

因此：+5 的补码：00000101，-5 的补码：11111011

补码具有两个重要特点：

第一，补码中零的表示唯一：0 的补码：00000000。

第二，补码可以统一加法和减法运算。减去一个数可以转换为加上该数的补码。例如：

5 - 5 = 5 + (-5)，用 8 位补码表示如下：

```
  00000101
+ 11111011
--------------
1 00000000
```

最高位产生的进位超出 8 位，舍弃后结果为：00000000，即结果为 0。

这里引出一个问题，就是补码的表示范围。若采用 n 位补码表示有符号整数，其表示范围为：

-2^(n-1) ~ 2^(n-1)-1，

例如，8 位补码的表示范围为：-128 ~ +127，16 位补码的表示范围为：-32768 ~ +32767

补码中负数比正数多一个，是因为补码取消了 “-0” 的表示形式，多出的一个编码用于表示最小负数。例如，8 位补码中的：10000000，表示 -128，而不是 -0。

9.2.2.定点数与浮点数（重点★★★★★）

计算机不仅需要表示整数，还需要表示小数。根据小数点位置是否固定，数值数据可分为定点数和浮点数。

9.2.2.1.定点数（重点★★★★★）

定点数是指小数点位置固定的数值表示方法。小数点本身并不存储在计算机中，而是由系统预先约定其位置。

例如，约定 8 位二进制数中，前 4 位表示整数部分，后 4 位表示小数部分，则 0011.0100：

因为：0011 = 3，0.0100 = 0 × 1/2 + 1 × 1/4 + 0 × 1/8 + 0 × 1/16 = 0.25

所以该定点数表示 3.25。

9.2.2.2.浮点数（重点★★★★★）

浮点数是指小数点位置可以变化的数值表示方法。它类似于数学中的科学计数法。

在十进制中：12345 = 1.2345 × 10^4，0.00123 = 1.23 × 10^-3，

在二进制中浮点数在 IEEE754 里面通常说成三部分：符号位 + 阶码 + 尾数。这里符号位表示数值正负；阶码表示小数点移动的位置；尾数表示有效数字。

例如：101.11 = 1.0111 × 2^2，其中：1.0111 为尾数，2 为阶码，也称指数 2 为基数。

这里离要注意的是，因为尾数有正负，阶码也可能有正负，所以有的教材常把它拆成四部分：阶符 + 阶码 + 数符 + 尾数，阶符表示阶码的正负。

所以可以得出结论，阶码的长度决定着浮点数的数值表示范围，阶码越长，可承载的指数取值区间就越大，能覆盖的极大值与极小值边界就越宽；而尾数的长度则决定浮点数的表示精度，尾数越长，可存储的有效数字位数就越多，对数值的细节刻画就越精准，能区分的相近数值间隔就越小，也就越能降低数值表示的误差。

浮点数的优点是表示范围大，既可以表示很大的数，也可以表示很小的数。但是缺点是表示精度有限，并且许多十进制小数不能用有限位二进制精确表示，因此可能产生舍入误差。例如，十进制小数 0.1 转换为二进制后是无限循环小数（因为不能被 2 的幂次累加逼近，感兴趣的可以了解一下），所以只能用有限位保存其近似值。因此在程序中可能出现浮点数：0.1 + 0.2 ≠ 0.3，这是浮点数表示方式造成的精度误差。

9.3.存储器系统（次重点★★★☆☆）

存储器系统通过多级层次结构组织不同类型的存储器，使计算机系统在保证高性能的同时，获得大容量和低成本，是现代计算机体系结构的重要组成部分。

这里选择题喜欢让你找茬，比如给你下面的一些概念，问你哪个是错的。这里的关键是要知道寄存器最快，辅存最慢，以及 Cache 的作用。这里我需要补充的是 RAID。NAS 没有玩过或者说存储接触不多的同学可能不清楚 RAID 的情况，所以要做一个补充，应付选择题为主。

类型	与 CPU 的访问关系	速度	容量	作用	在多级存储体系中的意义
寄存器	可被 CPU 快速访问，是 CPU 内部存储单元	最快(比 Cache 还快)	极小	暂存 CPU 运算过程中的数据、指令和地址等	作为 CPU 运算时的高速数据暂存区域，提升运算速度
Cache	可快速向 CPU 提供指令和数据	最快	更小	存放当前最急需处理的程序和数据	确保获得高存取速率
主存	可由 CPU 直接访问	快	较小	存放当前正在执行的程序和数据	作为 CPU 直接处理数据和指令的存储区域
辅存	CPU 不可直接访问需传送到主存	较慢	大	存放暂时不参与运行的程序和数据，作为主存的补充和后援	以较低成本提供大容量存储

9.3.1.磁盘调度（次重点★★★☆☆）

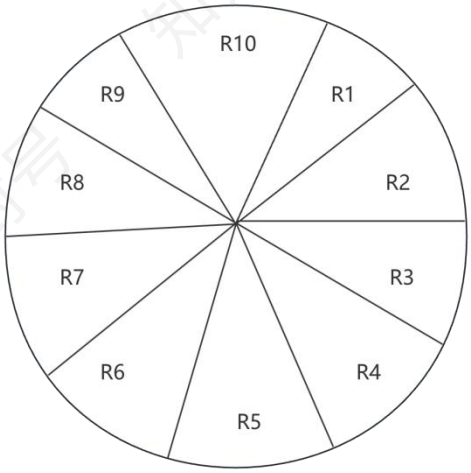
这里是书本上没有提到的，但是选择题会考。下面的例题必须掌握。

例题 1: 在磁盘上存储数据的排列方式会影响 I/O 服务的总时间。假设每磁道划分成 10 个物理块，每块存放 1 个逻辑记录。逻辑记录 R1，R2，… R10 存放在同一个磁道上，记录的安排顺序如下表所示：

物理块	1	2	3	4	5	6	7	8	9	10
逻辑记录	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10

假定磁盘的旋转速度为 30ms/周，磁头当前处在 R1 的开始处。若系统顺序处理这些记录，使

用单缓冲区，每个记录处理时间为 6ms，则处理这 10 个记录的最长时间为（__）；若对信息存储进行优化分布后，处理 10 个记录的最少时间为（__）。



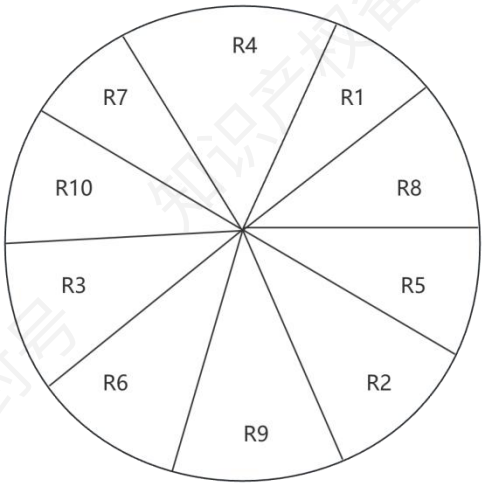
处理完 R1：磁头初始在 R1 开始处，处理 R1 花费 6ms。由于磁盘旋转速度是 30ms / 周，每磁道 10 个物理块，每块旋转时间为 3ms。所以读取 R1 并处理总共需要 3ms + 6ms = 9ms。在这 6ms 处理时间内，磁盘旋转了 6÷3 = 2 个物理块，此时磁头位于 R4 开始处。

处理完 R2：此时磁头位于 R4 开始处，磁头需重新转一圈到 R2，需要经过 8 个物理块，需要 8*3 = 24ms，然后读取 R2 并处理需要 9ms，合计需要 9 + 24 + 9 = 42ms。

处理 R3：需要 9 + 24 + 9 + 24 + 9 = 9 + 33*2 = 75ms。

同理处理完 R10：需要 9 + 33*9 = 306ms。这是第一题的答案。

再看第二问，所谓分布优化，就是让 R1 处理完之后，磁头的位置在 R2 处，那么我们可以做这样一个图。这样就需要 9 *10 = 90ms。



例题 2：在磁盘调度管理中，应先进行移臂调度，再进行旋转调度。假设磁盘移动臂位于 20 号柱面上，进程的请求序列如下表所示。如果采用最短移臂调度算法，那么系统的单应序列应为（__）

请求序列	柱面号	磁头号	扇区号
①	18	8	6
②	16	6	3
③	16	9	6
④	21	10	5
⑤	18	8	4
⑥	21	3	10
⑦	18	7	6
⑧	16	10	4
⑨	22	10	8

A.②⑧③④⑤①⑦⑥⑨ B.②③⑧④⑥⑨①⑤⑦ C.④⑥⑨⑤⑦①②⑧③ D.④⑥⑨⑤⑦①②③⑧

最短移臂调度算法（SSTF，Short -est Seek Time First）是指每次选择与当前磁头所在柱面距离最近的请求进行服务，以减少寻道时间。已知磁盘移动臂位于 20 号柱面上，计算各请求柱面与 20 号柱面的距离：接着计算剩余请求与 21 号柱面的距离：距离当前移臂位置 20 号柱面最近的显然是 21 号柱面，对应的请求序列为 ④、⑥，排除 A、B 选项。其次对于 16 号柱面，序列 ②⑧③访问扇区 3、4、6 的路径最优（按照扇区从小到大，磁盘转动次数较少）。因此选择 C 选项。④⑥的访问位置不能交换。因为这里默认都按扇区从小到大的访问顺序，这样磁盘按顺序旋转就能依次访问到，时间上最优，后面的⑤⑦/②⑧③都是这个规律。

有的同学问柱面相同，扇区也相同怎么处理。那就真的没有优先级差别了。因为磁臂位置一样，旋转到的扇区也一样，谁先谁后在时间上没有区别。

9.3.2.RAID（次重点★★★☆☆）

RAID 技术为缩小 CPU 速度与磁盘存储器速度差距，用多个小磁盘替换大磁盘，合理分布数据以支持多磁盘同时读写，从而提升系统 I/O 性能，现代表独立磁盘冗余阵列，强调其带来性能改善和更高可靠性。常用的 RAID 级别有：RAID 0，RAID 1，RAID 5，RAID 6，RAID 1+0。书上写了 8 种，重要的就是下面 5 个。需要看熟悉，预防选择题。

RAID 级别	硬盘数量	利用磁盘块数	校验	保护能力	读写性能	应用方面
RAID 0	N	N	无	无	最好（因并行而提高）	个人用户
RAID 1	N（偶数）	N/2	无	允许一个设备故障	读和单个磁盘无区别，写则要写两边	适用于存放重要数据，如服务器和数据库存储等领域
RAID 5	N≥3	(N - 1) / N	有	允许一个设备故障	读：RAID 5 = RAID 0（相近的数据读取速度），写：慢于对单个磁盘进行写入（多了一个奇偶校验信息的写入）	一种存储性能、数据安全和存储成本兼顾的存储解决方案
RAID 6	N≥4	(N - 2) / N	有	允许两个设备故障	读：同 RAID 5，写：慢于 RAID 5，需要写入两个奇偶校验信息	相较于 RAID 5，有更强的安全性能
RAID 1+0	N≥4（偶数）	N/2	无	允许两组中各坏一个		集合了 RAID 0 和 RAID 1 的优点，但是空间上由于使用镜像，磁盘利用率为 50%

9.3.3.Cache 基本原理（次重点★★★☆☆）

使用 Cache 改善系统性能的依据是程序的局部性原理，时空局部性都需要了解，选择题会考。

局部性类型	定义	原因
时间局部性	如果一个存储单元被访问，则可能该单元会很快被再次访问	程序存在着循环，比如程序中的循环体部分，其中的指令和数据会被反复使用
空间局部性	如果一个存储单元被访问，则该单元邻近的单元也可能很快被访问	程序中大部分指令是顺序存储、顺序执行的，数据一般也是以向量、数组、树、表等形式簇聚地存储在一起

相关计算题也会考察但一般不会太难。例：设某计算机主存的读/写时间为 100ns ，有一个指令和数据合一的 Cache，已知该 Cache 的读/写时间为 10ns，取指令的命中率为 98 %，取数的命中率为 95 %。在执行某类程序时，约有 1/5 指令需要存/取一个操作数。假设指令流水线在任何时候都不阻塞，则设置 Cache 后，每条指令的平均缓存时间约为：

$$(2\%\times100ns + 98\%\times10ns)+ 1/5 \times (5\%\times l00ns + 95\%\times10ns) = 14.7ns$$

这里的关键是你要知道，要程序取数分为单纯取指令和取操作数，最后的结果是两者的加权之和。取指令的时间 = 直接在缓存里就读取的时间 $98\%\times10ns$ + 从内存读取的时间 $2\%\times100ns$，取数的时间 = 直接在缓存里就读数据时间 $95\%\times10ns$ + 从内存读取的时间 $5\%\times l00ns$。

这里的关键是如何理解约有 1/5 指令需要存/取一个操作数？这里你要理解成有 1/5 的指令会额外存取一个操作数，此时你就明白，最后的总时间就是取指令的时间 + 1/5*取数的时间。

Q: 如何判断一个内存地址是否在缓存里？这个时间没计入，是不是等于这个缓存读取时间？

A: 不是的。这个是可以通过内存地址到 Cache 地址映射判断（下节内容）。这个时间可以忽略。不需要按位去比较。

等于

$$(2\%\times100ns + 98\%\times10ns)+ 1/5 \times (5\%\times l00\ s + 95\%\times10ns) = 14.7ns$$

9.3.4.网络存储技术（非重点☆☆☆☆☆）

此章节不重要，选择题很难得出一道选择题，看过算过，它们各自的连接方式要知道。

存储技术	连接方式	特点	优势和不足	适用场景
直接附加存储（DAS）	通过 SCSI 电缆直连服务器，硬件堆叠	存储操作依赖服务器	存在传递限制，难扩展，影响服务器性能	特定环境（现基本被 NAS 替代）
网络附加存储（NAS）	存储设备通过网络接口与网络直连，类似专用文件服务器	去掉通用服务器多数计算功能，以数据为中心，存储设备独立；支持多种协议；小文件级共享存取	降低成本，优化体系结构，响应快，传输率高，可配置为文件服务器	对文件共享存取有需求场景
存储区域网络（SAN）	通过专用交换机连接磁盘阵列与服务器的高速专用子网，采用块级别存储	存储设备从以太网分离，有 FCSAN、IPSAN、IBSAN 技术	扩大存储能力，提高性能，集中存储、有效存取文件，支持多操作系统，分离存储，提供高可用性，降低运营成本	对存储性能和可用性要求高场景

9.4.输入输出系统（次重点★★★☆☆）

在计算机中，I/O 系统可以有 5 种不同的工作方式，分别是程序控制方式、程序中断方式、DMA 工作方式、通道方式、I/O 处理机。你只需要知道 DMA 是主存和外设之间交换数据即可。

数据传送方式	控制主体	工作原理	特点
程序控制方式	CPU	CPU 执行 I/O 程序实现数据传送，分无条件传送和程序查询方式	简单，CPU 参与度高，效率低
程序中断方式	CPU	CPU 执行现行程序时，遇异常或特殊请求，暂时中止程序处理紧急事件，处理后返回原程序	提高 I/O 能力和 CPU 效率，CPU 需处理中断
DMA 工作方式	DMA 控制器	在主存与外设间实现高速、批量数据交换，DMAC 控制管理数据传输	高速批量传输，减少 CPU 干预
通道方式	通道	在一定硬件基础上利用软件手段实现 I/O 控制和传送	主机和外设并行程度高，减少 CPU 介入
I/O 处理机	I/O 处理机（专用或通用）	具有丰富指令系统和完善中断系统，独立处理 I/O 操作	高度自治，分担 CPU I/O 处理任务

9.4.1.总线（次重点★★★☆☆）

总线是一组能为多个部件分时共享的公共信息传送线路。共享是指总线上可以挂接多个部件，各个部件之间相互交换的信息都可以通过这组公共线路传送。分时是指同一时刻只允许有一个部件向总线发送信息，如果出现两个或两个以上部件同时向总线发送信息，势必导致信号冲突。当然，在同一时刻，允许多个部件同时从总线上接收相同的信息。此章节不重要，真题会做即可。

（1）总线宽度。总线宽度指的是总线的线数，它决定了总线所占的物理空间和成本。例如，32 位的 PCI 总线允许寻址的主存空间的大小为 2^{32}=4G 个单元。

（2）总线带宽。总线带宽定义为总线的最大数据传输速率，即每秒传输的字节。

总线带宽 = 总线宽度×总线频率

例题：假设某系统总线在一个总线周期中并行传输 4B 信息，一个总线周期占用 2 个时钟周期，总线时钟频率为 10MHz ，则总线的频率为 10MHz/2=5Mhz。

这里要注意的是它们之间是除法。你可以这么理解，时钟信号在一秒钟内周期性变化 10M 次，其中 2 次用于总线的传输，那么一秒周期内总线变化或者说可以传输 5M 次，总线周期是 5MHz，单次传输数据量（总线宽度为 4B），则总线带宽为 4B*5MHz=20MBps。

9.5.指令系统（次重点★★★☆☆）

此章节会考选择题，其中复杂指令系统计算机和精简指令系统计算机的区别要知道。指令系统是指示计算机执行某些操作的命令，一台计算机的所有指令的集合构成指令系统，也称为指令集。

指令系统类型	指令特点	寻址方式	实现方式	其他
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

9.6.流水线技术（重点★★★★★）

这一章节的内容只会在选择题考察，关键是要记住下面这些公式，光看没用，必须对照着把选择题做会。新的系分教材已经弱化这块内容，优先级略有降低。

流水线技术把一项任务分解为若干项顺序执行的子任务,不同的子任务由不同的操作部件负责执行，而这些部件可以同时并行工作。在任一时刻，任一任务只占用其中一个操作部件，这样，就可以实现多项任务的重叠执行，从而提高了工作效率。

9.6.1.流水线执行时长（重点★★★★★）

在传统流水线（无并行处理特殊情况）执行时长公式等于

第一条指令顺序执行时间+（指令条数 - 1）* 周期

这里的周期就是流水线中各阶段中占最长时间的操作的时间（关键）

9.6.2.吞吐率（重点★★★★★）

流水线的吞吐率（TP）也称为平均吞吐率或实际吞吐率，是指在单位时间内流水线所完成的任务数量或输出的结果数量，其计算公式如下：

吞吐率 $TP = \frac{n}{T_k}$ 其中， n 为任务数，T_k 为处理完成 n 项任务所用的时间。如果流水线中各段的执行时间不完全相等，假设各段的执行时间分别为 t_1，t_2, t_k，则实际吞吐率为：

$$TP = \frac{n}{\sum_{i=1}^{k} t_i + (n-1)\max(t_1, t_2, t_k)}$$

此时，流水线的最大吞吐率为：

$$TP = \frac{1}{\max(t_1, t_2, t_k)}$$

也就是说，当流水线中各段的执行时间不完全相等时，吞吐率主要是由流水线中执行时间最长

的那个功能段来决定的。

9.6.3.加速比（重点★★★★★）

如果顺序执行所用的时间为 T_0 ，使用流水线的执行时间为 T_k 则计算流水线加速比的基本公式如下：

$$S = \frac{T_0}{T_k}$$

当流水线的各个流水段的执行时间不相等时，一条 k 段线性流水线完成 n 个连续任务的实际加速比为：

$$TP = \frac{n\sum_{i=1}^{k} t_i}{\sum_{i=1}^{k} t_i + (n-1)\max(t_1, t_2, t_k)}$$

例如，假设某流水线浮点加法器分为 5 段，若每一段所需要的时间分别是 6ns、7ns、8ns、9ns 和 6ns。则其加速比为

$$S = \frac{(6+7+8+9+6)n}{(6+7+8+9+6)+9(n-1)} = \frac{36n}{36+9(n-1)} = \frac{4n}{4+n-1}$$

其最大加速比为

$$S_{max} = \lim_{n\to\infty} \frac{4n}{4+n-1} = 4$$

9.6.3.1.多条流水线的情况（次重点★★★☆☆）

如果系统中同时存在多条流水线，则需要进行变通处理。例如，假设指令由取指、分析、执行 3 个子部件完成，并且每个子部件的时间均为 t。若采用常规标量单流水线处理机，连续执行 12 条指令，流水线执行时长公式 = 第一条指令顺序执行时间+（指令条数 - 1）* 周期 ，则共需

$$3t + (12-1)*t = 14t$$

若采用度为 4 的超标量流水线处理机，连续执行上述 12 条指令，则因为同时运行 4 条流水线，平均每条流水线只需执行 **3 条**指令，因此只需 3t + **（3-1）t** = 5t。

9.7.阿姆达尔解决方案（次重点★★★☆☆）

这一章节只会在选择题中出现，关键是下面的题目你会做就行，其实不用了解阿姆达尔定律。

例如，在某计算机系统中，假设某一功能的处理时间为整个系统运行时间的 50%，若使该功能的处理速度加快 10 倍，则根据阿姆达尔定律，这样做可以使整个系统的性能提高。这里根据中学知识，处理速度和时间是反比，假设原本总的运行时间是 1，处理时间是 0.5，加快 10 倍之后，时间就是原来的 0.1 就是 0.05，加速的时间 0.05 + 不能被加速的时间 0.5（1-0.5 得到不能被加速的时间），那么加速之后的总时间就是 0.55。那和原来的比值就是 1/（0.55） = 1.818

$$R = \frac{1}{（1 - 0.5 + \frac{0.5}{10}）} = 1.818 \text{ 倍}$$

9.8.系统性能评估（次重点★★★☆☆）

9.8.1.测试程序（次重点★★★☆☆）

把应用程序中用得最多、最频繁的那部分核心程序作为评估计算机系统性能的标准程序，称为基准测试程序。当然可以进一步划分，形成下表提到的四种测试程序以及准确度顺序。

类型	描述	准确度	适用场景
真实的程序	直接基于实际应用程序的性能表现，能够全面反映计算机在实际使用中的表现。	最准确	全面评估实际工作负载的计算机性能
核心程序	应用程序中最频繁、最重要的部分，能够代表计算机处理的常见任务。相比真实程序，可能存在一定的简化和理想化。	较为准确	评估常见、关键任务的性能表现
小型基准程序	通常较小，针对特定的计算任务进行测试，能提供一定的性能评估，但未涵盖复杂任务或实际工作负载，代表性较低。	一般准确	针对特定小规模计算任务的性能评估
合成基准程序	通过合成多个不同任务来模拟实际工作负载，但因由多个简化任务组成，测试结果不如真实程序和核心程序准确，适合提供大致性能趋势评估。	最不准确	提供大致性能趋势的评估，适合初步比较不同系统的性能

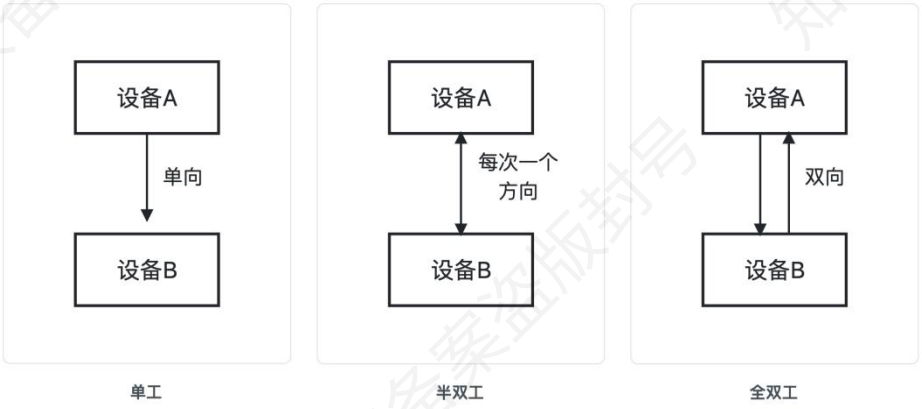
10.计算机网络（次重点★★★☆☆）

选择题会重点考察，案例论文基本不出现。复习方式以刷真题为主。

10.1.数据通信基础（次重点★★★☆☆）

10.1.1.传输信道（次重点★★★☆☆）

按照数据传送的方向与时间不同，信道传输可以分为单工、半双工和全双工三种传输方式。



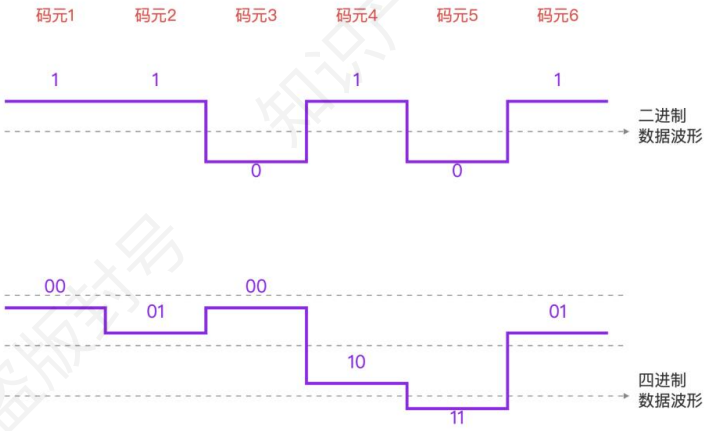
传输方式	数据传送方向与时间特点	示例
单工	数据只能在一个固定方向传输，不能反向传输	无线电广播、电视广播
半双工	通信双方都能发送和接收信息，但不能同时进行，需交替进行	对讲机
全双工	两个方向的传输能同时进行，有独立的发送和接收信道	一般电话系统、交换式以太网、计算机之间的通信、手机

10.1.2.码元的概念（次重点★★★☆☆）

码元是用时域波形表示数字信号时，代表不同离散数值的基本波形。比如二进制码元只有 “0” 和 “1” 两种状态，而四进制码元有 4 种状态，八进制有 8 种状态。

一个码元可以携带多个比特的信息：比如八进制码元有 8 种状态，需要 3 个比特（$2^3=8$）来表示，因此 1 个八进制码元可携带 3 个比特的信息。通过增加码元的状态数（如从二进制到十六

进制），能在相同码元传输速率下提高数据传输速率。



10.1.3.奈氏准则（重点★★★★★）

奈氏准则指出:在带宽为 W(Hz)的理想低通信道中,码元传输的最高速率为 2W(码元 / 秒)。如果是 V 进制码元（即码元有 V 种状态），则极限数据传输率为:

极限数据传输率 $=2W \times \log_2 V$（b/s）

这里的 “理想” 指无噪声，且信道带宽受限。比如带宽为 4kHz 的理想信道，采用 16 进制码元（V=16）时，极限传输率为 $2 \times 4000 \times \log_2 16 = 32000$ b/s（即 32kb/s）。

奈氏准则的意义：激励我们通过更先进的编码技术，让每个码元携带更多比特（提高 V），从而提高传输速率。

10.1.4.香农公式（重点★★★★★）

实际信道存在噪声，香农公式给出了有噪声时的极限传输速率:

$C = W \times \log_2 (1 + S/N)$（b/s）

其中，W 是信道带宽（Hz），S 是信号平均功率，N 是噪声平均功率，S/N 称为信噪比（常用分贝 dB 表示：信噪比（dB）$=10 \times \log10(S/N)$）。

香农公式表明：信道带宽越宽、信噪比越大，极限传输速率越高。只要实际传输速率低于 C，

就一定能找到无差错传输的方法；超过 C 则必然出现差错。

例：某无线通信信道的带宽为 5 MHz，信号平均功率为 10 mW，噪声平均功率为 0.1 mW。

（1）求该信道在有噪声情况下的极限传输速率（单位：Mb/s）。

（2）若系统当前实际传输速率为 20 Mb/s，是否可能做到无差错传输?

已知条件：带宽 W = 5MHz = 5×10^6 Hz，信号功率 S = 10 mW，噪声功率 N = 0.1 mW。

求信噪比（线性值）：$S/N = \frac{10}{0.1} = 100$，代入香农公式：

$$C = W \cdot \log_2(1 + S/N)$$
$$C = 5 \times 10^6 \cdot \log_2(1 + 100)$$
$$= 5 \times 10^6 \cdot \log_2(101)$$

$$\log_2(101) \approx \frac{\log_{10}(101)}{\log_{10}(2)} \approx \frac{2.0043}{0.3010} \approx 6.656$$

得到极限速率：

$$C \approx 5 \times 10^6 \times 6.656 \text{ b/s} \approx 33.28 \text{ Mb/s}$$

判断是否可无差错传输：当前实际速率 20 Mb/s < 极限速率 33.28 Mb/s，所以理论上可实现无差错传输（只要采用合适的编码方式）。

10.1.5.数据编码（次重点★★★☆☆）

在计算机中，数据是以离散的二进制比特流方式表示的,称为数字数据。在网络中传输时，通信信道有模拟信道和数字信道两种类型。计算机数据在模拟信道中传输时，要将数字信号转换成模拟信道能够识别的模拟信号；在数字信道中传输时，要把计算机中的数字信号转换成网络媒体能够识别且利于网络传输的数字信号。

10.1.5.1.模拟数据编码（次重点 ★★★★☆）

模拟数据编码是指：将计算机中的数字数据在模拟信道中传输时，需要通过调制技术，将数字信号转换为适合模拟信道传输的模拟信号。调制的本质是对载波进行波形变换或频谱变换，使其某

一物理参数随数字信号变化。最基本的调制方式包括：调幅、调频、调相三种。

调制方式	核心定义	核心特点
调幅（AM/ASK）	载波振幅随基带数字信号变化	实现简单、技术成熟；抗干扰能力差
调频（FM/FSK）	载波频率随基带数字信号变化	实现相对简单；抗干扰能力较强
调相（PM/PSK）	载波相位随基带数字信号变化	抗干扰能力强；实现技术较复杂

10.1.5.2.数字数据编码（次重点 ★★★★☆）

在数字信道中传输时,要对计算机中的数字信号重新编码后进行基带传输。在基带传输中，数字信号的编码方式主要有以下 5 种。这里，基带传输是指数字信号在传输过程中不经过调制，直接以原始数字脉冲的形式在信道中传输，其频谱从零频开始分布。在数字信道中，通常采用基带传输方式，并通过线路编码来解决同步和识别问题，这个和前面的模拟信号不一样需要了解。

编码名称	核心编码规则	优缺点
不归零编码	用低电平表示二进制 0，高电平表示二进制 1	缺点无法判断每一位的开始与结束，收发双方难以保持同步；需额外信道传送同步信号
曼彻斯特编码	用电平跳变表示二进制，每比特中间均有跳变（既作时钟信号，也作数据信号）；高到低跳变表示 1，低到高跳变表示 0。是数据通信常用编码方式	优点无须另发同步信号；缺点编码效率低（传送 10Mb/s 数据需 20MHz 脉冲）；
差分曼彻斯特编码	曼彻斯特编码的改进版，每比特中间跳变仅用于同步；每比特开始无跳变表示 1，有跳变表示 0。是数据通信常用编码方式	优点无须另发同步信号；缺点编码效率低
多电平编码	码元可取多个电平之一，每个码元可代表多个二进制位（例：M=4 时，1 个码元表示 2 个二进制位，$M=2^n$）	优点数据速率大于波特率，可提高频带利用率；缺点抗噪声特性差，传输中信号易畸变

4B/5B 编码	用 5bit 二进制码代表 4bit 二进制码	优点提高编码效率，降低电路成本；解决曼彻斯特类编码时波特率过高、硬件成本高昂的问题

10.1.6.差错控制－校验码（次重点★★★☆☆）

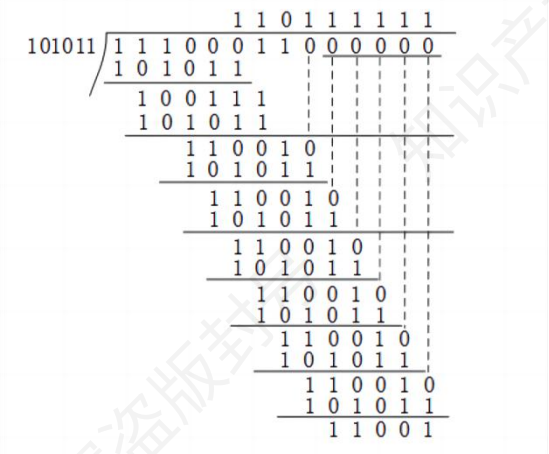
CRC（Cyclic Redundancy Check，循环冗余校验）校验码是一种用于数据传输差错控制的校验码，之前很爱考选择题。下面直接看例题。

若信息码字为 111000110，生成多项式 $G(x)=x^5+x^3+x+1$，则计算出的 CRC 校验码为（__）。

(1) 首先确定生成多项式 $G(x)=x^5+x^3+x+1$，其对应的二进制码为 101011（根据多项式中各项系数确定 $G(x)=1*x^5+0*x^4+1*x^3+0*x^2+1*x+1*x^0$，有该项则对应位为 1，无则为 0，你把这些系数按照顺序写出来就是 101011）。

(2) 信息码字为 111000110，由于生成多项式 $G(x)$ 的最高次幂为 5，所以在信息码字后面补 5 个 0，得到 11100011000000。

(3) 用信息码 11100011000000 除以生成多项式对应的二进制码 101011，进行模 2 除法（异或运算的规则为：如果两个值不相同，异或结果为 1；如果两个值相同，异或结果为 0。得到余数为 11001。11001 就是最后的答案。



10.2.网络体系结构与协议（次重点★★★☆☆）

10.2.1.开放系统互联参考模型 OSI/RM（重点★★★★★）

在 OSI 参考模型提出时，TCP/IP 模型已经在 ARPANET 和互联网中得到了广泛应用。TCP/IP 模型相对简单、实用，更注重实际应用和网络互联；而 OSI 参考模型则更具理论性和系统性，层次划分更加细致。OSI 模型不会直接考你具体有哪些层次，而是会问你哪些协议属于哪些层次。所以下表的内容需要熟悉，特别是最后一列的常见协议。



层级	名称	功能描述	常见协议
7	应用层	为应用软件提供网络服务，如 HTTP、FTP、SMTP 等	HTTP（超文本传输协议，用于网页浏览）、FTP（文件传输协议，用于文件上传和下载）、SMTP（简单邮件传输协议，用于邮件发送）、POP3（邮局协议版本 3，用于邮件接收）、DNS（域名系统，用于域名与 IP 地址转换）、Telnet（远程登录协议，用于远程控制）、SNMP（简单网络管理协议，用于网络管理）
6	表示层	数据格式转换、数据加密解密、数据压缩解压等	SSL/TLS（安全套接字层 / 传输层安全

5	会话层	建立、管理和终止应用程序之间的会话	NetBIOS（网络基本输入 / 输出系统，用于在局域网中实现会话管理等功能）、RPC（远程过程调用，允许程序调用另一台计算机上的程序或服务）
4	传输层	提供端到端的数据传输服务，如 TCP（可靠连接）和 UDP（无连接）	TCP（传输控制协议，面向连接，提供可靠数据传输）、UDP（用户数据报协议，无连接，传输速度快但不保证可靠性）
3	网络层	负责数据包从源到目的地的传输和路由选择	IP（网际协议，用于数据包寻址和路由）、ICMP（互联网控制报文协议，用于网络诊断和控制）、IGMP（互联网组管理协议，用于组播管理）、ARP（地址解析协议，用于将 IP 地址解析为物理地址）、RARP（反向地址解析协议，用于将物理地址解析为 IP 地址）
2	数据链路层	传输有地址的帧，错误检测和纠正，流量控制	PPP（点到点协议，用于拨号上网等场景）、HDLC（高级数据链路控制协议，用于广域网连接）、Ethernet（以太网协议，用于局域网通信）、VLAN（虚拟局域网协议，用于在局域网中划分虚拟子网）、MAC（媒体访问控制协议，定义数据链路层的访问控制方式）
1	物理层	定义物理设备标准，如电缆规范和信号传输速率	RS - 232（串行通信接口标准）、RJ - 45（以太网接口标准）、IEEE 802.3（以太网物理层标准）、光纤标准（如光纤通道协议 FC 等，用于光纤通信）

通过口诀 “物联网书会使用” 记忆 7 层顺序，对应物理层、数据链路层、网络层、传输层、会话层、表示层、应用层。

10.2.2.TCP/IP 模型（重点★★★★★）

在 20 世纪 70 年代诞生了 TCP/IP 协议，它包括传输控制协议（TCP）和网际协议（IP）等一系列协议，用于解决网络互联和数据传输的问题。TCP/IP 协议以其简单、实用和高效的特点，逐渐成为 ARPANET 以及后来互联网的核心协议。TCP/IP 可以分为下面四个层次。

通过口诀“应传网接”记忆 4 层顺序，应用层，传输层，网际层，网络接口层

层级	名称	功能描述
4	应用层	为终端用户的应用提供网络服务，如 HTTP、FTP、SMTP、DNS 等。

3	传输层	负责在网络中提供端到端的数据传输服务，主要协议有 TCP（面向连接、可靠）和 UDP（无连接、不可靠）。
2	互联网层	负责将数据包从源路由到目的地，主要协议是 IP。还包括 ICMP、IGMP 等辅助协议。
1	网络接口层	负责在物理网络媒介上进行数据传输，包括数据链路层和物理层的功能，如以太网、Wi-Fi 等。

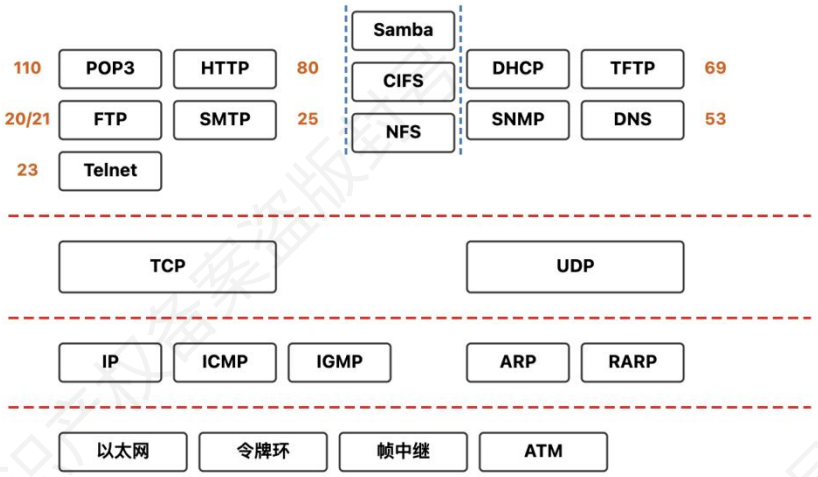
10.3.常见的应用层协议（次重点★★★☆☆）

10.3.1.常见的应用层协议（次重点★★★☆☆）

应用层是最高层，为用户提供各种网络应用服务，如 HTTP、FTP、SMTP 等。TCP/IP 模型中也有应用层，包含了类似的各种应用协议，如 HTTP 用于网页浏览、SMTP 用于电子邮件传输等。常见的应用层协议如下，选择题主要考察这些协议占用的端口以及对应的传输层协议是 TCP 还是 UDP。

传输层	协议名称	占用端口	功能描述
TCP	POP3	110	用于从邮件服务器将邮件下载到本地客户端，允许用户访问并管理存储在服务器上的电子邮件。
	FTP	20（数据端口）、21（控制端口）	用于在网络上进行文件传输，20 端口传输数据，21 端口传输控制命令，实现文件的上传和下载。
	Telnet	23	提供远程登录服务，允许用户通过网络远程连接到其他计算机系统，像在本地一样操作远程计算机。
	HTTP	80	用于在 Web 浏览器和 Web 服务器之间传输超文本，是万维网数据通信的基础，实现网页内容的请求和响应。
	SMTP	25	用于发送电子邮件，将邮件从发件人的邮件客户端传输到邮件服务器，以及在不同邮件服务器之间传输邮件。
UDP	DHCP	67（服务器端）、68（客户端）	用于自动分配 IP 地址等网络配置参数，减少手动配置的工作量，提高网络管理效率。
	SNMP	161（管理端）、162（代理端）	用于网络管理，管理端通过此协议收集网络设备（如路由器、交换机等）的信息，进行监控和管理。

	TFTP	69	用于简单的文件传输，常用于设备初始化时的文件传输等场景，相对 FTP 更简单、占用资源少。
	DNS	53	负责将域名解析为对应的 IP 地址，或者反向解析，是实现网络访问中域名与 IP 地址转换的关键服务。





10.3.2.传输层协议（次重点★★★☆☆）

传输层负责在不同主机上的进程之间提供可靠的端到端通信服务，主要协议有 TCP 和 UDP。这一章节主要是要知道 TCP 和 UDP 的特点，选择题会辨析就可以。

TCP： 在 IP 协议提供的不可靠数据服务的基础上，采用了重发技术，为应用程序提供了一个可靠的、面向连接的、全双工的数据传输服务。TCP 协议一般用于传输数据量比较少，且对可靠性要求高的场合。

UDP：是一种不可靠的、无连接的协议，可以保证应用程序进程间的通信，与 TCP 相比，UDP 是一种无连接的协议，它的错误检测功能要弱得多。可以这样说，TCP 有助于提高可靠性，而 UDP 则有助于提高传输速率。UDP 协议一般用于传输数据量大，对可靠性要求不是很高，但要求速度快的场合。

10.3.3.网络层协议（次重点★★★☆☆）

网络层主要负责将分组从源节点传输到目的节点，实现网络间的路由选择，主要协议是 IP。TCP/IP 模型的网络层也承担着类似的功能，核心协议也是 IP，负责处理网络地址、路由选择等，在这一层上两者的功能和核心协议基本相同。这里有哪些协议需要掌握，选择题会考。

协议名称	功能描述
IP	负责网络层核心功能，包含数据包的路由与传输，不保障可靠性
ARP/RARP	承担 IP 地址与物理地址相互转换的工作
ICMP	处理网络中的错误检测与报告，辅助 IP 协议运行
IGMP	负责多播组的管理以及组播数据的传输

10.4.网络地址（次重点★★★☆☆）

Internet 依靠 TCP/IP 协议簇在全球范围内实现不同硬件结构、不同操作系统、不同网络系统的互联。在 Internet 上，每个节点都依靠唯一的 IP 地址互相区分和相互联系。IP 地址是一个 32 位的二进制数逻辑地址（这种表示方式称为 IPv4），为了人们使用方便，习惯上将这个 32 位的数字划分成 4 个字节，并在每个字节之间以“.”来区分。

例如，IP 地址 11000000 10101000 11001000 10000000，每字节用十进制数来表示，字节之间用圆点分隔，表示为 192.168.200.128。每个 IP 地址由两部分组成，分别是网络号和主机号。网络号用于唯一标识一个网络，主机号则确定了某个网络上的某一台主机。根据网络号和主机号的不同划分，IP 地址可以分为 5 类，如下表所示。

类别	网络号位数	主机号位数	第一个字节取值范围（十进制）	用途
A 类	8	24	0 – 127	大型网络
B 类	16	16	128 – 191	中等规模网络

C 类	24	8	192 – 223	小型网络
D 类	–	–	224 – 239	组播地址
E 类	–	–	240 – 255	保留地址，用于实验等

实际上软考的真题设计的知识还有广播地址，回环地址等，凯恩梳理如下。特别是网络地址，主机地址， 子网地址和广播地址是必须掌握的。

地址类型	含义	特点及用途
网络地址	用于标识一个网络	网络中的所有主机都属于这个网络，网络地址的主机位全为 0。例如在 C 类网络 192.168.1.0/24 中，192.168.1.0 就是网络地址，用于标识整个这个网段
主机地址	用于标识网络中的具体主机	在一个网络中，主机地址是分配给每台主机的唯一地址，主机位不全为 0 和全为 1。比如在 192.168.1.0/24 网络中，192.168.1.1 到 192.168.1.254 都可以作为主机地址分配给具体的设备
广播地址	用于向同一网络中的所有主机发送消息	主机位全为 1。如在 C 类网络 192.168.1.0/24 中，广播地址为 192.168.1.255，当发送数据到这个地址时，该网络中的所有主机都会接收并处理这条消息
子网地址	当对一个大网络进行子网划分后，用于标识各个子网	是在网络地址的基础上，根据子网掩码进一步划分出来的，用于区分不同的子网。例如将 192.168.1.0/24 划分为多个子网，每个子网都有自己的子网地址
组播地址	用于标识一组主机，实现组播通信	可以让数据发送到一组特定的主机，而不是整个网络中的所有主机。D 类地址就是组播地址，范围是 224.0.0.0 到 239.255.255.255。常用于视频会议、在线直播等应用场景，多个接收者可以加入同一个组播组来接收数据
回环地址	用于本地主机进行自我测试和通信	主要是 127.0.0.1，它始终指向本地主机自身。当应用程序使用回环地址发送数据时，数据不会发送到网络中，而是直接在本地主机的网络协议栈中进行处理，常用于测试网络应用程序、本地进程间通信等

10.5.子网和子网掩码（次重点★★★☆☆）

子网：可以从原本的 N 比特主机号当中选出前 K 个比特来作为子网号，用剩余的 N 减 K 个比特作为主机号，这样的话就可以划分出 2 的 K 次方个大小相等的子网。是可以划分成多个网络。



子网掩码：子网掩码是一个 32 位的二进制数，其网络标识和子网标识部分全为 1，主机标识部分全为 0。判断两台计算机是否在同一个子网内，需要用到子网掩码，其方法是将两个 IP 地址与给定的子网掩码分别进行逻辑与运算，如果结果相等，则属于同一个子网，否则就不属于同一个子网。

例子：假设存在一个 B 类主网络 172.16.0.0/16（默认网络位 16 位：前 16 位为网络号，后 16 位为主机位）。为了细分网络，我们从主机位借 4 位作为子网位，因此：子网掩码变为 255.255.240.0（二进制：11111111.11111111.11110000.00000000），前缀长度为 16+4=20 位；网络号结构：16 位主网络号 +4 位子网号 +12 位主机号（子网号是从主机位借位后新增的层级）。

现在我们想判断设备 A 和设备 B 是不是处于同一网络：已知设备 A：IP 地址：172.16.10.5，子网掩码：255.255.240.0（/20）；设备 B：IP 地址：172.16.14.8 子网掩码：255.255.240.0（/20）

首先计算设备 A 172.16.10.5 计算网络号（含子网号）：

IP 地址二进制：10101100.00010000.00001010.00000101

子网掩码二进制：11111111.11111111.11110000.00000000

两者按位与结果（网络号）：10101100.00010000.00000000.00000000 → 十进制 172.16.0.0/20

其中，172.16 是主网络号，0000（二进制）是子网号（4 位子网位全为 0）。

然后计算设备 B172.16.14.8 网络号（含子网号）：

IP 地址二进制：10101100.00010000.00001110.00001000

子网掩码二进制：11111111.11111111.11110000.00000000

两者按位与结果：10101100.00010000.00000000.00000000 → 十进制 172.16.0.0/20

子网号同样是 0000（4 位），与设备 A 的子网号一致。

10.6.无分类域间路由（CIDR）

无分类域间路由（Classless Inter-Domain Routing，CIDR）划分子网在一定程度上缓解了 Internet 在发展中遇到的问题。

（1）CIDR 消除了传统 IP 地址的分类和划分子网的概念，可以更加有效地分配 IPv4 的地址空间。CIDR 把 32 位的 IP 地址划分为两个部分：前面的部分为网络前缀，用来指明网络；后面的部分用来表示主机。它的记法为在 IP 地址后加上斜线“1”，然后在后面写上网络前缀所占的位数，例如，128.2.3.4/20 表示网络前缀为高 20 位，主机号为低 12 位。这里容易混淆的是，网络号是 IP 地址中标识主网络的全局部分，由 CIDR 前缀长度决定（如 /19）。子网号是划分子网后，从原网络的主机位中借用部分位形成的，划分之后这个网络的 CIDR 表示法也会改变。看例题 2。

例题 1：IP 地址 10.10.33.66 /16 的网络地址是（__）。

/16 表示子网掩码为 255.255.0.0，即前 16 位（前两个字节）为网络位，后 16 位（后两个字节）为主机位。网络地址 ＝IP 地址 与 子网掩码（按位与操作）。得到网络地址为 10.10.0.0。

例题 3：某校园网的地址是 202.115.192.0/19，要把该网络分成 32 个子网，则子网掩码该是（__）

CIDR 允许进一步的子网划分。通过从主机号中借用位来创建子网号，可以将一个大的网络划分为多个小的子网。/19 表示网络地址是 19 位，主机地址是 13 位，若要划分 32 个子网，必须从这 13 位中借用 5 位作为子网位（2^5=32），所以网络位变为 19+5=24 位，因此子网掩码变为 /24，

也就是 255.255.255.0。

例题 3：路由器收到一个目标地址为 201.46.17.4 的数据包，应将该数据包发往（__）网。

A.201.46.0.0/21 B.201.46.16.0/20 C.201.46.8.0/22 D.201.46.20.0/22

当路由器收到一个数据包时，它会根据目标 IP 地址和路由表中的子网掩码来确定应该将数据包发送到哪个网络。具体来说，路由器会将目标 IP 地址与每个路由条目中的子网掩码进行按位与操作，得到一个网络地址，然后匹配路由表中对应的网络地址。如果有多个匹配项，路由器会选择最长前缀匹配（即子网掩码位数最多的那个）。

先将目的地址 201.46.17.4 的后两部分转化为二进制：201.46.0001 0001.0000 0100，然后再计算各个网口的对应的网络号（网络号通过子网掩码来确定）。网络号和目的地址与子网掩码计算的计算结果相同即位最后的转发的端口。如下表所示。最后的结果为 B。

选项	网络位位数	网络地址	各个网口的对应的网络号	目的地址与子网掩码计算
A	21	201.46.0000 0000.00000000	201/46/00000	201/46/0001
B	20	201.46.0001 0000.00000000	201/46/0001	
C	22	201.46.0000 1000.00000000	201/46/0000/10	
D	22	201.46.0001 0100.00000000	201/46/0001/01	

10.7.IPv4 VS IPv6（次重点★★★☆☆）

IPv6 通过更大的地址空间、简化报头、内置安全和自动配置等特性，解决了 IPv4 地址耗尽和扩展性不足的问题。其他维度的对比如下表所示，其中标记颜色的重点记忆。

对比维度	IPv4	IPv6
地址空间	32 位（约 43 亿个地址）	128 位（约 3.4×10^{38} 个地址）
地址表示	点分十进制（如：192.168.1.1）	冒号分隔十六进制（如：2001:0db8:85a3::8a2e:0370:7334）

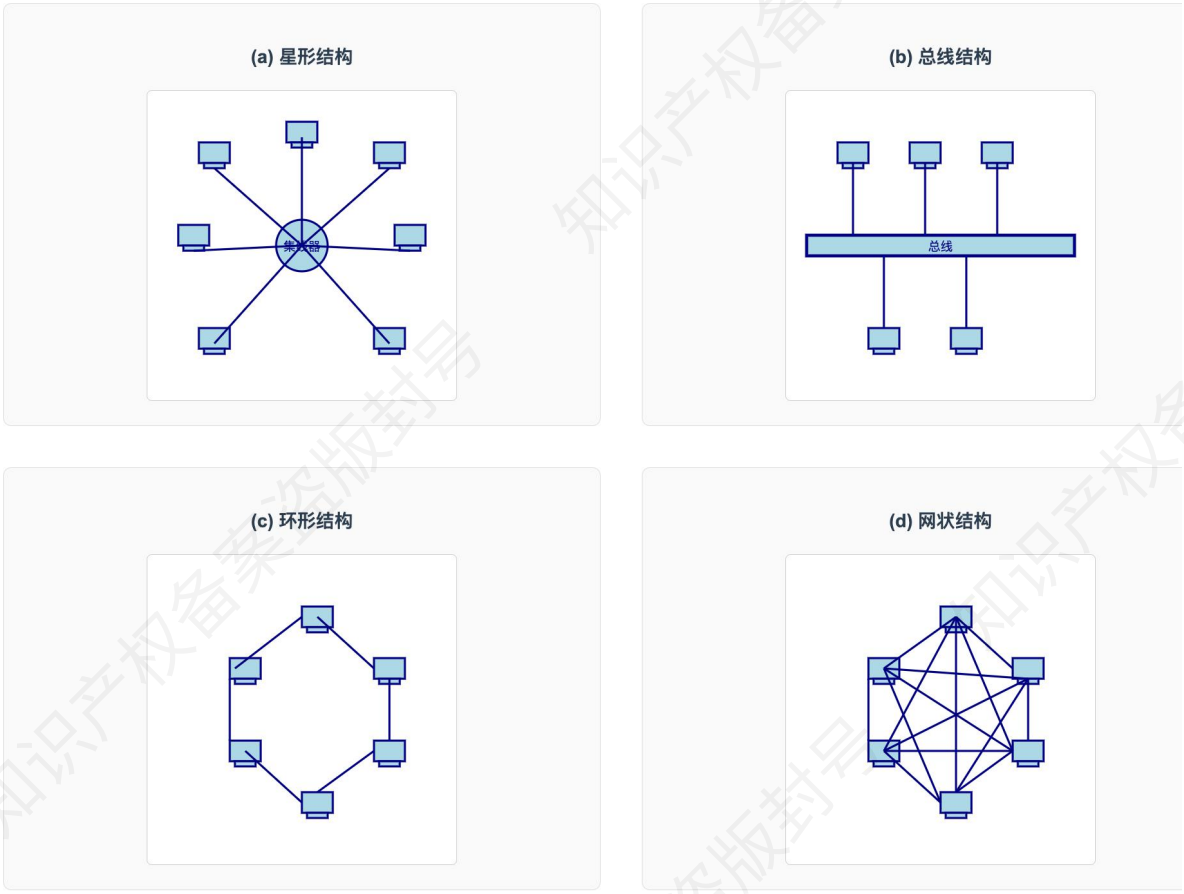
地址分配	手动配置 / DHCP（需公有 / 私有地址规划）	无状态地址自动配置（SLAAC）+ DHCPv6 可选
报头格式	可变长报头（20-60 字节），包含校验和、分片等字段	固定 40 字节报头，简化字段（无校验和，分片由源端处理）
NAT 依赖	依赖 NAT 解决地址短缺问题（如家庭网络共享公网 IP）	无需 NAT（地址充足，支持端到端通信）
安全性	无内置加密，需额外配置 IPsec	强制支持 IPsec（AH/ESP 协议）
路由效率	路由表庞大，处理复杂	路由表精简（聚合性更好），转发速度更快
应用场景	传统网络、低地址需求场景	物联网、大规模网络、云服务、未来互联网基础设施
过渡成本	已成熟，但面临地址枯竭问题	部署成本高（需升级设备 / 软件），但长期扩展性更好

10.8.局域网（次重点★★★☆☆）

局域网是一种覆盖范围较小（数百米至数公里）、传输速率高（可达千兆）、误码率低的计算机网络，通常以 PC 为主体，配合终端和外设构成，采用简单协议、结构灵活，建网成本低、周期短，便于管理与扩展。

10.8.1.局域网结构（重点★★★★★）

这一块内容多次出选择题进行考察，需要记忆下表的核心结构。



拓扑结构	核心结构	主要特点
星形结构	以一个枢纽（网络交换设备）为中心，所有节点均直接连接到该枢纽，形成星形布局。	结构直观，易于理解；依赖枢纽设备，枢纽故障可能影响整个网络。
总线结构	所有节点通过一条共享通信线路（如同轴电缆）连接，线路为非闭合结构。	共享通信线路，结构简单；线路故障可能导致整个网络瘫痪。
环形结构	所有节点通过共享通信线路连接成闭合环，每个节点仅与相邻两个节点直接相连。	与总线结构类似但线路闭合；消息传递需经过中间节点，单个节点故障可能影响环路通信。
网状结构	任何节点之间均通过独立物理通信线路直接相连，无中心枢纽。	节点故障不影响其他节点；但布线复杂、建设成本高、控制方法复杂，实际应用较少。

10.8.2.以太网核心技术（次重点★★★☆☆）

以太网（以太网技术在局域网领域占据主导地位，以太网几乎成为局域网的代名词）采用带冲突检测的载波监听多路访问技术，多个站点共享同一传输介质，每个站点在发送数据前先监听信道

状态，若信道空闲则发送数据，在发送过程中同时进行冲突检测，若检测到冲突则立即停止发送并等待一段时间后重试。这里有两个技术一个就是载波监听算法，一个就是冲突检测，如下表所示。

技术分类	具体技术	解释
载波监听算法	非坚持型监听算法	站点在发送数据前先监听信道，若信道空闲则立即发送；若信道忙，则随机等待一段时间后再监听，而不是持续监听信道状态。这种算法减少了站点间的冲突概率，但可能会增加信道的空闲时间，降低信道利用率。
	1-坚持型监听算法	站点在发送数据前先监听信道，若信道空闲则立即发送；若信道忙，则持续监听，直到信道空闲后立即发送数据。这种算法只要信道空闲就会马上发送数据，提高了信道的利用率，但当有多个站点同时监听到信道空闲时，容易产生冲突。
	P-坚持型监听算法	站点在发送数据前先监听信道，若信道空闲，则以概率 P 发送数据，以概率（1 - P）推迟到下一个时间片再监听；若信道忙，则持续监听，直到信道空闲后再按照上述概率规则决定是否发送数据。这种算法综合了非坚持型和 1 - 坚持型算法的特点，在一定程度上平衡了冲突概率和信道利用率。
冲突检测	边发边听	站点在发送数据的同时，持续监听信道上的信号。如果监听到的信号与自己发送的信号不一致，则说明发生了冲突。一旦检测到冲突，站点会立即停止发送数据，并发送一个拥塞信号通知其他站点，然后等待一段随机时间后重新尝试发送数据。

10.9.无线局域网（次重点★★★☆☆）

无线局域网（Wireless Local Area Network，简称 WLAN）是一种利用无线通信技术替代传统有线线缆，在局部区域（如家庭、办公室、校园、商场等）内实现设备互联和资源共享的计算机网络。它通过无线信号（如无线电波）传输数据，摆脱了物理线缆的限制，为用户提供灵活、便捷的网络接入方式。建议记住下面的 WIFI4~WIFI7 的速率（0.6/6/9/46Gbps）和标准（n/ac/ax/be）。

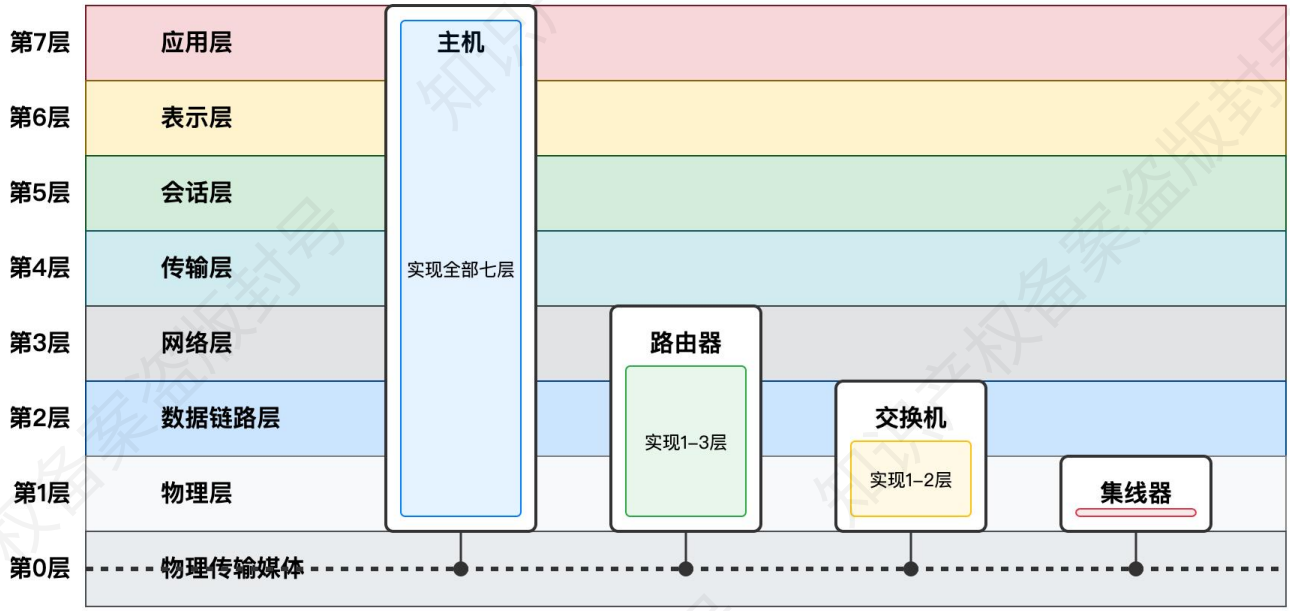
标准 / Wi-Fi 名称	发布年份	频段 (GHz)	最大理论速率
原始 802.11	1997	2.4 GHz	1–2 Mb/s
IEEE 802.11b (Wi-Fi 1)	1999	2.4 GHz	11 Mb/s
IEEE 802.11a (Wi-Fi 2)	1999	5 GHz	54 Mb/s
IEEE 802.11g (Wi-Fi 3)	2003	2.4 GHz	54 Mb/s
IEEE 802.11n (Wi-Fi 4)	~2009	2.4 GHz / 5 GHz	最高可达 600 Mb/s
IEEE 802.11ac (Wi-Fi 5)	~2013	5 GHz	最高可达 ≈6.9 Gbps（部分设备常标 1.3 Gbps）

IEEE 802.11ax (Wi-Fi 6)	~2019	2.4 GHz / 5 GHz（部分支持 6 GHz）	最高理论 9.6 Gbps
IEEE 802.11be (Wi-Fi 7)	2024	2.4 / 5 / 6 GHz	最高可达 ~46 Gbps

10.10.网络互连与常用设备（重点★★★★★）

这里选择题为主，主要需要辨析哪些设备工作在哪些层次上。

设备类型	工作层次	主要功能与特点
中继器	物理层	对接收信号进行再生和发送，扩展传输距离，对高层协议透明，使用个数有限（如以太网最多 4 个）
集线器		多端口中继器
网桥	数据链路层	根据 MAC 地址进行网络间信息转发，缓解网络通信繁忙，提高效率，连接相同 MAC 层的网络
二层交换机		传统意义上的交换机，多端口网桥
路由器	网络层	通过逻辑地址进行网络间信息转发，完成异构网络互连互通，连接使用相同网络层协议的子网
三层交换机		带路由功能的二层交换机
网关	高层（4 - 7 层）	最复杂的网络互联设备，连接网络层以上执行不同协议的子网



计算机网络中用户多、传输距离远时采用交换技术提高传输设备利用率、降低系统费用。

交换类型	描述/特点

电路交换	数据传送前先设通路，通路被一对用户独占
报文交换	存储转发，需缓冲存储和排队，不能满足实时通信
分组交换	类似报文交换，但报文分组，规定最大分组长度，分数据报分组交换和虚电路分组交换，是数据网络中使用最广泛的交换技术

路由器工作在网络层，是重要网络互连设备，主要功能是路由选择。路由选择协议按应用范围分解如下【选择题可能会考】：

路由选择协议	描述/特点
内部网关协议	如 RIP、OSPF、IGRP、EIGRP 等，在一个自治系统内运行
外部网关协议	如 BGP，在两个自治系统间使用，控制路由策略
核心网关协议	主干网中核心网关交换路由信息时使用

从使用算法看，路由协议分解如下【不太重要】

路由协议算法	描述/特点
距离向量协议	计算链路矢量和距离确定最佳路径，定期发送路由表
链路状态协议	创建拓扑数据库计算最短路径形成路由表，定期发送链路状态信息
平衡型协议	结合距离向量和链路状态协议优点

10.11.网络工程（次重点★★★☆☆）

网络工程是复杂的系统工程，分网络规划、网络设计和网络实施三个阶段。每个阶段做什么事情需要了解即可，对付选择题。

阶段	具体内容
网络规划	需求分析、可行性研究、对现有网络的分析与描述
网络设计	确定网络总体目标和设计原则，通信和资源子网设计、设备选型、网络操作系统与服务器资源设备选择、网络安全设计

网络实施	编制工程实施计划、网络设备到货验收、设备安装、系统测试、系统试运行、用户培训、系统转换

11.操作系统（次重点★★★☆☆）

操作系统这一章内容目前考察频率和难度逐渐下降，主要以概念和计算题为主，非重点，速读过一遍，应付选择题即可。

11.1.进程管理（次重点★★★☆☆）

我们编写的代码只是一个存储在硬盘的静态文件，通过编译后就会生成二进制可执行文件，当我们运行这个可执行文件后，它会被装载到内存中，接着 CPU 会执行程序中的每一条指令，那么这个运行中的程序，就被称为「进程」。

进程（Process）是计算机系统中正在运行的程序实例，具有一定的独立功能，并在某个数据集合上进行一次运行活动。它是操作系统进行资源分配的基本单位（调度的最小单位是线程），也是操作系统结构的基础。

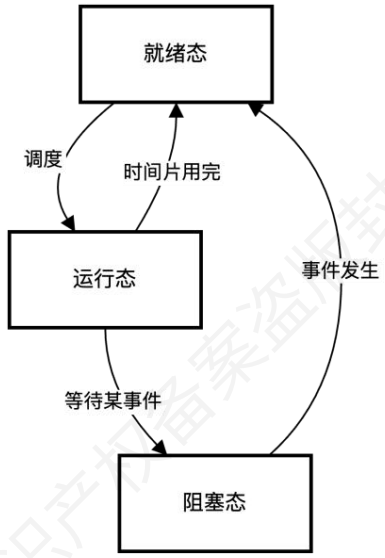
进程管理是操作系统中非常重要的一部分，涉及进程的控制、同步、调度和死锁等问题。这些管理活动确保了进程能够高效、安全地运行，避免资源冲突和系统崩溃。【这里的概念了解即可】

11.1.1.进程模型（次重点★★★☆☆）

一个进程从创建而产生至撤销而消亡的整个生命周期，可以用一组状态加以刻画。为了便于管理进程，把进程划分为三态模型和五态模型，架构系分常考的是三态模型。这里关键要知道就绪态和等待态是不一样的概念。就绪是在队列等待调度了。而等待或者阻塞态是这个进程暂停运行了。

三态模型		
状态名称	**状态描述**	**举例**

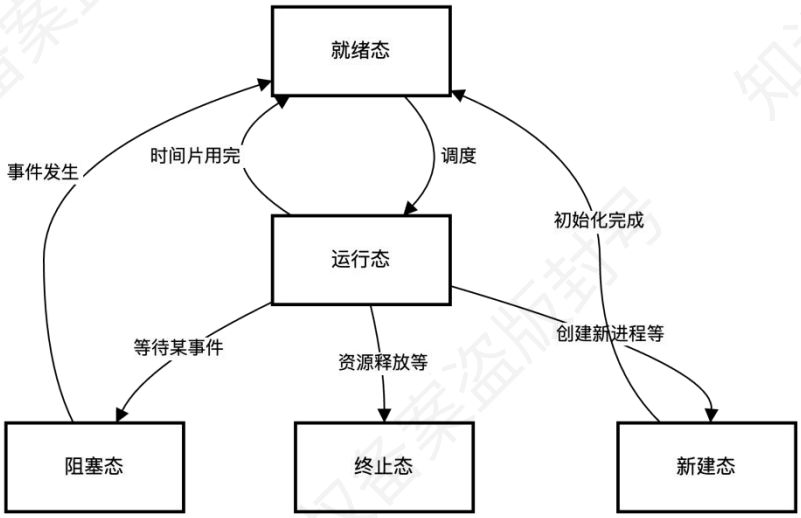
就绪态	进程已获取除 CPU 外的所有必需资源，处于随时可运行状态，但因 CPU 资源有限，需在就绪队列中等待系统分配处理器	多个等待运行的应用程序进程，已准备好内存、文件等资源，在就绪队列等待 CPU 调度
运行态	进程获得处理器资源，正在执行。同一时刻，单 CPU 系统中只有一个进程处于运行态	当前正在被 CPU 处理的浏览器进程
阻塞态	又称等待态或睡眠态。进程因等待某一事件（如 I/O 操作完成、等待特定资源等）而暂停运行，此时即便有空闲 CPU 也无法运行，需等到事件完成才会转为就绪态	向硬盘写入数据的进程，在写入操作完成前处于该状态



记忆口诀：舅（就绪）运（运行）祖（阻塞）宗

五态模型		
状态名称	**状态描述**	**举例**
新建态	进程刚被创建时所处状态，系统正在为其分配资源、进行初始化等操作，尚未进入就绪队列	新启动的应用程序，在完成一系列初始化设置前处于新建态
就绪态	进程准备好运行，等待 CPU 调度	多个已准备好资源但等待 CPU 分配时间片的应用程序进程
运行态	进程正占用 CPU 执行	正在被 CPU 处理的浏览器进程

阻塞态	进程因等待特定事件而暂停，不具备运行条件	向硬盘写入数据的进程，在写入操作完成前处于阻塞态
终止态	进程执行完毕正常结束，或因错误异常终止，又或被其他有权限的进程或操作系统终止。进入该状态后，进程不再执行，但系统会保留部分内核区数据用于善后处理，用户区数据通常被删除，一旦其他进程抽取完相关信息，操作系统就会彻底删除该进程	程序正常结束任务，或因代码错误崩溃

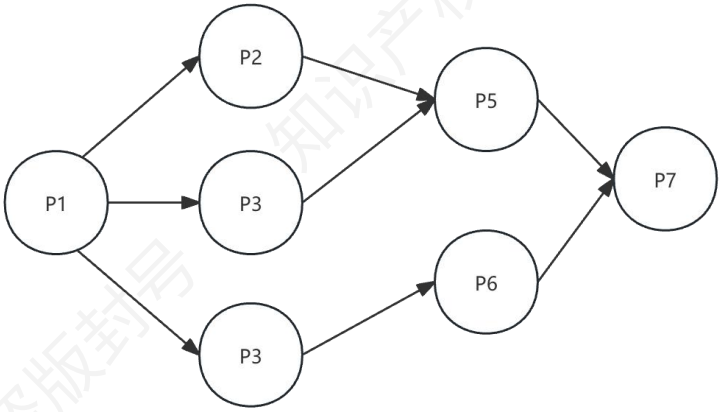


记忆口诀：新（新建态）舅（就绪）运（运行）祖（阻塞）宗（终止）

11.1.2.PV 操作（重点★★★★★）

11.1.2.1.前趋图（重点★★★★★）

这是下面 PV 操作解题的基础，需要掌握。在操作系统中，前趋图是一个有向无环图，用于描述进程之间的执行顺序。每个结点可以表示一个进程、程序段或一条语句，而结点间的有向边则表示两个结点之间存在的偏序或前驱关系。是各个活动之间的先后依赖关系。图下图所示，可以知道 P1→P2，P2→P5。那么能不能写成 P1→P5。答案是不能。前驱图的每一对结点必须是相邻关系，他们之间有直接联系。



11.1.2.2.同步与互斥（重点★★★★★）

同步与互斥选择题会考，案例倒是没有考过，参看下表凯恩总结的内容。

概念	定义	关键要点	举例
进程同步（按顺序）	在多个进程之间协调执行顺序，以保证它们按照既定的顺序共享资源	注重进程执行的先后顺序，通过协调使进程按特定顺序访问资源	读进程必须等写进程写满管道才开始读，写进程必须等读进程读完管道数据才开始写
进程互斥（独占）	不同进程间对某些关键资源进行独占访问，某一时刻只有一个进程可访问该资源，其他进程需等待资源释放	聚焦于资源的独占性，保证资源在同一时刻不被多个进程同时使用	在一个时间段内只允许一个进程使用的 “临界资源”，如打印机等

11.1.2.3.PV 操作（重点★★★★★）

PV 操作是一种实现进程同步与互斥的有效方法，它由两个原子操作组成，分别是 P（wait）操作和 V（signal）操作。PV 的词源是由荷兰语中的 Proberen（测试）和 Verhogen（增加）来的，对应到英文是 Wait 和 Signal。建议看一下我 B 站视频（芝士架构凯恩）操作系统相关的内容。这一

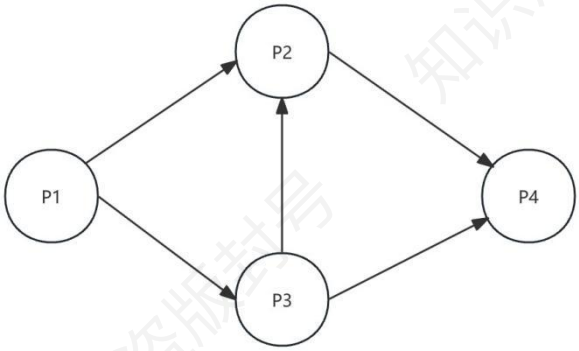
章节喜欢考选择题，一般 2-3 分，建议掌握。

操作	步骤	条件判断及处理	本质作用
P 操作	信号量 S 的值减 1	若 S-1 之后， S≥0，执行 P 操作的进程继续执行	申请资源，当资源足够时进程继续运行，资源不足时进程被阻塞
		若 S-1 之后， S<0，则该进程为阻塞状态，将其插入阻塞队列	
V 操作	信号量 S 的值加 1	若 S+1 之后，S>0，执行 V 操作的进程继续执行	释放资源且唤醒等待资源的进程
		若 S+1 之后，S≤0，从阻塞状态唤醒一个进程，将其插入就绪队列，然后执行 V 操作的进程继续	

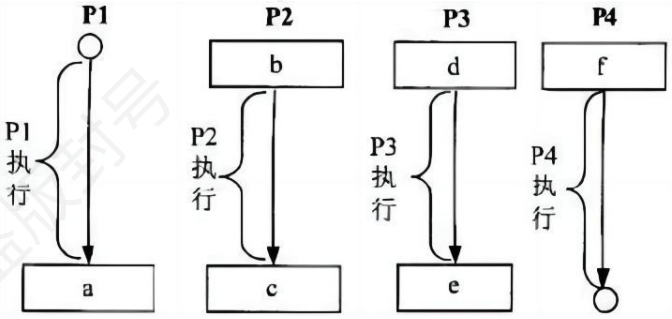
这个知识点有两种考法。第一告知临界资源数量，问你有多少个资源在进行等待。比如，系统中有 **1 个临界资源**，采用**计数信号量 S** 实现互斥与同步。某一时刻，系统中信号量的当前值为 **S = -3**。问：**此时有多少个进程处于等待（阻塞）该临界资源的状态？**

系统中采用计数信号量 S 来实现对临界资源的互斥与同步。计数信号量的值用于描述系统中可用资源数量及进程的等待情况。当信号量 S 的值大于或等于 0 时，表示系统中仍有可用的临界资源，其数值大小等于当前可用资源数；当信号量 S 的值小于 0 时，表示系统中不存在可用资源，且其绝对值表示因申请该资源而被阻塞的进程数。因此，当 S = -3 时，表示当前有 3 个进程处于等待该临界资源的阻塞状态。

再来来看一道例题，进程 P1、P2、P3 和 P4 的前趋图如下所示：



若用 PV 操作控制进程 P1～P4 并发执行的过程，则需要设置 5 个信号量 S1、S2、S3、S4 和 S5，且信号量 S1-S5 的初值都等于 0。下图中 a、b 和 c 处应分别填写（__）；d、e 和 f 处应分别填写（__）。



在信号量里我们知道，P 操作是占用临界资源，V 操作是释放资源（通知别人可以用）。看懂了继续往下。

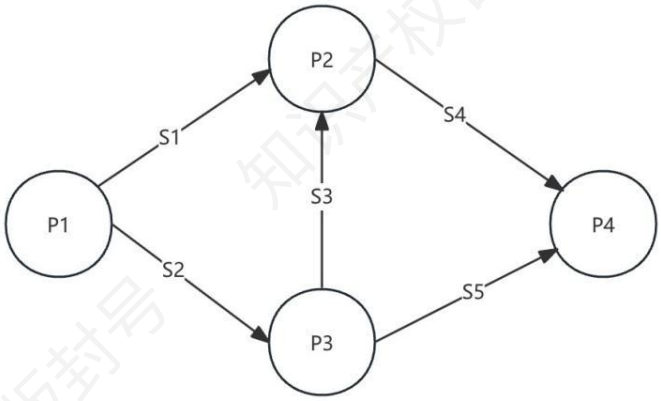
P1 是 P2 和 P3 的前驱进程，它们都要等 P1 执行完才能执行，所以在 P1 执行完的位置要有 V 操作来释放信号量，通知 P2 和 P3 可以执行了，所以 a 处为 V (S1)V(S2)。

P2 是 P1 和 P3 的后继进程，要等待它们执行完才能执行，所以要用 P 操作等待，b 处为 P (S1) P(S3)。执行完之后通知 P4 执行，所以 c 处为 V(S4)。

再看 P3，P3 是 P1 的后继进程，所以要等待 P1 执行完毕，所以 d 处为 P(S2)。P3 是 P2 和 P4 的前驱进程，它们都要等 P3 执行完才能执行，所以在 P3 执行完的位置要有 V 操作来释放信号量，通知 P2 和 P4 可以执行了，所以 e 处为 V (S3)V(S5)。

P4 是 P3 的后继进程，要等待它执行完才能执行，所以要用 P 操作等待，f 处为 P (S4) P(S5)。

看下图会比较清楚。



有没有简单做法？实际上真题中往往不需要判断信号量标记顺序。只要给出某个节点上要做 V 操作还是 P 操作就能快速的筛选出符合条件的答案。另外凯恩在视频里说的信号量标记顺序不一定在所有场景都适用。

11.1.3.死锁（重点★★★★★）

当若干个进程竞争使用资源时，如果每个进程都占有了一定的资源，又申请使用已被另一个进程占用且不能抢占的资源，则所有这些进程都纷纷进入阻塞状态，不能继续运行，即系统中两个或两个以上的进程无限期地等待永远不会发生的条件，系统处于一种停滞状态，这种现象就称为死锁。产生死锁的 4 个必要条件如下。这里会考选择题，下面的定义要掌握。

条件名称	定义
互斥条件	任一时刻只允许一个进程使用资源，难改变且应保证
不剥夺条件	进程已占用的资源不会被强制剥夺，实现复杂且有弊端
请求与保持条件	进程请求其余资源时，不主动释放已占有的资源，简单但可能资源浪费和进程延迟
环路条件	环路中每一条边是进程在请求另一个进程已占有的资源，主要缺点是资源分配序号问题和限制编程

那如何避免死锁呢？这就是死锁预防要做的事情。死锁预防针对破坏 4 个必要条件进行，包括互斥条件、不剥夺条件、请求与保持条件、环路条件。【选择题爱考】

11.2.内存管理（重点★★★★★）

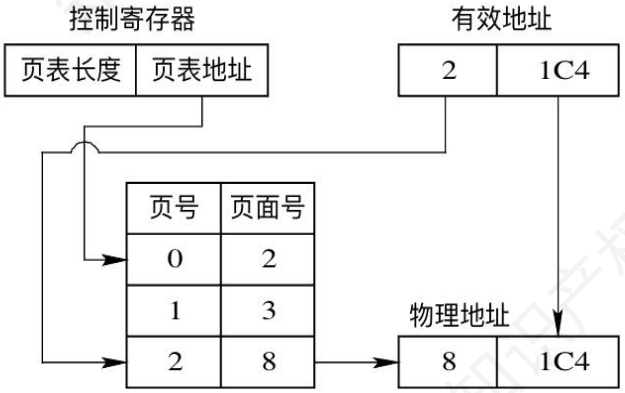
这一章节爱考选择题，凯恩在 B 站视频里讲得比较清楚。内存管理的基本原理是将系统的内存资源划分为多个逻辑空间，并在逻辑空间和物理内存之间建立映射关系，通过内存分配、内存回收等操作来管理内存资源的分配和释放。其主要目标是提高内存的利用率和效率，为系统提供高效、稳定地运行环境。

11.2.1.内存映射（重点★★★★★）

内存映射解决的问题：程序逻辑地址到物理地址的转换。这里主要介绍两种方式，页式管理和段式管理。

11.2.1.1.页式管理（重点★★★★★）

页式管理将主存空间和程序逻辑地址划分为固定大小的页面，并使用页表将虚拟地址与物理地址进行转换。每个页面在物理内存中占据一个连续的内存块，称为页帧。逻辑地址=页号+页内地址



例题：某计算机采用**页式存储管理**，逻辑地址空间大小为 **64KB**，页面大小为 **1KB**。已知页表如下所示：

页号	物理块号
0	5
1	9
2	10
3	7
4	3
5	8
6	6
7	2
8	1

物理内存的物理块大小同样为 1KB。问：

（1）逻辑地址用多少位表示？页号和页内地址各占多少位？

逻辑空间 =64KB= 2^{16}→ 逻辑地址共 16 位。

页面大小 =1KB =2^{10}字节 → 页内地址占 10 位。

页数 =64KB/1KB=64 → 需要 6 位表示页号。

逻辑地址 =[6 位页号 | 10 位页内地址]。

（2）给定逻辑地址 2010H (16 进制)，请计算它对应的物理地址。

逻辑地址 =2010H= 十进制 8208。

转换为二进制（16 位）：0010 0000 0001 0000。

前 6 位：001000= 页号 =8

后 10 位：0000010000= 页内地址 =16 (十进制)。

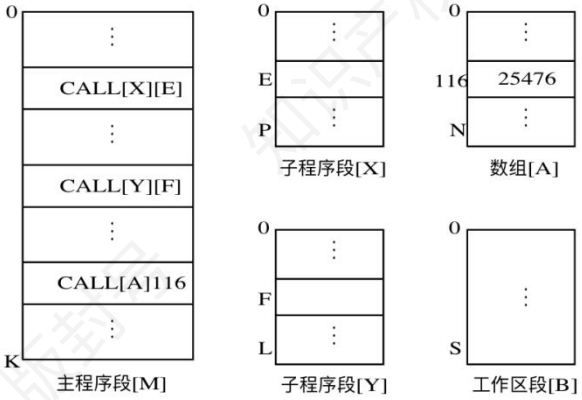
查页表：页号 =8 → 物理块号 =1。

物理地址计算：物理地址=物理块号×页大小+页内地址= 1×1024+16=1040（十进制）= 0410H（16 进制）。

11.2.1.2.段式管理（重点★★★★★）

段式管理把程序逻辑地址空间按段划分，每段是一个独立的逻辑实体，每个段分配一个连续的内存区，各段之间不要求连续。为了完成逻辑地址到物理地址的映射，需要查找段表。逻辑地址=

段号+段内地址。



11.2.2.虚拟内存（重点★★★★★）

虚拟存储管理使得程序运行前不必全装入内存，可先装入部分启动，其他留外存，根据需要调入内存，内存满时置换，可使大程序在小内存运行，多进程并发，系统具有请求调入和置换功能，从用户角度看内存容量扩充，这种系统称虚拟存储系统。内存无空闲页面需淘汰页面装入新程序数据，良好算法应淘汰访问概率低的页。常见的淘汰算法如下所示，需要关注的是 LRU 的相关概念。

算法名称	算法描述
随机算法	随机地从内存中的页面中选择一个进行淘汰。
轮转算法	按照一定的顺序（如页面在内存中的存储顺序）依次循环选择页面进行淘汰，就像一个环形队列，每次轮到的页面被淘汰。
先进先出（FIFO）算法	优先淘汰最早进入内存的页面。它把页面按进入内存的先后顺序排成队列，当需要淘汰页面时，选择队首的页面。
最近最久未使用（LRU）算法	当需要淘汰某一页时，选择离当前时间最近的一段时间内最久没有使用过的页先淘汰。它认为过去一段时间内久未使用的页面，在未来一段时间内也很可能不会被使用。
最近没有使用页面置换算法（NUR）	为内存中的每个页面设置一个访问位，定期检查这些页面的访问位。如果页面的访问位为 0（表示最近没有被访问过），则将其淘汰；如果访问位为 1（表示最近被访问过），则将访问位清零，让其留在内存中继续观察。
最优置换算法	淘汰在未来最长时间内不会被访问的页面。这是一种理想情况下的算法，在实际中很难实现，因为很难准确预知页面未来的访问情况，但可以作为衡量其他算法性能的一个参考标准。

时钟页面替换算法（Clock）	也叫循环缓冲区算法，它将内存中的页面组织成一个环形队列，类似于时钟的指针在环形队列上移动。每个页面都有一个访问位，当需要淘汰页面时，从当前指针位置开始扫描，若页面的访问位为 0，则淘汰该页面；若访问位为 1，则将其访问位清零并将指针移动到下一个页面，直到找到一个访问位为 0 的页面进行淘汰。

例题：如果页面的访问顺序为（0，0，1，1，3，1，2），有 2 个页帧可供程序使用，按照先进先出页面置换算法，共产生（__）缺页中断。

根据访问序列（0，0，1，1，3，1，2）并假设初始内存空：

首次访问 0 缺页装入 0；再次访问 0 命中；访问 1 缺页装入 1；

再次访问 1 命中；访问 3 缺页并按先进先出淘汰最早进入的 0，装入 3；

访问 1 命中（此时在内存）；访问 2 缺页并按 FIFO 淘汰最早进入的 1，装入 2。

全程共有 4 次缺页，分别发生在首次访问 0、首次访问 1、访问 3、访问 2。

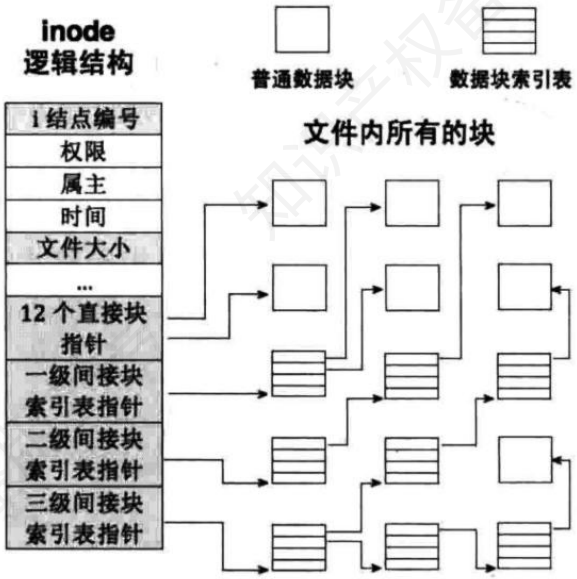
FIFO 关键在于淘汰最早驻留内存的页，与是否近期使用无关。

11.3.文件系统（次重点★★★☆☆）

这一章节只会出现在选择题里，你要关注例题就行。

11.3.1.文件索引（次重点★★★☆☆）

操作系统中文件索引分直接索引和间接索引，直接索引简单高效，适用于文件少且大小固定场景，查找速度快。间接索引节点存储指向其他索引节点指针，包括一级间接索引（指向单层索引表）和二级间接索引（通过一级索引找中间索引表再找数据块地址），适用于文件多或大小变化大的场景。

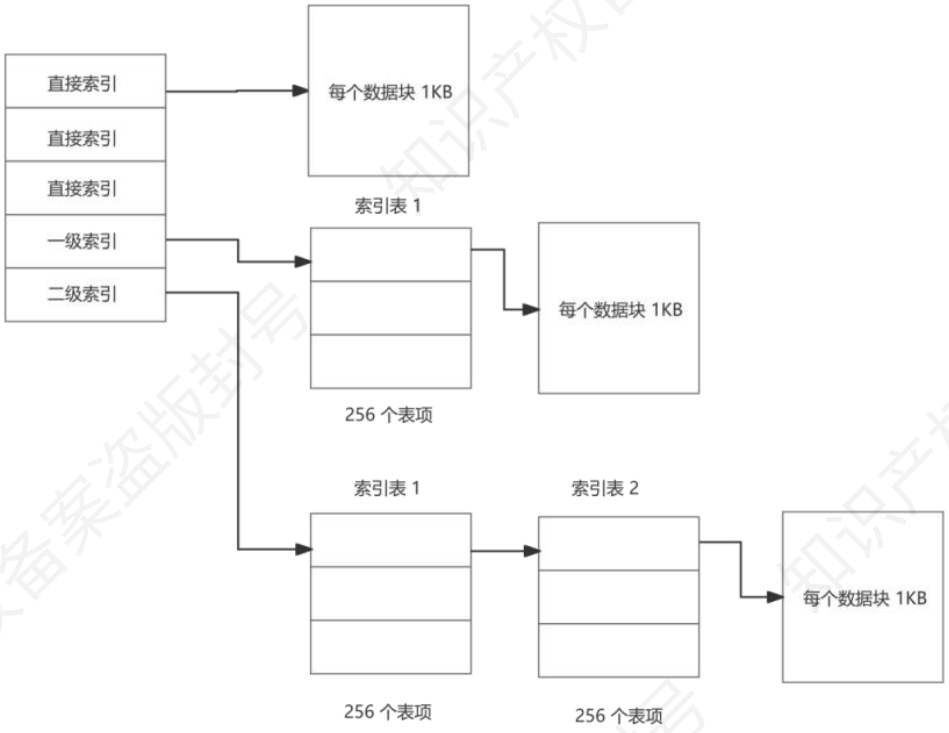


例题 1：某文件系统文件存储采用文件索引节点法。每个文件索引节点中有 8 个地址项，每个地址项大小为 4 字节,其中 5 个地址项为直接地址索引,2 个地址项是一级间接地址索引,1 个地址项是二级间接地址索引。磁盘索引块和磁盘数据块大小均为 1KB。若要访问 iclsClient.dll 文件的逻辑块号分别为 1、518，则系统应分别采用（__）

A.直接地址索引和直接地址索引　　B.直接地址索引和一级间接地址索引

C.直接地址索引和二级间接地址索　D.—级间接地址索引和二级间接地址索引

本题主要考查文件系统中不同逻辑块号对应的地址索引方式，关键在于理解直接地址索引、一级间接地址索引和二级间接地址索引的寻址范围。凯恩为此画了一个图。可以看到索引表实际上和一个数据块的大小是一样的。它能存放多少地址项是要根据数据块的大小除以地址项长度得到的。这里就是 1024 / 4 = 256 个。一个一级索引先索引到一个索引表，然后索引表的每个表项对应一个数据块，所以可以对应 256 个数据块。一个二级索引表，首先对应一个一级索引表，每一个一级索引表的表项再去对应一个索引表 2，每个索引表 2 的表项对应一个数据块。所以最多有 256*256 个，依次类推三级索引就是 256*256*256。



直接地址索引范围：已知每个文件索引节点中有 5 个直接地址索引，由于磁盘数据块大小为 1KB，每个逻辑块大小与磁盘数据块大小相同（通常情况），所以直接地址索引可表示的逻辑块号范围是 0 - 4（因为有 5 个直接地址项，从 0 开始计数）。

一级间接地址索引范围：每个地址项大小为 4 字节，磁盘索引块大小为 1KB（1024 字节），那么一个一级间接地址索引块能存放的地址项数量为 1024 / 4 = 256 个。所以两个一级间接地址索引可表示的逻辑块号范围是 5 - 516（512 个）。

二级间接地址索引范围：一个一级间接地址索引块能存放 256 个地址项，那么二级间接地址索引可表示的逻辑块号范围从 517 开始。

对于逻辑块号 1，因为 1 在 0 - 4 范围内，所以采用直接地址索引。

对于逻辑块号 518，因为 518 >517，所以采用二级间接地址索引。

11.3.2.磁盘空闲区管理（重点★★★★★）

常用的磁盘空闲区管理方法有空闲文件目录、空闲块链、位示图和成组链接法。位示图是利用

二进制的 1 位来表示文件存储空间中的 1 个块的使用情况。这是架构系分常考的知识点。

例题:假设某计算机的字长为 32 位,该计算机文件管理系统磁盘空间管理采用位示图(bitmap)记录磁盘的使用情况。若磁盘的容量为 300GB，物理块的大小为 4MB，那么位示图的大小为（__）个字。

若磁盘的容量为 300GB，物理块的大小为 4MB，那么该磁盘有 300 * 1024 / 4 = 76800 个物理块。根据题意系统字长为 32 位，所以一个字可记录 32 个物理块的使用情况。所需位示图的大小为 76800 / 32 = 2400 个字。所以答案为 2400。

12.数据库系统（超级重点★★★★★）

数据库系统是系统架构设计师和系统分析师常考知识点，超级重点毫无疑问。

选择案例常考数据库模式、规范化理论，并发控制，完整性，ER 图等。

案例部分现在也开始考察数据库分区，NOSQL，反规范化，主从复制，读写分离，一致性等常用的数据库设计维护运行技术，我在案例冲刺红宝书上都做了补充，需要重点关注。数据库系统这里的知识比较抽象，凯恩尽可能利用一些例子把事情说明白，我在下面提到的一些例子可以辅助你的理解，看的时候多加体会。系分架构都爱考常考，属于不看必挂系列。

论文考察不算太多，以当年凯恩押题为准。

12.1.数据库模式（重点★★★★★）

数据库系统的结构可以有多种不同的层次或不同的角度，其中典型的是三级划分法，其中包括三级模式和两级映射。记忆要点如下所示。这是纯数据库设计理论方面的，选择题爱考。

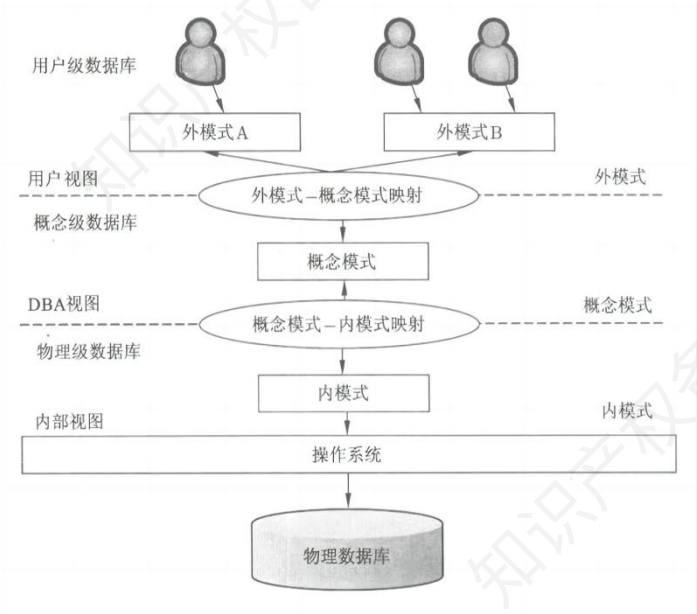
三级模式	面向人群	内涵

外模式	用户(程序员和终端用户)	描述组成用户视图的各个记录的组成、相互关系、数据项的特征、数据的安全性和完整性约束条件，是局部数据的逻辑结构和特征的描述
概念模式	数据库管理员	概念模式是数据库中全体数据的逻辑结构和特征的描述，用以描述现实世界中的实体及其性质与联系，定义记录、数据项、数据的完整性约束条件及记录之间的联系
内模式	系统程序员	用以描述存储记录的类型、存储域的表示和存储记录的物理顺序，以及索引和存储路径等数据的存储组织，是数据在数据库内部的表示方式

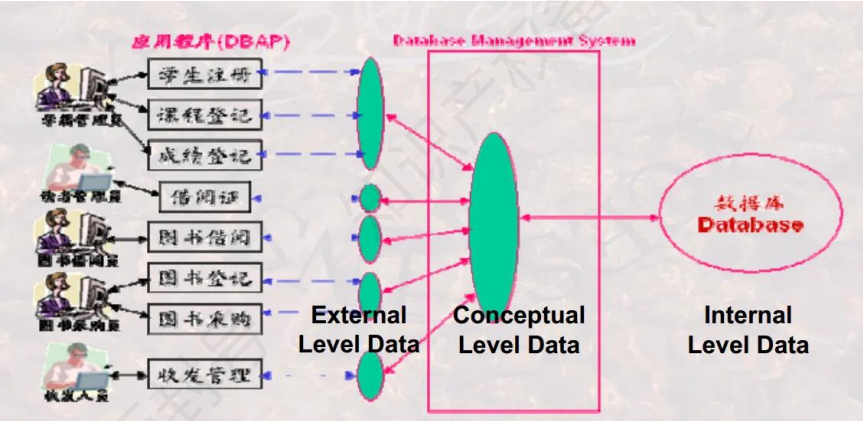
记忆口诀：外（外模式）局（局部数据）改（概念）全（全体）内（内模式）务（物理顺序）。外来的局长改了全部的内务。

两级独立性	内涵
物理独立性	用户的应用程序与存储在磁盘上的数据库中的数据是相互独立的
逻辑独立性	用户的应用程序与数据库中的逻辑结构是相互独立的，当数据的逻辑结构改变时，应用程序不需要改变

数据库系统的三级模式如下图所示，数据库系统由外模式、概念模式和内模式三级构成。



到这里还是很抽象，凯恩通过一个简单的例子来说明这三个层次之间的关系（主要为了辅助理解，不需要去背）。



这张图表示数据库管理系统（DBMS）从三个层次来管理数据：外部层次（External Level）、概念层次（Conceptual Level）和内部层次（Internal Level）。其中外部层次的数据是用户所看到的数据，所以又叫『用户』层次；概念层次的数据是 DBMS 中全局管理（可以简单地理解为所有的）数据，及数据之间的约束，所以又叫『逻辑』层次；内部层次的数据是存储在介质上的数据，包括存储路径、存储方式、索引方式等，所以又叫『物理』层次。而『模式』，就是『对于数据的结构性描述』，所以对于不同层次的数据，就对应着不同层次的模式，分别是：外模式（用户模式）、模式（概念模式/逻辑模式）与内模式（物理模式）。

12.2.关系模型（重点★★★★★）

关系模型包含元素如下所示，这里所有的名词都要理解。倒不是会出题目，而是因为下面的术语是你理解后面进一步概念的基础。

术语	描述
关系(表文件)	一个二维表，由行和列组成，对应数据库中的一张表。
元组（记录）	表中的一行，代表一个元组或一条记录。
属性（字段）	表中的每一列，定义了数据的意义和数据类型。
属性值	行和列交叉点的值，代表特定记录的特定属性的具体数据。
主码（主键）	用于唯一确定表中一个元组的数据，可以是一个或多个字段。

域	属性的取值范围，定义了字段可以存储的数据类型和可能的值。
关系模式	关系的逻辑描述，一般表示为：关系名（属性 1，属性 2，...，属性 n）
主属性	包含在任何一个候选键中的属性。
非主属性	不包含在任何候选键中的属性。

12.2.1.关系运算基础（次重点★★★☆☆）

这一章需要重点掌握。在选择题中时常考到。特别这里的关系运算符号一定要记住，有时候会给你一个关系代数让你写出等价的 SQL 表达式，所以看懂非常关键。关系代数的基本运算主要有并、交、差、笛卡尔积、选择、投影、连接和除法运算。下面的≡是等价于的意思。

（1）并。计算两个关系在集合理论上的并集，即给出关系 R 和 S （两者有相同元/列数） 的元组包括 R 和 S 所有元组的集合，形式定义如下，式中 t 是元组变量（下同）：

$$R \cup S \equiv \{t|t \in R \vee t \in S\}$$

R

A	B	C
1	1	1
1	2	1
2	3	3

S

A	B	C
5	2	0
1	2	2
1	1	1

R ∪ S

A	B	C
5	2	0
1	2	2
1	1	1
1	2	1
2	3	3

（2）差。计算两个关系的区别的集合，即给出关系 R 和 S （两者有相同元/列数），R-S 的元组包括 R 中有而 S 中没有的元组的集合，形式定义如下：

$$R - S \equiv \{t|t \in R \wedge t \notin S\}$$

R	S	上面是 R – S 下面是 S – R

A	B	C
1	1	1
1	2	1
2	3	3

A	B	C
5	2	0
1	2	2
1	1	1

A	B	C
1	2	1
2	3	3

A	B	C
5	2	0
1	2	2

（3）交。计算两个关系集合理论上的交集，即给出关系 R 和 S（两者有相同元/列数），$R \cap S$ 的元组包括 R 和 S 相同元组的集合，形式定义如下：

$$R \cap S \equiv \{t | t \in R \wedge t \in S\}$$

R	**S**	**R ∩ S**

A	B	C
1	1	1
1	2	1
2	3	3

A	B	C
5	2	0
1	2	2
1	1	1

A	B	C
1	1	1

（4）笛卡尔积。计算两个关系的笛卡尔乘积，令 R 为有 m 元的关系，S 为有 n 元的关系，则 $R \times S$ 是 m + n 元的元组的集合，其前 m 个元素来自 R 的一个元组，而后 n 个元素来自 S 的一个元组。形成定义如下：

$$R \times S \equiv \{t | t = < t_r, t_s > \wedge t_r \in R \wedge t_s \in S\}$$

若 R 有 u 个元组，S 有 v 个元组，则有 $u \times v$ 个元组。

R	**S**	**R × S**

A	B	C
1	1	1
1	2	1
2	3	3

C	D	E
5	2	0
1	2	2
1	1	1

R.A	R.B	R.C	S.C	S.D	S.E
1	1	1	5	2	0
1	1	1	1	2	2
1	1	1	1	1	1
1	2	1	5	2	0
1	2	1	1	2	2
1	2	1	1	1	1
2	3	3	5	2	0
2	3	3	1	2	2
2	3	3	1	1	1

（5）投影。从一个关系中抽取指明的属性（列），就是 SQL 中 SELECT 指定的列名。假设有一个包含属性 A 的关系，则

$$\pi_A(R) \equiv \{t[A] | t \in R\}$$

（6）选择。从关系中抽取出满足给定限制条件的记录，就是 SQL 中 WHERE 后面带的条件记作，其中 F 表示选择条件，是一个逻辑表达式（逻辑运算符+算术表达式）。

$$\sigma_F(R) \equiv \{t | t \in R \wedge F(t) = true\}$$

（7）θ 连接。可以把它看作是先做笛卡尔积然后筛选出满足条件的元组，如果两个关系中进行比较的分量是相同的属性组，并且在结果中将重复的属性去掉，则称为自然连接，记作：

$$R \rhd\lhd S \equiv \{t_r, t_s | t_r \in R \wedge t_s \in S \wedge t_r[C] = t_s[C]\}$$

R

A	B	C
1	1	1
1	2	1
2	3	3

S

C	D	E
5	2	0
1	2	2
1	1	1

R ⋈ S

A	B	C	D	E
1	1	1	2	2
1	1	1	1	1
1	2	1	2	2
1	2	1	1	1

除法不展开讨论比较复杂毫无意义。考到就放弃。

12.2.2.元组演算基础（次重点★★★☆☆）

元组演算这里只会考选择题，并且最多 1 分，因为太理论了，实在看不懂可以跳过，凯恩在视频里举过若干例子。元组演算其实就是 SQL 的前身。在元组演算中，元组演算表达式简称为元组表达式，其一般形式为其中，t 是元组变量，表示一个元数固定的元组；P 是公式，在数理逻辑中也称为谓词，也就是计算机语言中的条件表达式。$\{t|P(t)\}$表示满足公式 P 的所有元组 t 的集合。在元组表达式中，公式由原子公式组成，原子公式有下列两种形式：

(1) R(s)，其中 R 是关系名，s 是元组变量。其含义是 s 是关系的一个元组。

(2) $s[i]\theta u[j]$，其中 s 和 u 是元组变量，θ是算术比较运算符，$s[i]$和 $u[j]$分别是 s 的第 i 个分量和 u 的第 j 个分量。原子公式 $s[i]\theta u[j]$表示“元组 s 的第 i 个分量与元组 u 的第 j 个分量之间满足θ运算”。例如，t[2]<u [3]表示元组 t 的第 2 个分量小于元组 u 的第 3 个分量。这个原子公式的一种简化形式是 $s[i]\theta a$，其中 a 为常量 。例如，t[4]=3 表示 t 的第 4 个分量等于 3。

同时，原子可用通过逻辑算子∧(与)、∨(或)和¬(非)组合成公式，而且我们可以使用存在量词(∃)和全称量词(∀)来绑定变量。

是不是很抽象？没事，实际考题不会很深，你只要掌握下面的四个式子就可以对付过去。假设我们有下面两个表，一个 R 和 S，给出的下面四个元组演算表达式，你要掌握。我在下面的视频里重点讲解了【系统架构设计师/系统分析师-数据库系统-元组演算】

R	S

A	B	C
1	2	3
4	5	6
7	8	9
10	11	12

A	B	C
3	7	11
4	5	6
5	9	12
6	10	14

下面四种关系搞懂已经毕业，不懂看一下 B 站芝士架构凯恩的视频

R1={t|R(t)∧¬S(t)}，找出在 R 中，但是不在 S 中的元组。

A	B	C
1	2	3
7	8	9
10	11	12

R2={t|S(t)∧t[3]>t[2]∧t[2]<8},找出在 S 中，且第三列分量大于第二列分量且第二列分量小于 8 的元组。

A	B	C
3	7	11
4	5	6

R3={t|(∃u)(R(t)∧S(u)∧t[3]<u[2])}读作找出关系 R 中在第三列分量上小于任意在 S 中的元组在第二列分量的所有元组。

A	B	C
1	2	3
4	5	6
7	8	9

R4={t|(∀u)(R(t)∧S(u)∧t[3]>u[1])}找出在关系 R 中的，且在第三列分量上大于所有 S 中元组在第一列分量上的所有元组。

A	B	C
7	8	9
10	11	12

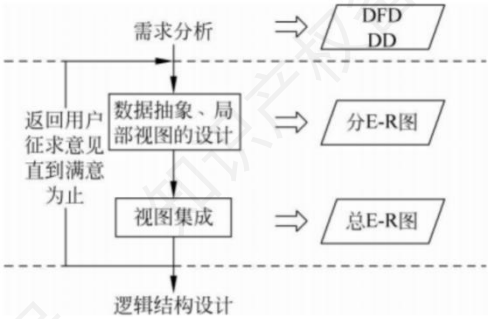
12.3.数据库设计与建模（超级重点★★★★★）

基于数据库系统生命周期的数据库设计在系分教材上可分为如下 5 个阶段：规划、需求分析、概念设计、逻辑设计和物理设计。有的教材写 6 个阶段，需求分析、概念设计、逻辑设计、物理设计、数据库实施和数据库运行和维护阶段。所以这里凯恩建议主要关注四块内容，既需求分析、概念设计、逻辑设计、物理设计阶段，其中概要设计、逻辑设计是最重要，其他不重要。

阶段	主要内容	具体任务
需求分析	设计者需了解和分析用户需求，双方密切合作明确系统总体设计方案，是整个设计过程的基础，结果影响后续阶段实施效率	编写需求说明书，包括数据流图、建立数据字典
概念设计	将需求分析得到的用户需求建立抽象的信息模型（概念模型），是现实世界的模型，便于与不熟悉计算机的用户交流，是数据库设计的关键	选择局部应用、逐一设计分 ER 图和 ER 图合并
逻辑设计	用具体 DBMS 实现用户需求，将概念结构转换为相应数据模型，根据用户处理要求、安全性考虑建立必要视图并优化数据模型	数据模型设计、E - R 图转换为关系模式、关系模式规范化、确定完整性约束、确定用户视图、反规范化设计
物理设计	为给定的逻辑数据模型选择高效、最适合的物理结构（主要指数据库的存储结构和存储方法），并对物理结构进行时间和空间效率方面的评价	

12.4.概念设计（超级重点★★★★★）

概念设计常用策略是运用自顶向下方法进行需求分析，然后运用自底向上方法设计概念结构。下图体现了自底向上方法设计概念结构方法，可以看到首先设计出局部 E-R 模型，然后将各个局部 E-R 图合并起来形成全局 E-R 模型，最后将全局 E-R 模型进行优化，评审后得到最终的 E-R 模型，即概念模型。

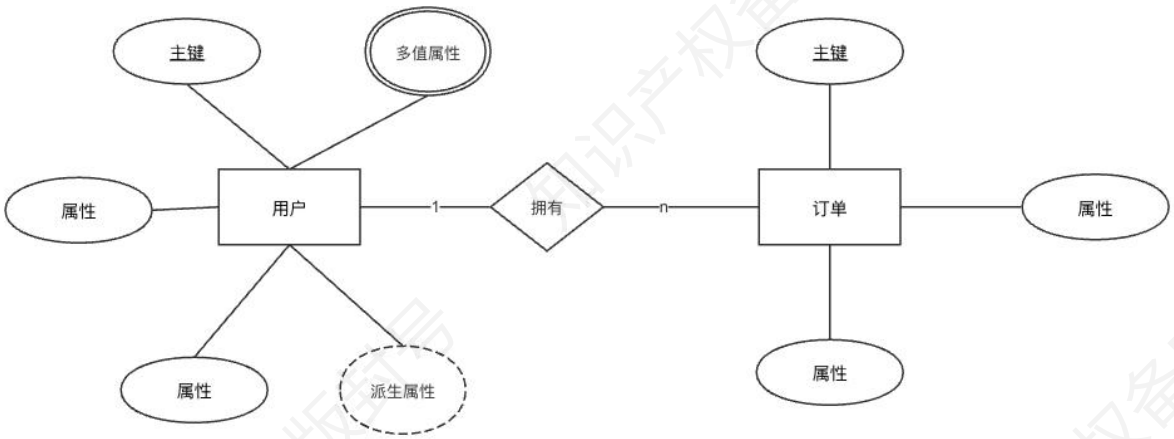


E-R 图的设计要依照上述的抽象机制，对需求分析阶段所得到的数据进行分类、聚集和概括，确定实体、属性和联系。概念结构设计工作步骤包括：选择局部应用、逐一设计分 E-R 图和 E-R 图合并。此章节属于选择案例重点考查的内容，特别是联系的类型，联系的转换必须熟练掌握，凯恩这里举了大量的例子辅助你理解。

E-R 模型也称为 E-R 图，它是描述概念世界，建立概念模型的实用工具。E-R 图三要素如下：

元素		表示方式	示例说明
实体（型）		用矩形框表示，框内标注实体名称	例如电商系统中的 “用户”“商品” 等实体，会分别用矩形框表示，框内写对应名称
属性	单值属性	用椭圆形表示，并用连线与实体连接起来	假设 “用户” 实体有 “姓名” 属性，“姓名” 就用椭圆形表示并与 “用户” 实体矩形框相连
	多值属性	用双实线椭圆表示，多值属性可有一个或两个以上的值	如学员信息数据库中，“学员” 实体的 “个人兴趣” 属性，可能有运动、电影等多个值，“个人兴趣” 用双实线椭圆表示并与 “学员” 实体相连
	派生属性	用虚线椭圆表示，从基本属性计算得出	像学员的 “总成绩”“平均成绩”，从基本成绩属性计算得出，用虚线椭圆表示并与 “学员” 实体相连
实体之间的联系		用菱形框表示，框内标注联系名称，用连线将菱形框与有关实体相连，并在连线上注明联系类型（1:1，1:n 或 m:n）	电商系统中，“用户” 与 “商品” 实体间可能存在 “购买” 联系，用菱形框表示 “购买”，分别与 “用户”“商品” 实体矩形框相连，连线上注明如 “m:n” 的联系类型

下图就是某电商系统的一个 E-R 图。



单单掌握上面的考点还不够，下面这些尚未考察的点，同样值得关注。

概念	定义	示例
弱实体	依赖于某个实体而存在的实体，依赖的对象为强实体	订单依赖于用户实体，订单是弱实体，用户实体是强实体
弱关系	一般与弱实体一起使用，仅弱实体才会涉及的关系	/
不完全概化	父实体的实例可以是子实体的实例，也可能不是任何子实体的实例，即父实体部分实例不属于任何子类别	职业作为父实体，其子实体为工程师、老师，存在部分职业不属于工程师和老师类别
全部概化	父实体的所有实例必须是某个子实体的实例，不存在独立的父实体实例，所有父实体实例都能归类到子实体中	人作为父实体，子实体为男人、女人，所有人必定属于男人或女人类别

12.4.1.整体 E-R 图设计（重点★★★★★）

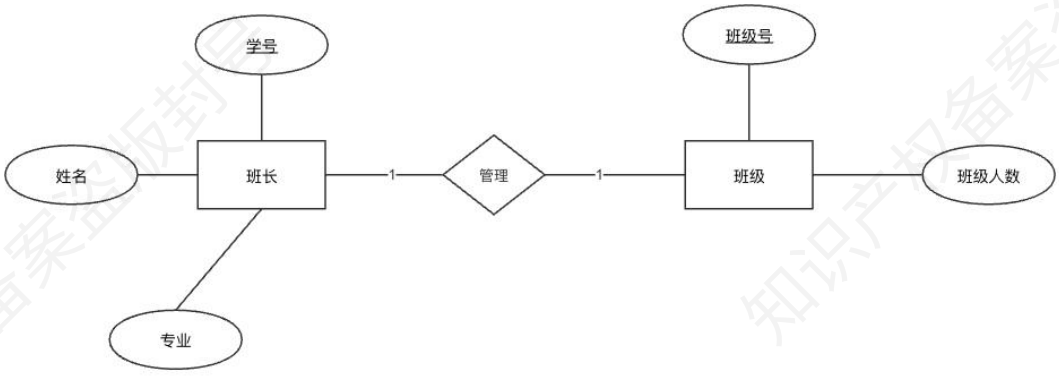
设计各子系统的局部 E-R 图设计理论过程是（1）确定局部视图的范围；（2）识别实体及其标识，确定实体之间的联系；（3）分配实体及联系的属性。各子系统的局部 E-R 图设计好后，下一步就是要将所有的分 E-R 图综合成一个系统的总体 E-R 图。这里仍然只会出选择题，真题掌握即可（记忆点——合并、消除冲突（属性/命名/结构）、消除冗余）。

步骤	内容		
合并	确定各局部 E-R 图的公共实体类型，以其为单位合并，直至所有相同实体类型合并完毕，得到全局 E-R 图		
消除冲突	因各子系统应用场景及设计人员不同，局部 E-R 图存在不一致问题即冲突，需消除冲突形成统一概念模型	属性冲突	包括属性域冲突和取值冲突，需各部门协商解决
		命名冲突	包含同名异义和异名同义，通过讨论、协商等行政手段解决
		结构冲突	同一对象在不同应用中抽象不同；同一实体在不同局部 E-R 图中属性个数或排列次序不同；实体间联系在局部 E-R 图中联系类型不同
消除冗余	冗余分为冗余属性（可由基本数据导出的数据）和冗余联系（可由其他联系导出的联系）		

12.5.逻辑设计（超级重点★★★★★）

逻辑结构设计的任务就是把概念结构设计阶段完成的整体 E-R 模型转换为与选用的 DBMS 产品所支持的数据模型相符合的逻辑结构。进行数据库的逻辑设计，要将 E-R 模型向关系模型转换（可以理解为转换成表的列的设计，下面我就直接说表结构转换方便理解），范式转换也发生在这里。系分可能出案例题，架构只会出选择题。这里直接看比较抽象，先去看一下我的 B 站视频（搜索芝士架构凯恩）。

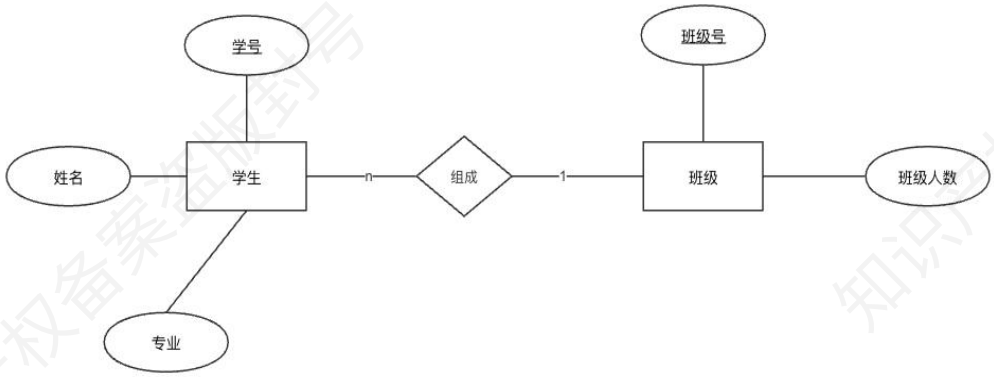
12.5.1.E-R 图转换关系模式（重点★★★★★）



这样一个 1:1 联系可以通过两种方法转换成符合范式的表结构。

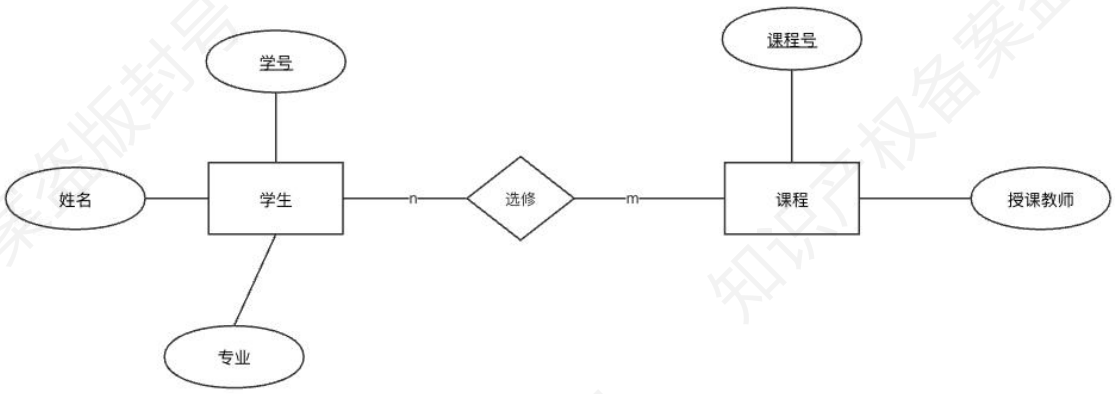
名称	转换方式	示例说明

转换为独立关系模式	关系的属性由联系相连各实体的主码以及联系本身的属性组成，主码是每个实体的主码	班长（学号，姓名，专业） 班级（班级号，班级人数） 管理（学号，班级号）
与联系一端实体的关系模式合并	将另一个实体的主码和联系本身的属性加入到该端实体关系模式的属性中	班长（学号，姓名，专业，班级号） 班级（班级号，班级人数）



这样一个 1:n 联系可以通过两种方法转换成符合范式的表结构。

名称	转换方式	示例说明
转换为独立关系模式	关系的属性由与该联系相连的各实体的主码以及联系本身的属性转换而来，主码是 n 端实体的主码	学生（学号，姓名，专业） 班级（班级号，班级人数） 组成（学号，班级号）
与 n 端实体对应关系模式合并	将联系本身的属性和 1 端实体的主码加入 n 端对应关系模式中	学生（学号，姓名，专业，班级号） 班级（班级号，班级人数）



一个 m：n 联系可以通过一种方法转换成符合范式的表结构。关系的属性由与该联系相连的各实体的主码以及联系本身的属性转换而来，各实体主码的组合是该关系的主码或关系主码的一部分。其

实说白话就是把关系独立出来单独成表。

学生（学号，姓名，专业）

课程（课程号，授课教师）

选修（学号，课程号）

12.5.2.函数依赖（重点★★★★★）

不用记定义，直接看下面例子。

例如：设一个职工关系为（职工号，姓名，性别，年龄，职务）。

解析：职工号函数决定姓名，或姓名函数依赖于职工号，记作“职工号→姓名”，职工号为该函数依赖的决定因素。

函数依赖会怎么考？直接给你例子让你分析。比如架构 2016 年第 30 题。



此题第一问就是考察函数依赖的定义。就是选出关系中谁决定谁。对于第一题直接使用代入法，由上表所示，由于 A1 为 a 时，A2 可能是 a 或 b 或 c，所以可以确定 A1→A2 不成立，所以它不是 R 的函数依赖。同理带入法可知 A1→A3 不成立。因此 ACD 三个选项均可排除，它们都不是 R 的函数依赖。假如继续细分，函数依赖还可以分为多种类型，见下表。其中部分函数依赖

和传递函数依赖特别关键，后面讲到范式的时候会提到。

函数依赖类型	定义	例子
平凡函数依赖	若 X→Y，但 Y⊆X，则称 X→Y 是平凡函数依赖	在职工关系 (职工号，姓名，性别，年龄，职务) 中，(职工号，性别)→职工号，因为职工号是 (职工号，性别) 的子集，所以是平凡函数依赖；(职工号，性别)→性别同理
非平凡函数依赖	若 X→Y，且 Y⊈X，则称 X→Y 是非平凡函数依赖	在职工关系 (职工号，姓名，性别，年龄，职务) 中，(职工号，姓名)→性别，性别不是 (职工号，姓名) 的子集，所以是非平凡函数依赖；(职工号，姓名)→(年龄，职务) 同理
完全函数依赖	在关系模式 R (U) 中，如果 X→Y，并且对于 X 的任何一个真子集 X'，都有 X'!→Y，则称 Y 对 X 完全函数依赖	在教师任课关系（教工号，姓名，职称，课程号，课程名，课时数，课时费）中，（职称，课程号）完全函数决定课时费，即只有（职称，课程号）能决定课时费，单独的职称或课程号都不能决定课时费
部分函数依赖	在关系模式 R (U) 中，如果 X→Y，但 Y 不完全函数依赖于 X，则称 Y 对 X 部分函数依赖	在教师任课关系（教工号，姓名，职称，课程号，课程名，课时数，课时费）中，（教工号，课程号）部分函数决定姓名，因为教工号本身就可以决定姓名，课程号在这里是多余的，即存在教工号→姓名，所以（教工号，课程号）是部分函数决定姓名；（教工号，课程号）部分函数决定课时数同理
传递函数依赖	在关系模式 R (U) 中，如果 X→Y，Y→Z，且 Y!→X，Z⊈Y，则称 Z 对 X 传递函数依赖	在学生关系（学号，姓名，性别，系号，系名，系主任名）中，学号→系号，系号→系名和系号→系主任名，且系号不能决定学号，系名和系主任名不属于系号，所以系名和系主任名传递依赖于学号

6.5.3.7.Armstrong 公理（次重点★★★☆☆）

从已知的一些函数依赖，可以推导出另外一些函数依赖，这就需要一系列推理规则。函数依赖的推理规则最早出现在 1974 年 W.W.Armstrong 的论文里，这些规则常被称作“Armstrong 公理”，不必记忆，了解就行。记忆方法也非常简单，就是顾名思义，比如传递合并分解你观察的公式的形态就可以记住。这里重点关注自反、增广、传递、合并和分解。

名称	公式
自反性	若 X⊇Y，则存在 X→Y
增广性	若 X→Y，则存在 XZ→YZ
传递性	若 X→Y 和 Y→Z，则存在 X→Z

合并性	若 X→Y 和 X→Z，则存在 X→YZ
分解性	若 X→Y，且 Y⊇Z，则存在 X→Z
伪传递性	若 X→Y 和 WY→Z，则存在 WX→Z
复合性	若 X→Y 和 Z→W ，则存在 XZ→YW
自增性	若 X→Y，则存在 WX→Y

凯恩这里举一个例子：假设我们有一个学生关系（学号，姓名，性别，年龄，所在专业）。

根据增广性规则，若学号→所在专业成立，则（学号，性别）→（所在专业，性别）也必然成立。

根据合并性规则，若学号→姓名和学号→性别成立，则学号→（姓名，性别）也成立。

根据分解性规则，学号→（姓名，性别）也成立，学号→姓名和学号→性别也同样成立。

12.5.3.无损联接分解（次重点★★★☆☆）

这里特别爱考选择题，所以凯恩标注了次重点。无损连接是指分解后的关系通过自然连接可以恢复成原来的关系，即通过自然连接得到的关系与原来的关系相比，既不多出信息、又不丢失信息。

无损联接分解的形式定义如下：设 R 是一个关系模式，F 是 R 上的函数依赖集。R 分解成数据库模式$\delta = \{R_1, \ldots, R_k\}$，如果对 R 中每个满足 F 的关系 r 都有下式成立：

$$r = \pi_{R1}(r) \bowtie \pi_{R2}(r) \bowtie \ldots \bowtie \pi_{Rk}(r)$$

则称分解 S 相对于 F 是无损联接分解，否则称为损失联接分解。这里的⋈表示自然连接。这一段是废话不需要记忆，可以理解直接看下面公式法

下面是一个很有用的无损联接分解判定定理，凯恩建议必须掌握：

设 $\rho = \{R_1, R_2\}$是 R 的一个分解，F 是 R 上的函数依赖集。那么分解ρ相对于 F 是无损级联分解的充要条件$(R_1 \cap R_2) \rightarrow (R_1 - R_2)$或$(R_1 \cap R_2) \rightarrow (R_2 - R_1)$。要注意的是，这两个条件只要有任意一个条件成立就可以了。具体做做真题或者看一下我的视频就能学会。

无损分解的公式法学到这里就可以了，有多个分解的情况更加复杂（表格法）不做展开，为了 1

分不值得，出到题目直接放弃。

12.5.4.保持函数依赖（次重点★★★☆☆）

这里特别爱考选择题，所以凯恩标注了次重点，一般和无损连接一起考。所谓的保持函数依赖就是将分解之后的函数依赖的集合合并起来,只要合并之后,与原来的函数依赖集合是保持等价的，我们就会说是保持函数依赖的。无损和函数依赖没有关系，可能一个分解是无损的但是没有保持函数依赖，反之亦然。

设数据库模式$\delta = \{R_1, \ldots, R_K\}$是关系模式 R 的一个分解，F 是 R 上的函数依赖集，δ中每个模式R_i上的函数依赖集是F_i。如果$\{F_1, \ldots, F_k\}$与 F 是等价的（即相互逻辑蕴涵），则称分解δ保持函数依赖。如果分解不能保持函数依赖，则δ的实例上的值就可能有违反函数依赖的现象。这里出过选择题，给你分解前模式和分解后的模式让你判断是否逻辑蕴含。这个通过作图和推理法都可以做，一般不会很难，做真题的时候看解析再学。

这里牵扯到一个逻辑蕴含的概念。设 F 是关系 R（U）中的一个函数依赖集合，X、Y 是 R 的属性子集，如果能从 F 这个函数依赖集合中推导出 $X \rightarrow Y$，则称 F 逻辑蕴含 $X \rightarrow Y$，或者说 $X \rightarrow Y$ 是 F 的逻辑蕴含。记作 $F \models XY$ 。F 是集合一定要清楚。集合意味着里面有 $X \rightarrow Y$，$X \rightarrow Z$ 的等等依赖关系。

12.5.5.规范化意义（重点★★★★★）

此章节极大概率在案例中进行考察，凯恩建议熟记这里的不规范化导致的 4 大异常情况（冗删改查）以及例子。一旦在案例中出现至少 8 分!

设有一个关系模式

R（SNO，SNAME, CNAME , TNO, TNAME, TADDRESS）

其属性分别表示学生编号、学生姓名、课程编号、课程名、任课教师姓名和任课教师地址。仔

细分析一下，就会发现这个模式存在下列异常，这些异常需要掌握，可能在案例中进行考察，系分同学要格外关注：

问题类型	描述	示例
数据冗余	相同数据在关系中多次重复出现	某课程有 100 个学生选修，课程的任课教师姓名和地址在关系 R 中重复出现 100 次
修改异常	因数据冗余，修改数据时需多处修改，否则会导致数据不一致	修改教师地址时，需修改 100 个元组中的地址值，不然会出现地址值不一致
插入异常	因部分信息缺失，导致相关数据无法正常插入数据库	不知道听课学生名单时，教师的任课情况和家庭地址无法进入数据库，或需在学生姓名处插入空值
删除异常	删除某类数据时，会意外删除其他不应删除的数据	删除某门课程的任课教师信息时，学生信息也被删除

记忆口诀：绒绣茶山。毛绒绣出来的茶山玩具。绒（冗余）绣（修改）茶（插入）山（删除）。

12.5.6.键/码/属性（重点★★★★★）

下面的键/码/属性是讨论范式的基础，特别是主属性和非主属性。这个在 BCNF 范式的讨论中会再次强调。

概念	定义
主键 / 码	被数据库设计者选中，用于在同一实体集中区分不同实体的候选码，应选从不或极少变化的属性。一个实体集仅有一个主码
超码	一个或多个属性的集合，能在实体集中唯一标识一个实体，可能含多余属性。可唯一标识实体，但可能存在冗余属性
候选码	若超码的任一真子集不能成为超码，则为候选码，不包含多余属性
主属性	包含在任一候选码中的属性
非主属性	不包含在任一候选码中的属性

这里引入一种求候选键的快捷方法（这是必须掌握，选择题遇到就是送分），即图示法。使用

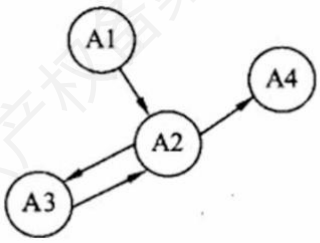
图示法求候选键，主要有两个步骤：

（1）将关系模式的函数依赖关系，用有向图的方式表示，其中顶点表示属性，弧表示属性之间的依赖关系。

（2）找出入度为 0 的属性集，并以该属性集为起点，尝试遍历有向图，若能正常遍历图中所有结点，则该属性集即为关系模式的候选键；若入度为 0 的属性集不能遍历图中所有结点，则需要尝试性地将一些中间顶点（既有入度，也有出度的顶点）并到入度为 0 的属性集中（真题中出现了这种情况），直至该集合能遍历所有顶点，则该集合为候选键。

下面这道题必会，给定关系 R（A1, A2, A3, A4）上的函数依赖集

$F = \{A1 \rightarrow A2, A3 \rightarrow A2, A2 \rightarrow A3, A2 \rightarrow A4\}$，现在要求 R 的候选键。需要针对函数依赖集画出有向图，如下图所示。这里的难点是可能有的题目会出现 A2，$A1 \rightarrow A4$ 的函数依赖，作图的时候你可以用特殊标记把他们标记出来（比如虚线，或者线段加粗共享一个箭头，防止搞错）。



从图 5-2 中找出入度为 0 的顶点，即 A1。通过尝试，可以发现从出发可以遍历所有顶点，因此，R 的候选键为 A1。

12.5.7.五大范式（超级重点★★★★★）

这块内容系分考生假如不明白不用去考了，案例必挂。

1NF、2NF、3NF、BCNF、4NF 全部在考试中出现过，选择案例都会考到，凯恩建议你一定要引起足够的重视，要学会判别和语言描述。（记忆：一范式，无重复，二范式，主键独，三范式，无传递，BCNF 传递无，四范式，多值除）。在判断一个关系属于哪种范式之前，一定要先找出它

的主键！然后再进行分析。这里要严格注意，24 年系分案例已经出现范式判断和 BCNF 规范化内容！死记硬背也要记住 1NF、2NF、3NF、BCNF 的内涵和规范化方法，凯恩这里每个范式都举了例子，你一定要学会，并默写出来。

目前共定义了多个范式，分别为 1NF、2NF、3NF、BCNF、4NF 和 5NF。（23 年架构考试已经出现 BCNF、4NF），总结如下表所示（暂时看不懂不要慌不要急，下面有详细的讲解）。

范式名称	定义	判断关键	示例及规范化方法
第一范式（1NF）	所有属性只包含原子值，每个分量不可再分	属性是否为原子属性	教师职称情况关系中“高级职称人数”非原子属性，拆分为 “教授” 和 “副教授” 属性可满足 1NF
第二范式（2NF）	满足 1NF，且不存在非主属性对候选码的部分函数依赖；若每一个候选码都是单码则也满足	是否存在非主属性对候选码的部分函数依赖	学生关系 （学号，姓名，性别，所在专业，课程号，课程名，成绩） 中姓名等部分函数依赖于 （学号，课程号），分解为学生 （学号，姓名，性别，所在专业）、课程 （课程号，课程名）、选课 （学号，课程号，成绩） 满足 2NF
第三范式（3NF）	满足 1NF，不存在非主属性对候选码的传递函数依赖；非主属性既不依赖也不传递依赖于任何候选码；不存在非主属性的关系模式一定为 3NF	是否存在非主属性对候选码的传递函数依赖	学生关系 （学号，姓名，年龄，学校编号，学校地址，学校电话） 存在传递函数依赖（学号->学校编号，学校编号->学校地址，学校电话），可分为学生（学号，姓名，年龄，学校编号）、学校 （学校编号，地址，电话） 或学生 （学号，姓名，年龄）、学校 （学校编号，地址，电话）、管理 （学号，学校编号） 满足 3NF
BCNF	满足 1NF，不存在任何属性对候选码的传递函数依赖，关系模式中任何函数依赖的左侧必须是码	函数依赖左侧是否为码	授课 （教工号，学号，课程号） 中存在教工号→课程号且教工号不是码，不是 BCNF 范式，分解为（学号，教工号）和（教工号，课程号）满足 BCNF
第四范式（4NF）	是 BCNF 的推广，针对有多值依赖的关系模型，需将一个表中多个多值依赖拆分开	是否存在多个多值依赖	职工表（职工编号，职工孩子姓名，职工选修课程）存在两个多值依赖，分为职工表 1（职工编号，职工孩子姓名）和职工表 2（职工编号，职工选修课程）满足 4NF

（1）第一范式（1NF）。在关系模式中，当且仅当所有属性只包含原子值，即每个分量都是不可再分的数据项，则称满足 1NF。例如，下表所示的教师职称情况关系就不满足 1NF。原因在于，该关系模式中的“高级职称人数”不是一个原子属性。将其拆分为“教授”和 “副教授”两个属性，则就满足 1NF 。

系名称	高级职称人数	
	教授	副教授
计算机系	6	10
电子系	3	5

（2）第二范式（2NF）。设一个关系为 R(U)，满足第一范式，若 R 中不存在非主属性对候选码的部分函数依赖，则称该关系是符合第二范式的，即 R∈2NF。推论：若关系模式 R∈1NF，且它的每一个候选码都是单码，则 R∈2NF。

看下面例子感受一下 2NF 的规范化方法。

假定存在学生关系（学号，姓名，性别，所在专业，课程号，课程名，成绩），使其符合 2NF。

姓名，性别，所在专业部分函数依赖于（学号，课程号）中的学号，因此符合第一范式但不符合第二范式，需分解为三个关系。

由于这个学生实体和课程实体是多对多关系，根据前面多对多关系 ER 图到关系模式转换办法我们知道，就是把联系单独提出来形成一个关系。

学生（学号，姓名，性别，所在专业）。

课程（课程号，课程名）。

选课（学号，课程号，成绩）。

经过分解，三个关系可以连接后仍得到原关系，且为无损分解和无损连接。

(3) 第三范式（3NF ）。设一个关系为 R(U)，满足第一范式，若 R 中不存在非主属性对候选码的传递函数依赖，则称该关系是符合第三范式的，即 R∈3NF。

推论 1：如果关系模式 R∈1NF，且它的每一个非主属性既不部分依赖，也不传递依赖于任何候选码，则 R∈3NF。

推论 2：不存在非主属性的关系模式一定为 3NF（所有属性都是主属性）。

有人说凯恩你写错了！第三范式是基于第二范式的，你怎么写了满足第一范式？其实不的，这是个推论，需要推导，推导过程很简单，通过反证法证明。

假设一个关系不满足 2NF，形式化表达如下：（A，B，C，D），其中（A，B）→(C，D)。假设存在 B→C（非主属性对码的部分依赖），因为（A，B）→B（平凡依赖），（A，B）→B，B→C，意味着存在 C 对码（A，B）的传递依赖。所以非 2NF 一定是非 3NF。那么这个结论的等价结果（逆否命题）就是 3NF 一定是 2NF。

看下面例子感受一下 3NF 的规范化方法。

假定学生关系（学号，姓名，年龄，学校编号，学校地址，学校电话），使其符合 3NF。

因为（学号）→（姓名，年龄，学校编号，学院地点，学院电话）和（学号）→（学校编号）→（学校地址，学校电话）存在非主属性对候选码的传递函数依赖。

这里有两个联系分别是学生实体和学校，他们之间是一对多关系，一个学校对应多个学生，一个学生对应一个学校。实体根据前面一对多联系转换成关系模式的方法，我们有两种办法来做规范化。

要将学生关系表分为如下关系的两个表。

学生（学号，姓名，年龄，学校编号）。

学校（学校编号，地址，电话）。

也可以分为三个表，把关系单独拿出来成表。

学生（学号，姓名，年龄）。

学校（学校编号，地址，电话）。

管理（学号，学校编号）。

（1）BCNF。如果对于关系模式 R 中存在的任意一个非平凡函数依赖 X->A，都满足 X 是 R 的一个超键，那么关系模式 R 就属于 BCNF。这个书本写得很抽象。所以直接看例子理解。

根据定义一个简单的判断方法是在 BCNF 范式中，关系模式中任何函数依赖的左侧必须是码（候选码），不能存在非码的属性。

假设有关系模式 R(A，B，C)，候选码为 AB 和 AC，函数依赖有 AB→C，以及 AC→B，在这两个函数依赖中，左边的和都是候选码（AB，AC），所以关系模式属于 BCNF 范式。

说明关系模式授课（教工号，学号，课程号）是否属于 BCNF 范式。

注意这里是有前提的：每名教师只教授一门课程，每门课程由若干名教师教授；一个学生可以选择多门课程；若某一学生选定某门课程，会被分配该课程的一名教师；同样地，某名学生选修了某名教师就确定了所选课程的名称。所以有下面的推论。

（教工号，学号），（学号，课程号）是候选码

但是由于存在教工号→课程号。

推论判断：因为教工号不是码（是主属性），所以不满足 BCNF 范式。

（如何规范化）可将授课分解为两个关系模式，即（学号，教工号）和（教工号，课程号），它们之中不存在任何属性对候选码的部分函数依赖和传递函数依赖，所以符合 BCNF 范式。

（5）第四范式（4NF）。第四范式是 BCNF 的推广，是针对有多值依赖的关系模型所定义的规范化形式。把一个表中多个多值依赖拆分开来。这里做不形式化的表达，单单讲不满足 4NF 的例子，以及如何分解，举个例子好理解一点。

职工表（职工编号，职工孩子姓名，职工选修课程）

在这个表中，同一个职工可能会有多个职工孩子姓名。同样，同一个职工也可能会有多个职工选修课程。由于存在两个多值依赖，就会导致表中数据冗余或者插入异常等其他问题。即这里存在

着多值事实，不符合第四范式。

如果要符合第四范式，只需要将上表分为两个表，使它们只有一个多值事实，例如职工表 1（职工编号，职工孩子姓名），职工表 2（职工编号，职工选修课程），两个表都只有一个多值事实，所以符合第四范式。

12.6.数据库的控制功能（重点★★★★★）

这一章节常考概念，系分容易出案例题，架构一般不会。要想使数据库中的数据达到应用的要求，必须对其进行各种控制，这就是 DBMS 的控制功能，包括并发控制、性能优化、数据完整性和安全性，以及数据备份与恢复等问题。这些技术虽然给人们的感觉是边缘性技术，但对 DBMS 的应用而言，却是至关重要的。

12.6.1.并发控制（重点★★★★★）

在多用户共享系统中，许多事务可能同时对同一数据进行操作，称为并发操作。此时，DBMS 的并发控制子系统负责协调并发事务的执行，保证数据库的完整性不受破坏，同时，避免用户得到不正确的数据。

12.6.2.事务的基本概念（重点★★★★★）

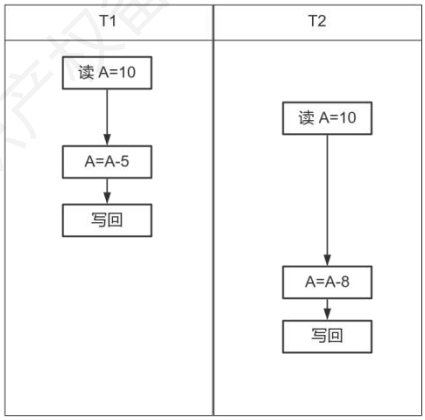
DBMS 运行的基本工作单位是事务，事务是用户定义的一个数据库操作序列，这些操作序列要么全做，要么全不做，是一个不可分割的工作单位。事务具有以下特性：（ACID 目前案例还没考过，这里的原理需要记住，可能出概念题，至少 8 分）

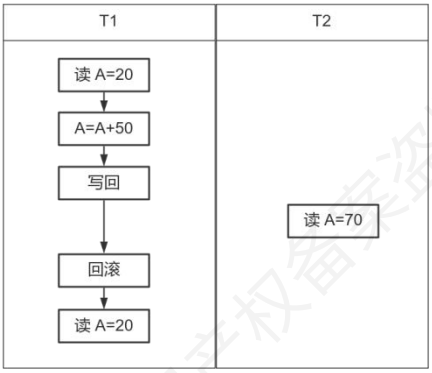
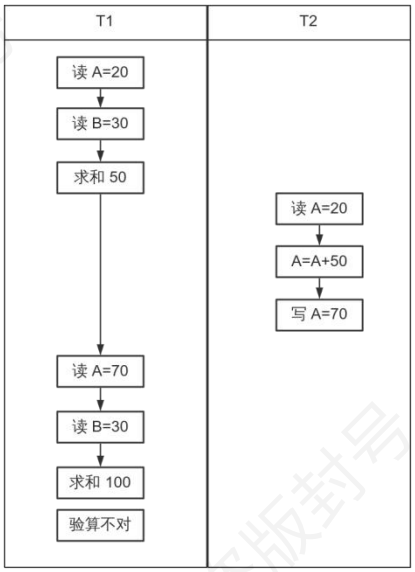
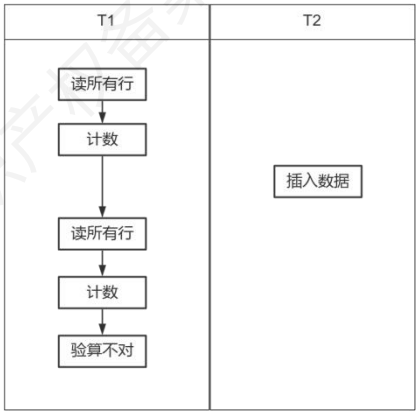
特性名称	含义	示例
原子性（Atomicity）	事务是数据库的逻辑工作单位，事务包含的一组更新操作是原子不可分的，是一个整体，不能部分地完成	强调事务中的操作要么全部执行成功，要么全部不执行

一致性（Consistency）	使数据库从一个一致性状态变到另一个一致性状态	转账操作中，各账户金额必须平衡。与原子性密切相关，由事务的隔离性来表示，逻辑上不独立
隔离性（Isolation）	一个事务的执行不能被其他事务干扰，一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰，即使多个事务并发执行，看上去也像每个事务按串行调度执行一样，也称为可串行性	强调并发事务之间的相互隔离，不互相干扰
持久性（Durability）	也称为永久性，事务一旦提交，改变就是永久性的，无论发生何种故障，都不应该对其有任何影响	比如订单提交成功后，相关数据的改变会永久保存，不会因故障等原因丢失

12.6.3.数据一致性问题（重点★★★★★）

数据不一致问题以往系分案例考过填空题，有概率改成案例概念题，这里的例子和三个不一致的场景必须掌握。数据库的并发操作会带来一些数据不一致问题，例如，丢失修改、读“脏数据”和不可重复读等。实际上可以概括为“写-写”并发场景和“读-写”并发场景。有的同学对事务隔离的意义不太理解。实际上一个事务是独立的，事务会在计算过程对同一个表反复读写。假如不做数据隔离，表根据外部的插入修改进行变化的话，那么在计算过程中会发生不可预计的错误。比如原本我已经读取到 100 个行了，我要对这些行做循环迭代 100 次并进行删除操作，假如这个过程中，外部把这个表变动了 1 行，变成了 99 行，那么迭代过程就会出错，诸如此类。

问题类型	描述	示例（结合下面的表格来看）	
丢失修改（写覆盖）	事务 A 与事务 B 从数据库中读入同一数据并修改，事务 B 的提交结果破坏了事务 A 提交的结果，导致事务 A 的修改被丢失。这里又可以细分为第一类更新丢失问题：A 事务回滚时，把已经提交的 B 事务的更新数据覆盖了。第二类更新丢失问题：A 事务覆盖 B 事务已经提交的数据，造成 B 事务所做操作丢失。	有 T1、T2 两个事务，执行顺序为：T1 读 A；T2 读 A；T1 执行 A＝A - 5，写回；T2 执行 A＝A - 8，写回。则“③A＝A - 5，写回” 操作会被“ A＝A - 8，写回” 操作覆盖掉，“③A＝A - 5，写回”将不起任何作用。	

读“脏数据”（读回滚）	事务 A 修改某一数据，并将其写回磁盘，事务 B 读取同一数据后，事务 A 由于某种原因被撤销，这时事务 A 已修改过的数据恢复原值，事务 B 读到的数据就与数据库中的数据不一致，是不正确的数据，称为“脏数据”。	有 T1、T2 两个事务，执行顺序为：T1 读 A；T1 将 A 修改为新值并写回；T2 读 A = 新值；T1 撤销，A 恢复原值。则 T2 中“读 A = 新值”就是读的脏数据，比如假设初始 A 为 20，T1 改为 70，T2 读 A = 70，T1 撤销后 A 变回 20，T2 读的 70 就是脏数据。	
不可重复读（读更新）	事务 A 读取数据后，事务 B 执行了更新操作，事务 A 前后两次读取结果就发生变化，造成了数据不一致性。	有 T1、T2 两个事务，执行顺序为：T1 读 A 并进行运算；T2 对 A 进行更新操作并写回；T1 再次读 A 并进行运算，然后验算，发现两次运算结果不同。因为 T1 第一次读 A 后，T2 修改了 A 的值，T1 第二次读 A 时用的更新后的值，导致验算结果不正确。	
幻读（读插入）	T1 读取某个范围的数据，T2 在这个范围内插入新的数据，T1 再次读取这个范围的数据，此时读取的结果和第一次读取的结果不同。本质上也属于不可重复读的情况。只不过是插入导致的。	例如 T1 读取年龄在 20 - 30 岁之间的学生记录，T2 插入了一条年龄为 25 岁的学生记录，T1 再次读取年龄在 20 - 30 岁之间的学生记录时，结果比第一次多了一条记录。	

12.6.4.封锁技术（重点★★★★★）

数据库事务锁很有可能在案例中出现，可能结合 MySQL 事务隔离机制出填空或者概念题。处理并发控制的主要方法是采用封锁技术，主要有两种封锁，分别是 X 封锁和 S 封锁。

封锁类型	含义	特点	操作权限	解除封锁要求
排他型封锁（X 封锁）	事务 T 对数据对象 A 实现 X 封锁	只允许一个事务独锁某个数据，具有排他性	事务 T 可读取和修改数据 A；其他事务在 T 解除 X 封锁前,不能对数据 A 进行任何类型的封锁	事务 T 完成对数据 A 的操作后解除
共享型封锁（S 封锁）	事务 T 对数据 A 实现 S 封锁	允许并发读，但不允许修改	事务 T 只能读取数据 A，不能修改；在所有 S 封锁解除前，任何事务不能对数据 A 实现 X 封锁	所有对数据 A 加 S 封锁的事务完成读取操作后解除

虽然只有两种锁，但是结合不同的场景就会有不同的功效，在多个事务并发执行的系统中，主要采取封锁协议来进行处理，常见的封锁协议如下：这里大致了解就行，基本不会做深入考察。

封锁协议	具体内容	能解决的问题	不能解决的问题
一级封锁协议	事务 T 在修改数据 R 之前必须先对其加 X 锁，直到事务结束才释放	防止丢失修改，保证事务 T 可恢复	不能保证可重复读和不读 “脏数据”，存在事务 T1 读数据 → 事务 T2 加 X 锁并改数据释放锁 → 事务 T1 再读数据导致的问题
二级封锁协议	一级封锁协议基础上，事务 T 在读取数据 R 之前先对其加 S 锁，读完后即可释放 S 锁	防止丢失修改、防止读 “脏数据”	不能保证可重复读
三级封锁协议	一级封锁协议基础上，事务 T 在读取数据 R 之前先对其加 S 锁，直到事务结束才释放	防止丢失修改、读“脏数据”，能保证可重复读	无

知识产权备案盗版封号

两段锁协议	所有事务必须分两个阶段对数据项加锁和解锁，扩展阶段：对任何数据进行读、写操作之前，首先要申请并获得对该数据的封锁；收缩阶段：释放一个封锁之后，事务不能再申请和获得任何其他封锁	若并发执行的所有事务均遵守，则任何并发调度策略都是可串行化的	遵守该协议的事务可能发生死锁

这里需要解释的是一级封锁协议。事务 T 在修改数据 R 之前必须先对其加 X 锁，直到事务结束才释放。一级封锁协议可防止丢失修改，并保证事务 T 是可恢复的，但不能保证可重复读和不读“脏数据”。这里同学可能会有疑问，不是说 X 是独占的吗！加了锁不就串行序列化了吗，不会有可重复和脏数据啊。这里要注意，有这样一种情况，事务 T1 读数据 → 事务 T2 加 X 锁并改数据释放锁 → 事务 T1 再读数据。换句话说，T2 加 X 锁的时候不能够判断是否已经有事务在读了，这个情况会导致问题，这是一级封锁协议的关键问题。

显然，使用封锁技术来解决并发控制问题，存在一个封锁粒度问题。所谓封锁粒度，是指被封锁数据对象的大小，这里记住结论，封锁粒度小则并发性高，但开销大；封锁粒度大则并发性低但开销小，综合平衡照顾不同需求，以合理选取适当的封锁粒度是很重要的。

12.7.数据库性能优化（次重点★★★☆☆）

这里极有可能出现案例题，让你写出常见的数据库优化手段。

12.7.1.反规范化（重点★★★★★）

从某种意义上来说，非规范化（反规范化）可以改善系统的性能。在进行数据库设计时，可以考虑合理增加冗余属性，以提升系统性能。书本知识点归纳如下。

类别	详情
反规范化	将常用的计算属性（如总计、最大值等）存储到数据库实体中

提升性能的措施	重新定义实体，以减少外部属性数据或行数据的开支
	将关系进行水平或垂直分割，以提升并行访问度
反规范化带来的问题	数据冗余增加，相同数据在多个地方重复存储
	更新异常，因数据冗余，更新一处数据可能遗漏其他重复存储处，导致数据不一致
	插入异常，为满足反规范化结构要求，可能需先插入不必要数据
	删除异常，删除某些数据可能意外删除其他有用信息

12.7.2.索引优化（重点★★★★★）

索引相关知识也是案例常考内容，也是书本上的内容，它的使用准则建议熟记，会在案例中让你默写概念。索引是提高数据库查询速度的利器，而数据库查询往往又是数据库系统中最频繁的操作，因此，索引的建立与选择对数据库性能优化具有重大意义。索引的建立与选择可遵循以下准则。

建议内容	具体描述
索引选用属性原则	选用经常作为查询，不常更新的属性建立索引；避免对常更新属性建立索引，因其严重影响性能
索引数量影响	关系上索引过多会影响 UPDATE、INSERT 和 DELETE 性能，因关系更新时所有索引都需相应调整
索引优化策略	分析每个重要查询使用频度，找出使用最多的索引并进行优化
小数据量关系处理	数据量非常小的关系不必建立索引，关系扫描更快且消耗系统资源更少

12.7.3.完整性约束（重点★★★★★）

数据库完整性由各种各样的完整性约束来保证，完整性约束可以通过 DBMS 或应用程序来实现，基于 DBMS 的完整性约束作为关系模式的一部分存入数据库中。此章节仍然是案例重点考察知识点，考察方式为概念默写。特别是下面表格的补充部分，需要理解记忆和掌握。

约束手段	描述	补充

实体完整性	实体完整性规则（Entity Integrity Rule）是指关系的主属性，即主码（主键）的组成不能为空，也就是关系的主属性不能是空值（NULL）。	/
参照完整性	若基本关系中含有与另一基本关系 S 的主键 PK 相对应的属性组 FK（FK 称为 R 的外键），则参照完整性要求，对及中的每个元组在 FK 上的值必须是 S 中某个元组的 PK 值，或者为空值。	插入删除问题（ 级联/受限/置空/递归）
触发器	触发器是一个数据库对象，当指定数据操作语言操作发生时（触发事件），该对象可以自动执行一个或多个 SQL 语句（触发操作）。	可以在一个表上定义一个或多个触发器以便在 INSERT、UPDATE 或 DELETE 触发事件发生之后进行操作 （不能直接对 SELECT 定义触发器，因为 SELECT 只是读取数据，不改变表的状态）。
用户定义完整性	针对特定数据库应用所定义的约束条件，由用户根据实际业务需求来制定。它可以对表中的列数据进行各种限制，如数据类型、取值范围、数据格式等，以确保数据满足业务规则。	例如，在员工信息表中，规定员工的年龄必须在 18 - 60 岁之间，或者规定某一列只能输入特定的值等。

12.7.4.触发器示例（重点★★★★★）

系分案例考过触发器填空，所以必须重视起来。以 MySQL 为例：

```
CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE | DELETE}
ON table_name
FOR EACH ROW
[trigger_body]


CREATE TRIGGER before_insert_user
BEFORE INSERT ON users
FOR EACH ROW
BEGIN
   SET NEW.create_time = NOW();
END;
```

这里，在触发器中，可以通过虚拟表访问行数据（NEW，OLD），如下表所示。

触发器	虚拟表
INSERT 触发器	只能用 NEW.column_name（新插入的值）
UPDATE 触发器	可以用 OLD.column_name（更新前的值）和 NEW.column_name（更新后的值）

DELETE 触发器	只能用 OLD.column_name（被删除的值）

12.8.备份与恢复技术（次重点★★★☆☆）

这一章节不太重要很少来考，看过算过，可以用自己的语言描述出来即可。数据库备份有多种分类方式，如下表所示。

分类方法	子分类	解读
按备份的实现方式	物理备份	指直接拷贝数据库的数据文件、日志文件等物理文件来进行备份。比如在关系型数据库中，将存储数据的 .mdf（SQL Server 数据文件）、dbf（部分数据库表文件）等文件以及日志文件复制到其他存储位置。这种备份方式与数据库的物理存储结构紧密相关，恢复时也是通过这些物理文件来恢复数据库到特定状态，恢复速度相对较快，常用于需要快速恢复数据库的场景。
	逻辑备份	通过数据库的导出工具，将数据库中的数据以逻辑对象（如表、视图、存储过程等）的形式导出为特定格式的文件，如 SQL 脚本文件等。它不直接操作数据库的物理文件，而是按照数据库的逻辑结构来备份数据。恢复时需执行这些逻辑文件中的指令来重建数据和对象，相对物理备份恢复速度可能较慢，但灵活性较高，可跨平台和数据库版本使用，常用于数据迁移、数据子集备份等场景。
按备份数据量情况	完全备份	对整个数据库进行完整的备份，包括所有的数据表、索引、视图、存储过程等数据库对象以及数据。它是最基础的备份方式，恢复时可以直接从完全备份中恢复整个数据库到备份时的状态，但备份所需时间长、占用存储空间大，因为每次都要备份全部数据。
	增量备份	只备份自上次备份（可以是完全备份或上一次增量备份）以来发生变化的数据。相比完全备份，它备份的数据量较小，备份速度快，占用空间少。但恢复时需要依次使用完全备份以及后续的所有增量备份来恢复数据，恢复过程相对复杂且耗时，如果其中某个增量备份损坏，可能影响恢复。
	差异备份	备份自上次完全备份以来发生变化的数据。它介于完全备份和增量备份之间，备份的数据量比增量备份大，但比完全备份小，备份速度也相对较快。恢复时，只需使用完全备份和最近一次的差异备份即可恢复数据，恢复过程比增量备份简单，在数据恢复和备份效率上取得一定平衡 。

12.9.数据仓库技术（次重点★★★☆☆）

这一章以往只在选择题中考察过 1-2 次，可以深挖的内容不多，建议大概了解即可。

12.9.1.联机分析处理（次重点★★★☆☆）

数据处理大致可以分成两大类，分别是 OLTP（联机事务处理）和 OLAP（在线分析处理）。OLTP 是传统数据库的主要应用，支持基本的、日常的事务处理（比如订单处理，客户信息管理，库存管理）；OLAP 是数据仓库系统的主要应用，支持复杂的分析操作，侧重决策支持，并且提供直观易懂的查询结果（比如数据挖掘、商业智能、决策支持）。两者更多对比如下所示。

对比维度	OLTP（联机事务处理）	OLAP（在线分析处理）
主要应用场景	传统数据库，支持基本日常事务处理（如订单处理、客户信息管理、库存管理）	数据仓库系统，支持复杂分析操作，侧重决策支持（如数据挖掘、商业智能、决策支持）
用户群体	操作人员，低层管理人员	决策人员，高层管理人员
功能	日常操作处理	分析决策
数据库设计	面向应用	面向主题
数据特点	当前的、最新的、细节的、二维的、分立的	历史的、聚集的、多维的、集成的、统一的
数据存取	读 / 写数十条记录	读上百万条记录
工作单位性质	简单的事务	复杂的查询
用户数量	多	少
数据库大小	MB 或 GB 级	GB 或 TB 级
数据组织形式	二维表	多维
例子	**电商系统**：用户下单、付款、修改收货地址。**银行系统**：取钱、转账、存款，每笔交易都要立即生效。 **库存管理**：商品入库、出库，每次数量要马上更新。 **这里的数据操作都是小而频繁的，比如插入一条订单记录、更新某个客户余额。**	**电商平台的 BI 报表**：统计某一季度的畅销商品、客户画像、各地区的销售额对比。 **银行风控**：分析过去 5 年所有交易数据，挖掘可疑的欺诈行为。 **超市经营决策**：通过历史销售数据，分析顾客偏好，决定下个月要多进哪类商品。 **这里的数据操作是大而复杂的，比如对上亿条订单记录做聚合查询、生成多维分析报表。**

12.9.2.多维分析（次重点★★★☆☆）

OLAP 的基本多维分析操作有钻取、切片和切块、旋转等，这个在选择题中考察过。

操作名称	定义	示例
钻取	改变维的层次，变换分析粒度，包括向上钻取和向下钻取	从汇总数据往下“钻”到更细的数据。比如：你先看“2025 年超市总销售额”，觉得太粗。再钻取到“按季度销售额”。继续钻取到“某季度 -> 按月份 -> 按天 -> 按小时”。
切片和切块	在部分维选定值后，关注度量数据在剩余维分布，剩余两维为切片，三维及以上为切块	切片：获取按月的销售数据；切块：获取按地区（如华东区、华北区等）的数据
旋转	变换维的方向，重新安排维的放置（如行列互换）	比如原来报表是“行=地区，列=季度”，旋转后变成“行=季度，列=地区”。

12.10.分布式数据库（次重点★★★☆☆）

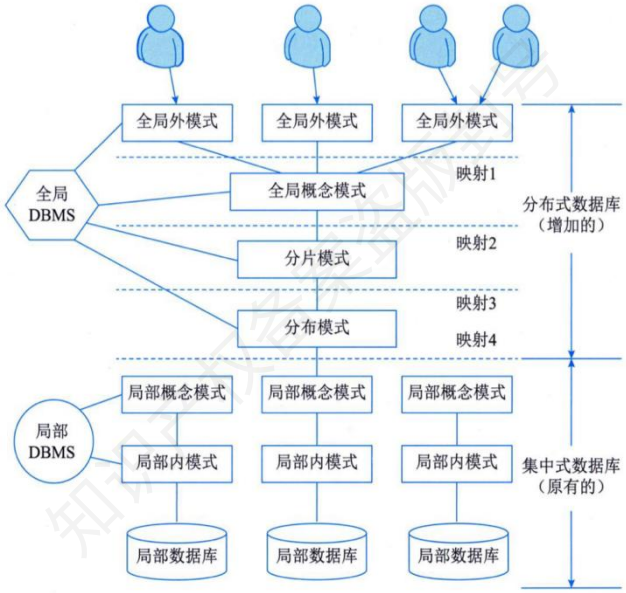
分布式数据库系统是数据库技术与网络技术相结合的产物,其基本思想是将传统的集中式数据库中的数据分布于网络上的多台计算机中。特点如下表所示。

特性名称	具体说明
数据独立性	在分布式数据库系统中更为重要，包含数据的逻辑独立性、物理独立性，以及额外的数据分布独立性（分布透明性）。
集中与自治共享结合的控制结构	各局部 DBMS 可独立管理局部数据库，具备自治功能；同时系统设集中控制机制，协调局部 DBMS 工作并执行全局应用。
适当增加数据冗余度	在不同场地存储同一数据的多个副本，以提高系统的可靠性、可用性及性能。
全局的一致性、可串行性和可恢复性	确保分布式环境下数据的全局一致性，事务执行的可串行性，以及系统故障后的可恢复性。

12.10.1.分布式数据库结构（次重点★★★☆☆）

这个结构如下图所示，由于分布式的引入，引入了全局和局部的概念。

模式名称	定义与功能
全局外模式	全局应用的用户视图，是全局概念模式的子集，直接与用户或应用程序交互。
全局概念模式	定义分布式数据库的整体逻辑结构，数据如同未分布般统一管理，通常采用关系模型，便于向其他层次映射。
分片模式	负责将全局关系模式分解为若干数据片段（分片），实现数据的逻辑拆分。
分布模式	定义数据片段的存放节点，决定分布式数据库的冗余性（一对多映射为冗余，一对一为非冗余）；支持全局查询分解为局部子查询，并通过映射将全局数据片段关联到局部概念模式。
局部概念模式	局部数据库的概念模式，与集中式数据库的概念模式一致。
局部内模式	局部数据库的内模式，与集中式数据库的内模式一致。



12.11.数据分片（次重点★★★☆☆）

数据分片（Data Sharding）是一种在数据库和分布式系统中常用的数据水平切分技术。它的基本思想是：当数据量过大、单一数据库或存储节点无法高效处理时，将一张表或一个数据集按照某种规则拆分成若干部分（即“分片”），分别存储在不同的数据库或服务器节点上，以提高系统的性能和可扩展性。这里主要关注分片的分类，如下表所示。

分片类型	定义与条件	重构方式

水平分片	将全局关系的元组按本关系属性值划分为多个子集	通过并操作恢复全局关系
垂直分片	将全局关系的属性划分为多个子集（关键字必须保留在所有分片中）	通过连接运算恢复全局关系
导出分片	分片条件来自其他关系的属性（又称导出水平分片）	通过并操作恢复全局关系
混合分片	在同一关系中同时采用水平分片与垂直分片	综合使用并操作与连接运算

13.信息安全（次重点★★★☆☆）

信息安全目前来看要么出选择要么出案例，论文好久不考也会考，是一个被低估的内容。想要稳扎稳打拿高分的同学这一章的内容还是熟悉。系分把一些容错技术放在这里，凯恩根据架构的分类把它们放在系统可靠性章节。

13.1.信息系统安全体系（次重点★★★☆☆）

13.1.1.系统安全分类（次重点★★★☆☆）

信息系统的安全是一个复杂的综合体，涉及系统的方方面面，主要包括实体安全、 信息安全、运行安全和人员安全 4 个部分，选择题常考爱考。

安全类型	主要内容	作用及要点
实体安全	保护计算机设备等实体，分环境、设备、媒体安全	避免经济损失，防止信息丢失破坏
运行安全	含系统风险管理、审计跟踪、备份与恢复、应急 4 方面	保障系统正常、可靠、连续运行
信息安全	分为操作系统 安全、数据库安全、网络安全、病毒防护、访问控制、数据加密和认证（鉴别）7 个方面	确保信息保密、完整、可用、可控
人员安全	涉及人员安全意识、法律意识、安全技能	从人员角度保障系统安全

13.1.2.等级保护（次重点★★★☆☆）

《计算机信息系统安全保护等级划分准则》（GB17859 - 1999）规定了计算机系统安全保护能

力的 5 个等级，关键是记忆适用对象这一栏目。

安全保护等级	主要特点	适用对象
用户自主保护级（第一级）	通过隔离用户与数据实现自主安全保护，对用户实施访问控制，保护用户和用户组信息	普通内联网用户
系统审计保护级（第二级）	实施更细粒度自主访问控制，用户通过登录规程等对自身行为负责	通过内联网或国际网进行商务活动，需保密的非重要单位
安全标记保护级（第三级）	具备第二级功能，提供安全策略模型、数据标记等，能准确标记输出信息并消除测试错误	地方各级国家机关、金融机构等众多重要单位
结构化保护级（第四级）	基于形式化安全策略模型，访问控制扩展到所有主体与客体，考虑隐蔽通道，抗渗透能力强，加强鉴别机制等	中央级国家机关、广播电视部门等关键部门与机构
访问验证保护级（第五级）	满足访问监控器需求，仲裁主体对客体全部访问，抗篡改且复杂性低，扩充审计机制，抗渗透能力很高	国防关键部门和依法需特殊隔离的单位

13.2.数据加密技术（重点★★★★★）

算法类型	算法名称	密钥长度	特点
对称加密算法	AES	128/192/256 位	目前最常用、最安全的对称加密算法之一。具有高效的加密速度和强大的安全性，能够在不同的硬件和软件平台上运行。
	DES	56 位	虽然曾经广泛使用，但由于密钥长度较短，安全性相对较低。不过在一些对安全性要求不高的老旧系统中仍可能会出现。
	3DES	168 位	通过对数据进行三次 DES 加密，增加了安全性，密钥长度相对较长。但加密速度比 DES 慢。在真题里说密钥长度是 112 位，这里它是从有效安全位角度考虑的。所以真题碰到你要写 112 位。
	RC-5	可变（0-2040 位，常用为 128 位）	具有参数可变的特点，可适应不同的安全需求和性能要求，加密轮数可变，可根据实际情况调整安全性和速度。
	IDEA	128 位	类似于 3DES，在 DES 基础上发展起来，明文和密文都是 64 位，加密强度高，效率较高，可在不同的硬件和软件平台上运行。
非对称加密算法	RSA	1024/2048 位	非常著名且应用广泛的非对称加密算法。安全性基于大整数分解的数学难题，具有较高的安全性和可靠性。可用于数字签名和密钥交换。
	ECC	160 - 521 位	与 RSA 相比，在提供相同安全级别的情况下，密钥长度更短，计算效率更高，占用的存储空间更小。

商用密码有很多，作为一览，我整理出下表，列举了常用的国际跟国产商密，如下表所示。

密码分类	国产商用密码	国际商用密码

对称加密	分组加密/块加密	SM1/SCB2 SM4/SMS4 SM7	DES IDEA AES RC5 RC6
	序列加密/流加密	ZUC SSF46	RC4
非对称加密/公钥加密	大数分解		RSA DSA ECDSA Rabin
	离散对数	SM2 SM9	RSA ECC ECDH
密码杂凑/散列		SM3	MD5 SHA-1 SHA-2

下表的密码中要注意 SM1-9 的概念，知道 SM2/SM9 是非对称加密即可。

国产商用密码汇总			
算法名称	**算法类型**	**密钥长度及分组长度**	**特点及应用**
SM1	分组加密算法	128 位分组长度和密钥长度	安全性与 AES 相当，以 IP 核形式存在于加密芯片中
SM2	基于椭圆曲线密码的非对称加密算法	256 位密钥	支持数字签名、密钥交换和公钥加密，旨在替代 RSA、ECDSA 和 ECDH 等国际算法
SM3	密码杂凑算法	-	适用于数字签名、消息认证和随机数生成，用于替代 MD5、SHA - 1 和 SHA - 2 等算法
SM4	分组加密算法	128 位密钥长度和分组长度	用于替代 DES 和 AES 等算法
SM7	分组加密算法	-	适用于非接触 IC 卡应用，如身份识别、票务和支付类应用
SM9	基于标识的非对称密码算法	-	提供数字签名生成与验证算法，可替代基于数字证书的 PKI/CA 体系，主要用于用户身份认证
祖冲之密码算法	流加密算法	-	包括 ZUC 算法、加密算法（128 - EEA3）和完整性算法（128 - EIA3），适用于 3GPP LTE 通信

13.3.认证技术（重点★★★★★）

认证技术是信息安全的核心技术之一，主要用于确认用户、设备或数据的真实性与合法性。它保证了信息传输与访问过程中的身份确认、数据完整性、防伪造和防抵赖。总结如下表所示。

分类	具体内容

数字签名	附加在数据单元上的数据或密码变换，用于确认数据来源、完整性和防止伪造；基于非对称加密算法，有普通和特殊数字签名算法；主要功能是保证信息传输完整性、发送者身份认证和防止抵赖。
杂凑算法	利用散列函数加密，如 MD5 将任意长度字节串变成定长大数，产生 128 位消息摘要；SHA 家族算法对长度不超过 264 位消息产生 160 位消息摘要，SHA 有 5 个算法。
数字证书	由认证中心签发对用户公钥的认证，内容包括 CA 及用户信息、公钥、时间等；国际上遵从 X.509 体系标准；不同 CA 发放证书需证书链通信，CA 维护证书吊销列表。
身份认证	口令认证：用户名 / 密码简单常用但不安全，易泄漏和被截获。
	动态口令认证：密码按时间或使用次数变化，一次一密较安全，但可能存在同步问题和输入不便。
	生物特征识别：通过身体或行为等生物特征认证，分为身体和行为特征，又有高级、次级和深奥生物识别技术分类。

13.3.1.数字签名（重点★★★★★）

数字签名（Digital Signature）是公钥基础设施（PKI）的核心组成部分。在 PKI 体系中，常见的要素包括数字证书、证书颁发机构（CA）、银行使用的安全 Key，以及 SSL/TLS 通信等，而数字签名正是这些应用的技术基础。

13.3.1.1.Hash 算法与数字签名（重点★★★★★）

在理解数字签名之前，必须先掌握 Hash（散列/杂凑）算法。Hash 算法的主要特性包括：易压缩性：可将任意长度的数据映射为固定长度的输出。单向性：从源数据得到哈希值容易，但无法由哈希值推回源数据。高灵敏性：输入数据发生微小变化，输出结果会有显著差异。抗碰撞性：不同输入得到相同哈希值的概率极低。

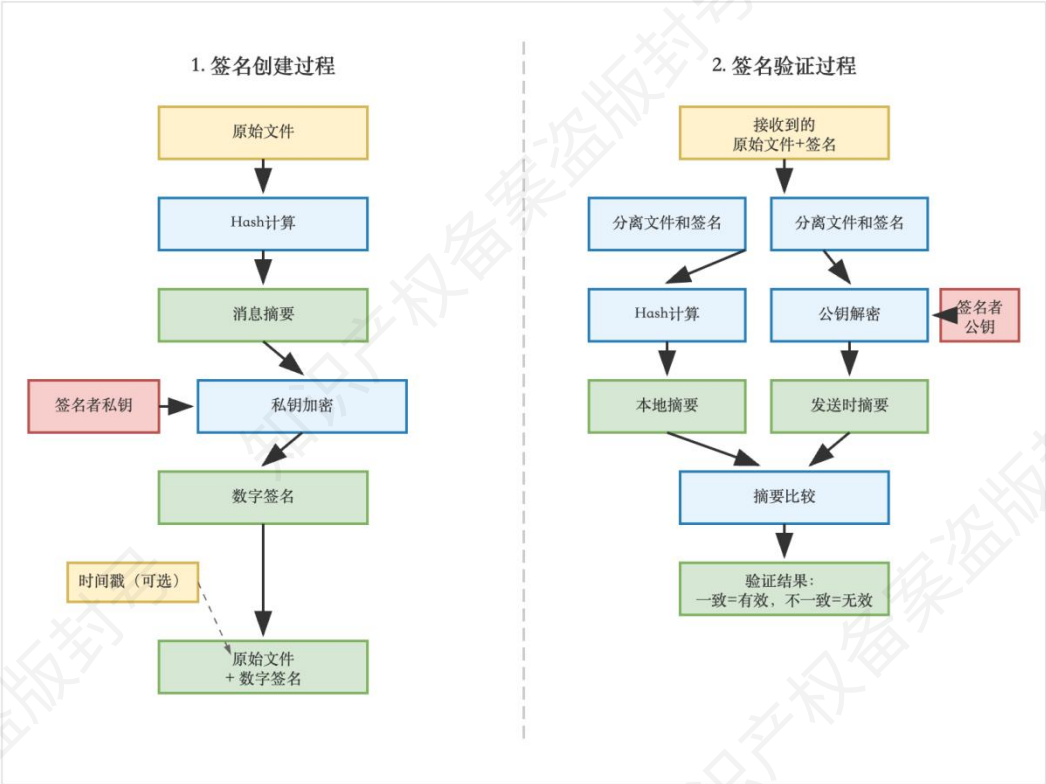
常用算法包括 SHA-1、SHA-256、SHA-384 以及我国的 SM3。由于计算能力的提升，SHA-1 已被证明存在安全风险，逐步退出历史舞台。由于具备这些特性，Hash 算法非常适合用于数据完整性校验，是数字签名的重要组成部分。

13.3.1.2.数字签名的工作原理（重点★★★★★）

1.签名的创建过程。对需要签名的文件进行 Hash 计算，得到消息摘要。使用签名者的私钥对消息摘要进行加密，生成数字签名。可以选择加入时间戳，标识签名的创建时间。最终发送的内容包括原始文件和数字签名。需要注意，数字签名只对文件的 Hash 值进行签名，而不是直接对整个文件签名，这样既节约了资源，又提高了效率。

2.签名的验证过程。接收方对收到的原始文件进行 Hash 计算，得到本地摘要。使用签名者的公钥解密数字签名，得到发送时的摘要。比较两个摘要，如果一致，则文件完整且签名有效；若不一致，则说明文件被篡改或签名无效。

数字签名创建与验证流程示意图



13.3.2.数字证书（重点★★★★★）

在数字签名的过程中，私钥用于加密消息的摘要，而公钥则用于对端解密以进行认证。这个过程虽然能有效验证信息的完整性和来源，但也引出了一个关键问题：如何确保公钥确实属于消息的

发送者，而不是被恶意篡改或伪造的？这个问题正是数字证书和证书认证体系的核心。

当接收方使用发送方的公钥来验证数字签名时，如果公钥在传输过程中被篡改或伪造，接收方便无法正确验证签名，从而导致安全漏洞。为了避免这种情况，就需要确保公钥的真实性——这正是数字证书的作用所在。

数字证书通过引入证书中心（CA）来解决这个问题。CA 充当了一个信任的第三方，它会验证公钥的归属关系，并通过自己的私钥对公钥及其相关信息进行加密，生成数字证书。接收方可以通过 CA 的公钥解密证书，验证公钥的真实性，确保收到的信息来自真实的发送者，而非被恶意篡改。

下面是一个典型的过程。首先，客户端（如浏览器）向服务器发起加密通信请求。服务器响应时，会返回自己的数字证书，并使用私钥对信息进行加密。接着，客户端使用证书中的 CA 公钥解密证书，并验证其有效性，以确保证书由受信任的 CA 签发且服务器身份无误。如果验证通过，客户端就可以使用服务器的公钥加密敏感信息，并与服务器安全地交换数据。这一过程中，数字证书确保了双方身份的真实性和数据的加密保护。

这里引申出非对称公钥的使用场景。这里我们看到的多是信源验证场景，验证信息来自于某个我们可靠的信息源。此时是通过私钥加密信息（摘要）形成数字签名，然后把公钥+信息+数字签名发送出来，在接收端公钥用于解密信息，并对传递过来的信息做一个摘要，然后进行比较，假如匹配说明信源可靠。

但是公钥私钥同样可以用于加密场景（比如交换对称加密的密钥，因为非对称加密效率低，对称加密效率高。但是对称加密双方要约定一个密钥用于通讯，所以此时出现了这个场景）。前面在签名信源验证场景，我们已经通过数字证书的形式把公钥带给客户端。此时客户端可就以用此公钥加密双方通信所需密钥，然后再传给服务端。服务端收到加密信息后，就可以用对应的私钥解密，能拿到密钥。

13.4.密钥管理体系（次重点★★★☆☆）

密钥管理在信息系统加密保护中至关重要，其处理密钥从产生到销毁全过程的相关问题。当前主要有三种密钥管理体制，分别是适用于封闭网的 KMI 机制、适用于开放网的 PKI 机制以及适用于规模化专用网的 SPK 机制。它们各自适用的网络需要清楚。

密钥管理体制	适用网络	特点及实现方式
KMI 机制	封闭网	设 KDC 发密钥，有静态和动态分发，基于秘密信道，静态含点对点、一对多、网格状分发，动态有对称和非对称加密分发
PKI 机制	开放网	遵循标准的密钥管理平台，解决依赖秘密信道问题，含 CA 等组件，基于加密等技术
SPK 机制	规模化专用网	两种实现方式，其中 LPK 用 RSA 算法，有缺陷；CPK 用 DLP 或 ECC 实现，克服 LPK 缺点

13.5.通信与网络安全技术（次重点★★★☆☆）

这一章节主要考察防火墙、虚拟专用网、安全协议、单点登录技术的相关概念。其中安全协议、单点登录技术是容易考察的内容。

13.5.1.安全协议（次重点★★★☆☆）

在保证计算机网络系统的安全方面，安全协议起核心作用，主要包括 IPSec、SSL、PGP 和安全套接层上的超文本传输安全协议（HTTPS）。网络安全协议常考以下四种辨析，必须熟记。

协议名称	基本信息	主要功能与特点
SSL	传输层安全协议，由握手、记录、警报协议组成	用于 Internet 传机密文件，提供身份认证、数据加密与完整性保护。使用 40 位 RC4 算法，实现过程主要包括接通阶段、密码交换阶段、会谈密码阶段、检验阶段、客户认证阶段和结束阶段，可用于安全邮件
HTTPS	HTTP 的安全版，应用 SSL 于 HTTP 应用层，使用特定端口	建立安全通道保障数据传输，确认网站真实性。难窃听，防中间人攻击，支持 X.509 认证
PGP	基于 RSA 的邮件加密软件	对邮件保密与签名，防非授权阅读，确认发件人，也用于文件加密。采用独特加密流程，承认两种证书格式

IPSec	工业标准网络安全协议，针对 IPv4/6	为 IP 网络通信提供透明安全服务，保护数据，抵御攻击。基于端对端模式，支持 IP 级流量加密认证

13.5.2.单点登录技术（次重点★★★☆☆）

单点登录（Single Sign-On，SSO）技术是通过用户的一次性认证登录，即可获得需要访问系统和应用软件的授权，在此条件下，管理员不需要修改或干涉用户登录，就能方便地实现希望得到的安全控制。常见的单点登录实现技术如下表所示。

机制	与单点登录关系	例子
Kerberos 机制	Kerberos 为单点登录提供基于信任第三方的身份认证方法，是实现单点登录的重要技术手段，保障在开放网络环境下的安全认证。	员工在登录办公系统时，向 Kerberos 服务器进行身份验证，验证通过后，服务器会为员工颁发票据。当员工访问邮件系统和文件共享系统时，无需再次输入账号密码，只需出示之前获得的票据，系统通过与 Kerberos 服务器验证票据的有效性，即可确认员工身份并授权访问
外壳脚本机制	通过原始认证进入系统外壳激活目标平台访问，简化登录流程，契合单点登录理念，但在口令同步等管理方面有欠缺，可在单点登录实施中改进或与其他机制配合。	员工通过原始用户名和密码登录系统外壳，外壳脚本会根据员工权限，激活对应的办公软件（专用脚本）的访问权限。员工登录一次即可使用这些集成的软件。

13.6.系统访问控制技术（次重点★★★☆☆）

访问控制是策略和机制的集合，允许对限定资源的授权访问，是系统安全保障机制核心，实现数据保密性和完整性主要手段。这里书本上提到了控制模型和控制分类，会在选择题考察。

13.6.1.访问控制分类（次重点★★★☆☆）

访问控制分类如下表所示，明白基本概念即可。

访问控制类型	基本思想	实现方式与特点	适用场景

自主访问控制（DAC）	依主体身份和所属组，主体指定其他主体对自身资源的访问及类型	给用户或组分配权限规则集，主体可自主授权，但需求变化时授权繁琐	多用户环境
强制访问控制（MAC）	主体和客体有既定安全属性，访问取决于两者关系，常需满足 BLP 模型特性	同 DAC 分配权限，安全性高但管理员工作重，易出漏洞	安全性要求高的系统
基于角色的访问控制（RBAC）	用户关联角色，角色关联访问许可权	有 4 种模型，灵活、方便、安全，能描述多种安全策略	大型系统，如大型 DBMS 权限管理
基于任务的访问控制（TBAC）	权限随任务上下文变化，是动态安全模型	由工作流等 4 部分组成，支持最小特权和最小泄露原则，动态管理权限	工作流、分布式处理等系统
基于对象的访问控制（OBAC）	ACL 关联受控对象及属性，设计访问控制选项，允许策略复用、继承、派生	可控制对象及属性，派生对象继承访问设置，减轻权限管理工作量	信息量大且变化频繁的管理系统

13.6.2.访问控制模型（次重点★★★☆☆）

访问控制一般都是基于安全策略和安全模型的。访问控制模型是一种从访问控制的角度出发，描述系统安全，建立安全模型的方法。建立规范的访问控制模型，是实现严格访问控制策略所必需的。选择题会考概念辨析。了解即可。

模型名称	模型概述	核心规则
Bell - LaPadula 模型	第一个正式的安全模型，基于强制访问控制（MAC）系统，是典型的信息保密性多级安全模型，主要应用于军事系统，将数据和用户安全等级划分为公开、受限、秘密、机密和高密 5 个等级	上读 no read up ：主体不可读取安全级别高于它的数据；下写 no write down ：主体不可写入安全级别低于它的数据
Lattice 模型	通过划分安全边界对 BLP 模型进行扩充，将用户和资源分类，允许它们之间交换信息，注重形成 “安全集束”，执行访问控制功能时本质与 BLP 模型相同，遵循 BLP 模型的上读和下写原则，但需各对象位于相同的安全集束中	上读 no read up （同 BLP 模型，但需在相同安全集束）：主体不可读取安全级别高于它的数据；下写 no write down（同 BLP 模型，但需在相同安全集束）：主体不可写入安全级别低于它的数据
Biba 模型	类似于 BLP 保密性模型，也使用 MAC 系统，对数据提供分级别的完整性保证，模仿 BLP 模型的信息保密性级别定义信息完整性级别，避免越权和篡改行为发生	下读 no read down：主体不能读取安全级别低于它的数据；上写 no write up：主体不能写入安全级别高于它的数据

14.嵌入式系统分析与设计（次重点★★★☆☆）

嵌入式系统相关的知识点每年有 2-3 分，并且有些概念会在案例中考察。这一章的学习主要是掌握嵌入式数据库、嵌入式操作系统、嵌入式系统开发，主要以概念记忆为主。难点是多任务调度算法，好在考的不多，不需要特别关注。

14.1.嵌入式系统概述（重点★★★★★）

嵌入式系统是一种以应用为中心、以计算机技术为基础，软硬件可以根据具体需求进行配置和裁剪的专用计算机系统。它是嵌入到某个具体设备或业务场景中，完成特定功能。比如汽车电子控制单元、智能家电控制器、工业控制设备、医疗监护设备、航空航天飞控系统等，都可以看作嵌入式系统的典型应用。

嵌入式系统设计时通常要同时考虑功能、可靠性、成本、体积和功耗等因素。它是在特定应用场景下做平衡：比如有的系统要求实时响应，有的系统要求低功耗，有的系统要求高可靠性，有的系统要求体积足够小。因此，嵌入式系统往往体现出“软硬件一体化设计”的特点。

14.1.1.嵌入式系统的主要特点（重点★★★★★）

嵌入式系统的主要特点可以概括为五个方面：一是专用性强，通常面向特定任务设计，功能目标明确；二是实时性强，很多系统必须在规定时间内完成响应，不能只追求计算正确，还要保证响应及时；三是资源受限，受成本、体积、功耗等因素影响，CPU、内存、存储和接口资源通常有限；四是软硬件依赖性强，软件往往需要结合具体芯片、接口和外设进行开发与优化；五是系统透明性强，用户通常感知不到其中的计算机系统，只关注设备本身提供的功能。

14.1.2.嵌入式系统的软件和硬件架构

从系统架构角度看，嵌入式系统通常可以分为硬件层、中间层、系统软件层和应用层。其中，

硬件层提供基础能力，BSP/HAL 屏蔽硬件差异，系统软件层提供运行环境，应用层实现具体业务功能。

硬件层是整个嵌入式系统的基础，主要包括嵌入式处理器、存储器、通用设备接口和 I/O 接口等。处理器负责计算和控制，存储器负责保存程序和数据，常见存储器包括 SDRAM、ROM、Flash 等。嵌入式系统中的操作系统和应用程序有时会固化在 ROM 或 Flash 中。除此以外，硬件层还包括电源电路、时钟电路、复位电路、通信接口、A/D 转换、D/A 转换等内容。

硬件层之上是中间层，也称为硬件抽象层 HAL 或板级支持包 BSP。这一层的作用非常关键，它负责屏蔽底层硬件差异，把硬件能力封装成上层软件可以调用的接口。简单说，BSP/HAL 的作用就是让上层软件不用直接面对复杂的底层硬件。它通常包含硬件初始化、设备驱动、输入输出操作、硬件配置等功能。

再往上是系统软件层，通常包括嵌入式操作系统、文件系统、图形用户接口、网络协议栈、嵌入式数据库以及一些通用组件。其中，实时多任务操作系统 RTOS 是很多嵌入式应用的重要基础。RTOS 可以提供任务调度、中断管理、定时器、任务通信、资源管理等能力，帮助开发人员构建复杂的实时控制系统。

最上层是应用层，也就是直接实现业务功能的软件。比如智能门锁中的指纹识别、密码校验、开锁控制；汽车系统中的发动机控制、车身控制、制动控制；工业设备中的参数采集、报警、控制策略等，都属于应用层内容。应用层通常建立在操作系统、驱动程序和支撑软件之上。

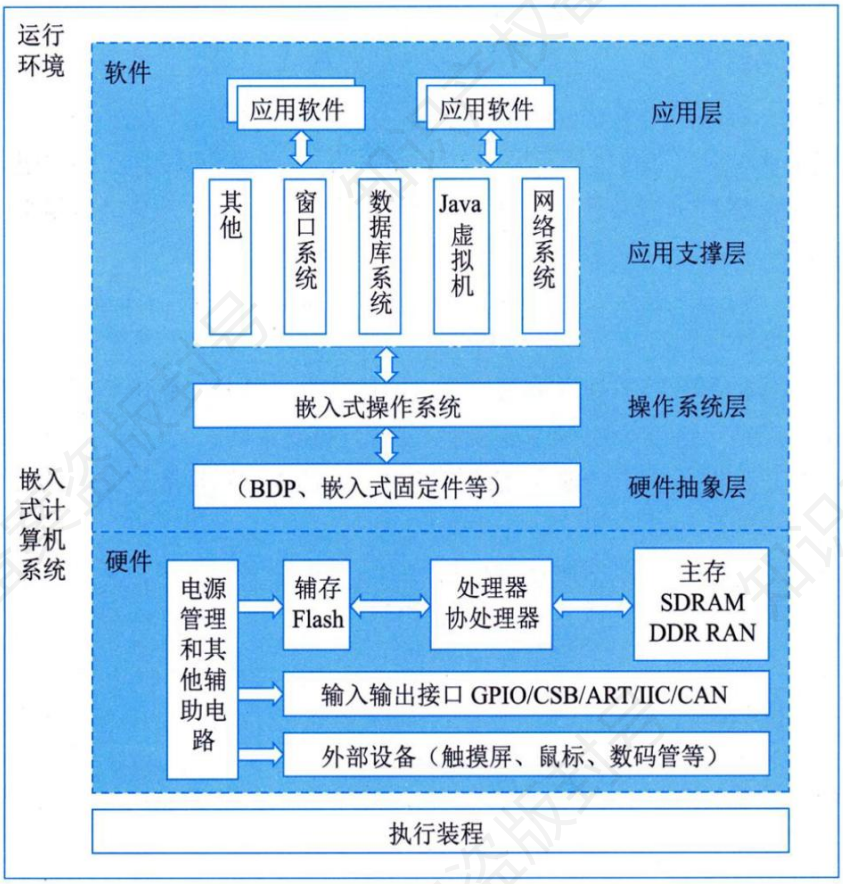


图 17-2　嵌入式计算机的组成

14.2.嵌入式数据库系统（重点★★★★★）

嵌入式数据库管理系统，简称 EDBMS，是指运行在嵌入式系统中，用于存储、管理和访问数据的数据库管理系统。它通常与嵌入式操作系统和应用软件集成在一起，固化或部署在嵌入式设备上，为设备中的专门应用提供数据管理能力。

嵌入式数据库更多运行在手机、车载设备、掌上终端、工业控制设备、智能仪表等资源受限的设备中。因此，它不能像传统数据库那样大量依赖 CPU、内存、磁盘和网络资源，而要适应嵌入式系统小型化、低功耗、低资源和专用化的特点。

14.2.1.嵌入式数据库系统特点（重点★★★★★）

嵌入式数据库管理系统的特点主要体现在嵌入性、实时性、移动性和伸缩性四个方面。

第一，嵌入性。它通常不是独立运行的大型数据库服务器，而是嵌入到软件或硬件设备中，作为嵌入式系统的一部分，负责内部数据的存储、查询和同步，用户一般不会直接感知它的存在。

第二，实时性。在工业控制、电网监测、智能交通等场景中，数据需要及时采集、保存和处理。因此，嵌入式数据库往往要求在限定时间内完成读写和响应，否则即使结果正确，也可能失去实际意义。

第三，移动性。嵌入式数据库常用于手机、PDA、车载设备、移动采集终端等移动设备。这类设备可能处于网络不稳定甚至离线环境中，所以数据库需要支持本地缓存、离线使用和数据同步。

第四，伸缩性。不同嵌入式设备的 CPU、内存、存储空间和操作系统环境差异较大，因此嵌入式数据库需要具备一定的裁剪和配置能力，以适应不同平台和资源条件。

14.2.2.嵌入式数据库系统的组成（重点★★★★★）

一个完整的嵌入式数据库系统通常由嵌入式数据库、同步服务器、数据库服务器和连接网络组成。

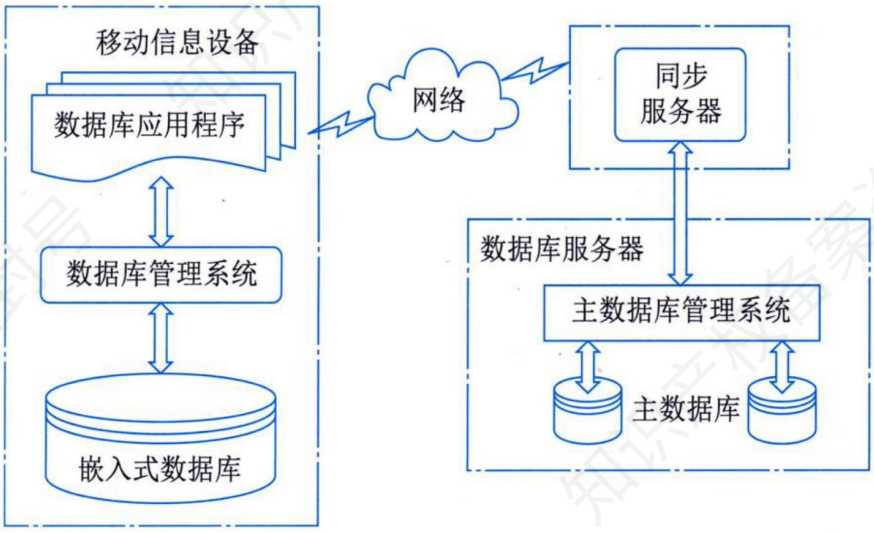


图 17-5　嵌入式数据库系统组成

嵌入式数据库运行在移动设备或嵌入式设备本地，负责管理设备端的数据。它可以在没有网络的情况下独立运行，支持本地数据的存储、查询和更新。比如移动采集终端在野外没有网络时，仍

然可以把采集数据保存到本地数据库中。

同步服务器是嵌入式数据库和主数据库之间的连接枢纽。它的核心作用是完成数据同步，保证设备端数据库和服务器端主数据库之间的数据尽量保持一致。比如移动终端本地录入了一批数据，网络恢复后，需要通过同步服务器上传到主数据库；服务器端数据更新后，也可以通过同步服务器下发到移动终端。

数据库服务器保存主数据库，通常运行大型通用数据库管理系统。它负责集中管理核心业务数据，是整个系统的数据中心。嵌入式设备上的本地数据库通常只是主数据库的一个局部副本或工作副本。

连接网络负责把移动设备、同步服务器和数据库服务器连接起来。数据库服务器和同步服务器之间一般使用较稳定的固定网络，而移动设备和同步服务器之间可能使用无线局域网、蜂窝网络、串口、红外或其他连接方式。

14.3.嵌入式操作系统（重点★★★★★）

嵌入式操作系统，简称 EOS，是运行在嵌入式系统上的操作系统。它负责管理嵌入式系统中的硬件资源和软件资源，为嵌入式应用程序提供运行环境和系统服务。

比如一个车载控制系统中，应用程序只关心如何采集车辆状态、如何控制执行机构、如何和其他模块通信，但底层的任务调度、中断响应、设备驱动、通信协议等，就需要由嵌入式操作系统提供支撑。

嵌入式操作系统的价值主要体现在两个方面：一是提高嵌入式软件开发效率，二是增强应用软件的可移植性。如果没有操作系统，开发人员往往需要直接操作硬件，程序和硬件绑定非常紧；有了 EOS 以后，应用程序可以通过标准接口调用系统能力，开发和维护都会更方便。

14.3.1.嵌入式操作系统的分类（重点★★★★★）

嵌入式操作系统通常可以按照实时性要求分为两类：硬实时操作系统和软实时操作系统。

硬实时操作系统主要用于控制、通信、航空航天、工业控制、汽车电子等对时间要求极高的领域。所谓硬实时，不是简单地说“运行速度快”，而是指任务必须在规定时间内完成。一旦错过截止时间，即使计算结果正确，也可能造成严重后果。比如汽车制动控制、飞行控制、工业安全保护系统，都属于典型硬实时场景。

软实时操作系统主要用于消费电子、智能终端、机顶盒、电子书、手机等领域。软实时系统也希望任务尽快完成，但偶尔超时通常只会造成体验下降，不一定导致系统失败。比如手机界面卡顿、视频播放掉帧，就是软实时系统中比较典型的问题。

14.3.2.宏内核、微内核和混合内核（重点★★★★★）

从内核结构看，操作系统内核可以分为宏内核、微内核和混合内核。

宏内核把很多系统服务都放在内核空间中运行，例如进程管理、内存管理、文件系统、设备驱动等。它的优点是结构相对直接，系统调用开销较小，性能和实时性较好；缺点是代码耦合度高，一个模块出问题可能影响整个内核，移植性和可维护性相对较差。

微内核只把最基本的功能放在内核中，例如最小的进程管理、内存管理和通信机制，其他服务尽量放到用户空间中运行。它的优点是结构清晰、模块隔离好、可移植性和扩展性强；缺点是不同服务之间需要频繁通过消息通信，系统性能可能受到影响。

混合内核介于宏内核和微内核之间。它保留微内核的结构思想，但为了提高性能，会把一些常用或性能敏感的服务放回内核空间中运行。可以把它理解成一种折中设计。

14.4.多任务调度算法（次重点★★★☆☆）

嵌入式操作系统通常要支持多任务运行。多个任务同时存在时，就会产生任务调度问题：CPU

在某一时刻只能执行一个任务，那么系统应该先执行哪个任务，什么时候切换任务，这就由调度算法决定。

调度算法可以按不同角度分类。按照时限要求，可以分为软时限算法和硬时限算法；按照任务是否周期出现，可以分为周期性算法和非周期性算法；按照运行任务是否可以被打断，可以分为可抢占算法和非抢占算法；按照优先级是否在运行过程中变化，可以分为静态算法和动态算法；按照调度信息是否在运行前确定，可以分为离线调度和在线调度。考试中最需要掌握的是几类典型调度算法。

14.4.1.优先级的抢占调度（重点★★★★★）

它的基本思想是：任何时候，系统都运行所有就绪任务中优先级最高的任务。如果一个更高优先级任务变为就绪状态，就可以抢占当前低优先级任务的 CPU。很多实时嵌入式操作系统都采用这种调度方式，因为它有利于系统及时响应重要外部事件。

优先级可以是静态的，也可以是动态的。静态优先级是在设计或创建任务时确定，运行过程中不变；动态优先级则可以在运行过程中根据任务状态、截止时间或系统策略进行调整。

14.4.2.时间轮转调度（重点★★★★★）

时间轮转调度主要用于同优先级任务之间。系统为每个任务分配一个时间片，一个任务运行完一个时间片后，就让出 CPU，排到同优先级就绪队列的尾部，等待下一轮调度。

时间片大小非常关键。时间片太大，轮转效果不明显，系统响应可能变慢；时间片太小，任务切换过于频繁，CPU 开销又会增大。纯粹的时间轮转调度通常难以满足强实时系统要求，所以实际中常用的是基于优先级的抢占式时间轮转调度：不同优先级之间按优先级抢占，同一优先级内部再按时间片轮转。

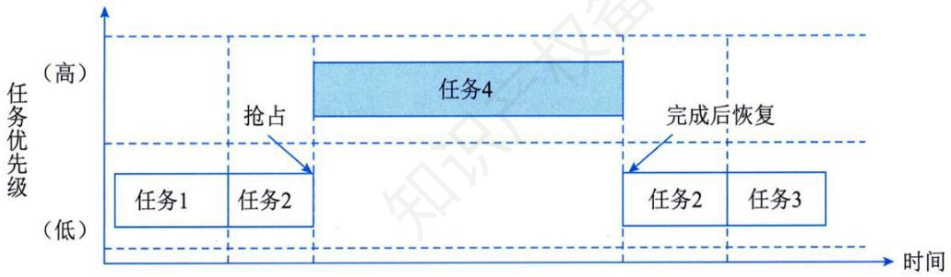


图 17-10　基于优先级的抢占式时间轮转调度

在此图中，任务 1、任务 2 和任务 3 具有相同的优先级，它们按照各自的时间片运行，任务 2 被更高优先级的任务 4 抢占，当任务 4 执行完毕后恢复任务 2 的执行。

14.4.3.单调速率调度 RMS（次重点★★★☆☆）

RMS 是一种静态固定优先级调度算法，适用于周期性实时任务。它的核心规则是：任务周期越短，优先级越高。因为周期越短，说明任务发生得越频繁，通常越需要及时调度。

14.4.4.截止时间优先调度 EDF（重点★★★★★）

EDF 是一种动态优先级调度算法，它的核心规则是：截止时间越近，优先级越高。每当有新任务到达时，系统都会根据任务截止时间重新排序。如果新任务的截止时间比当前任务更早，就可以抢占当前任务。

任务	到达时间	持续时间	时限
任务1	0	15	26
任务2	6	3	20
任务3	8	6	21

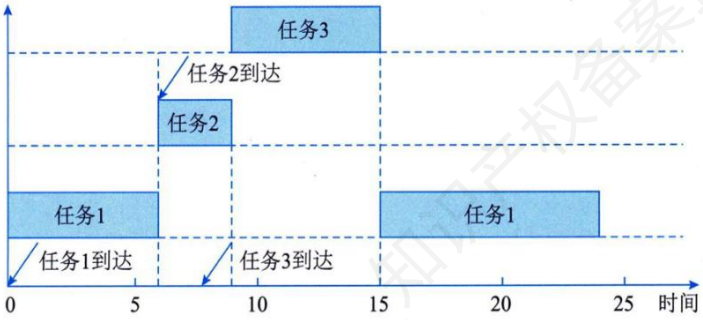


图 17-11　基于截止时间优先调度算法的任务调度图

从这个图里可以看到。先看左侧表格，到达时间和持续时间很好理解。关键这里引入了时限。意味着我的这个任务必须在这个时限内完成。

所以，当任务 1 到达时，它是系统中等待运行的唯一任务，因此立即得到执行。任务 2 在时间 6 到达，由于任务 2 的截止时间(20)早于任务 1 的截止时间(26)，因此，任务 2 的优先级比任务 1 的更高，任务 2 抢占任务 1 的 CPU 得到执行。

任务 3 在时间 8 到达，由于任务 3 的截止时间晚于任务 2 的截止时间，因此，任务 2 的优先级高于任务 3，任务 2 继续运行，待任务 2 执行结束，任务 3 才能运行。任务 3 执行结束后，任务 1 才能继续运行，在时间 24 执行完毕。

EDF 的优点是理论 CPU 利用率上限可以达到 100%，也就是说，只要任务集合能够被调度，EDF 通常具有很强的调度能力。但是 EDF 需要在运行过程中动态调整优先级，调度开销较大，实现复杂度也更高。

14.4.5.分区调度和 APPS 调度（次重点★★★☆☆）

这类调度主要用于综合化模块化航空电子系统等强安全、强隔离场景。它的基本思想是把处理器时间划分成多个固定时间窗口，不同分区在自己的时间窗口内运行，从而实现时间隔离。分区调度通常是静态的、确定的、按固定周期重复执行。这个内容相对专业，考试中只要知道它强调“分区、时间窗口、时间隔离、静态确定调度”即可。

15.法律法规（重点★★★★★）

选择题重点考察，以往难度不大基本属于送分，这两年开始考专利法相关内容后就变得比较难了。这一块每年都会有 2-4 分。

知识产权也称为“智力成果权”“智慧财产权”。它是人类通过创造性的智力劳动而获得的一项权利。根据我国民法通则的规定，知识产权是指民事权利主体（自然人、法人）基于创造性的智力成果。知识产权具有无形性、专有性、地域性和时间性四大特点。我国相关法律法规主要包括《著作权法》、《计算机软件保护条件》、《专利法》、《商标法》和《反不正当竞争法》。本章就针

对这些主要的法律法规进行详细地介绍。

中国公民、法人或者其他组织的作品，不论是否发表，都享有著作权。开发软件所用的思想、处理过程、操作方法或者数学概念不受保护、软件作品并不是指代码，而是指带有特定业务逻辑的程序以及软件文档。

著作权法不适用于下列情形：法律、法规，国家机关的决议、决定、命令和其他具有立法、行政、司法性质的文件，及其官方正式译文；时事新闻；历法、通用数表、通用表格和公式。

15.1.保护对象（重点★★★★★）

法律法规	保护对象	注意事项
著作权法	文学、艺术和科学领域内具有独创性并能以一定形式表现的智力成果，如文字作品、音乐、美术、影视等。	不需要申请，作品完成即开始保护；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	软件作品，包括计算机程序及其有关文档。	不需要申请，作品完成即开始保护；登记制度便于举证。
专利法	发明创造，包括发明、实用新型和外观设计。	需要申请，专利权有效期是从申请日开始计算。
商标法	商标，用于区别商品或服务来源。	需要申请，核准之日起商标受保护。
反不正当竞争法	经营者的合法权益、市场竞争秩序。	商业秘密包括技术与经营两个方面；必须有保密措施才能认定为商业秘密。

15.2.保护期限（重点★★★★★）

客体	保护对象	保护期限
公民作品	署名权，修改权，保护作品完整权	没有限制
	发表权，使用权和获得报酬权	作者终生及其死亡后的 50 年（第 50 年的 12 月 31 日）

单位作品	发表权、使用权和报酬权	首次发表后的第 50 年
公民软件作品	署名权，修改权	没有限制
	其他	作者终生及死后 50 年（第 50 年 12 月 31 日）。合作开发，以最后死亡作者为准。
注册商标		有效期 10 年（若注册人死亡或倒闭 1 年后，未转移则可注销，期满后 6 个月内必须续注）
发明专利		保护期为 20 年（从申请日开始）
实用新型		保护期为 10 年（从申请日开始）
外观设计专利权		保护期为 15 年（从申请日开始）
商业秘密		不确定，公开后公众可用

15.3.职务作品（重点★★★★★）

情况说明	场景	产权归属
作品	利用单位的技术条件进行创作	单位（署名权除外）
	合同约定著作权属于单位	单位（署名权除外）
	其他	作者拥有著作权，单位有权在业务范围使用
软件	本职工作明确的开发目标	单位（署名权除外）
	属于从事本职工作活动的结果	单位（署名权除外）
	使用单位资源并由单位承担责任的软件	单位（署名权除外）
专利	本职工作中作出的发明创作	单位（署名权除外）
	履行本单位本职工作之外的发明创造	单位（署名权除外）
	离职、退休或调动工作后一年内，与原单位工作相关	单位（署名权除外）

情况说明		场景
作品软件	委托创作	默认归创作方，可以约定给委托方
	合作开发	共同享有
商标		谁先申请谁拥有（除知名商标的非法抢注）、同一天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，同一天申请则协商归属，但不能够同时驳回双方的专利申请

（21 年系分真题）某软件公司参与开发管理系统软件的程序员丁某，辞职到另一公司任职，该公司项目负责人将管理系统软件的开发者署名替换为王某，该项目负责人的行为（__）。

A 不构成侵权，因为丁某不是软件著作权人 B 只是行使管理者的权利，不构成侵权

C 侵犯了开发者丁某的署名权　D 不构成侵权，因为丁某已离职

答：本题考查署名权的相关知识，凯恩建议掌握。这里的关键是署名权是开发者的永久权利（和其他著作权益分开讨论），不管开发者有没有在该公司任职，该项目负责人都不能将署名权替换成其他人，所以该项目负责人的行为侵犯了开发者丁某的署名权。

选项 A：丁某参与了管理系统软件的开发，是软件的开发者之一，享有署名权，所以 A 选项错误。

选项 B：项目负责人这种未经许可擅自将丁某署名替换为王某的行为，不属于正常行使管理者权利范畴，已构成对丁某权利的侵犯，并非不构成侵权，所以 B 选项错误。

选项 C：署名权是著作权中的人身权，属于开发者的永久性权利，它保护开发者在作品上表明自己身份的权利。即便丁某离职，其对参与开发软件所享有的署名权依然受法律保护。项目负责人擅自将丁某的署名替换为王某，侵犯了丁某的署名权，C 选项正确。

选项 D：开发者的署名权不会因离职而丧失，只要参与了软件的开发创作，就享有署名权，所以不能因为丁某离职就认为不构成侵权，D 选项错误。

所以选择选项 C。

15.4.侵权判定（重点★★★★★）

不侵权场景	侵权场景
个人用于自身学习、探究或者品鉴； 进行适度的援引； 有关公开演讲的内容； 应用于教学或者科学探究方面； 对馆藏作品进行复制； 无偿表演他人的作品； 在室外公共场合对艺术品进行临摹、绘画、拍摄、录像； 把汉语作品翻译为少数民族语言作品或者以盲文形式出版。	在未获得许可的情况下，对他人作品进行发表； 在未得到合作作者应允的情形下，把与他人合作创作的作品当作是自身单独创作的作品予以发表； 没有参与创作，却在他人作品中署名； 对他人作品进行歪曲、修改； 窃取他人作品； 运用他人作品时，没有支付相应报酬； 在未取得出版者同意的状况下，使用其出版的图书、期刊的版式设计。

15.5.专利法（补充重点★★★★★）

专利法的客体是发明创造，也就是其保护的对象。这里的发明创造是指发明、实用新型和外观设计。发明是指对产品、方法或者其改进所提出的新的技术方案；实用新型是指对产品的形状、构造及其组合，提出的适于实用的新的技术方案；外观设计是指对产品的形状、图案及其组合，以及色彩与形状、图案的结合所作出的富有美感并适于工业应用的新设计。

考点分类	考点详情	具体内容
授予专利权的条件	发明和实用新型的条件	须具备新颖性（申请前国内外无同样发明或实用新型，特定情况不丧失新颖性）、创造性（相比原有技术有突出特点和显著进步）、实用性（能制造或使用且有积极效果）
	外观设计的条件	与国内外发表的外观设计不相同、不相近似
	不授予专利权的对象	科学发现、智力活动的规则和方法、疾病的诊断和治疗方法、动植物品种及用原子核变换方法获得的物质
确定专利权人	基本归属原则	专利权归属于对发明创造作出创造性贡献的发明人或设计人，组织、辅助人员不属于此列

	职务发明	执行单位任务或利用单位物质技术条件完成的发明创造，包括本职工作中、本职工作外任务、退职退休调动 1 年内相关任务作出的发明；专利申请批准后单位为专利权人，也可签合同重新规定归属
	合作发明、设计	专利权通常共同所有，可依合同确定归属
	委托发明	未签订合同规定归属时，专利权归发明、设计者
	特殊情况	非职务发明单位无权压制个人申请；多个类似专利申请，专利权归最先提交的申请人
专利权	不视为侵权的情形	专利权人相关产品售出后使用等；申请日前已制造相同产品等在原有范围继续；国外运输工具临时使用；专为科研和实验使用
	保护期限	发明专利权 20 年，实用新型 10 年，外观设计专利权 15 年，均从申请日起算
	专利权终止情形	未按时缴纳年费、书面声明放弃；专利复审未通过
	专利实施许可	具备条件单位可请求许可实施；国家紧急状态等可强制实施发明、实用新型专利许可

16.后记

各位会员同学，当你看到这里时，我们已经共同完成了一段知识探索与整合的旅程。

这份讲义从最初成形到如今的版本，每一处增删、每一次调整，都源于我对学习便利性与高效性的执着追求。为了让大家少走弯路，我尽可能把重复内容进行归并，把零散知识点进行分类标注，把容易混淆的概念用对比的方式整理出来，希望你翻到任何一页，都能更快找到重点，更顺畅地建立起完整的知识框架，从而节省宝贵的备考时间。

在内容取材上，我尽力做到有据可依、互相印证：一方面参考官方教材，保证主干不偏离考试要求；另一方面查阅专业书籍与相关资料，补足教材中讲得不够细、不够透的部分，力求使内容更系统、更科学。对关键内容的强调、对常考点的归纳、对解题思路的提炼，都是为了帮助大家在复

习中抓住主线，在选择题训练中更快突破。

然而，我也必须坦诚说明：受限于个人能力与精力，这份红宝书仍然存在不足。某些知识点的深度与广度，可能尚未达到最理想的程度；个别表述与推导，或许还不够精准、严谨；复杂概念的解释方式，也未必完全契合每一位同学的理解习惯；在知识关联的梳理上，也可能存在疏漏与欠完善之处。对此，我心中始终怀有歉意，也希望大家能够理解。

但请相信，这些不足不会成为你们前进路上的障碍。相反，它们提醒我们：学习与完善从来不是一次完成的事情，而是不断修订、不断靠近正确答案的过程。你们在使用过程中提出的每一个疑问、每一条建议，都将成为我持续改进的重要依据，也会让这份讲义更贴近考试、更贴近学习，更配得上你们的信任。

最后，衷心感谢各位同学一路以来的支持与陪伴。若在使用过程中有任何疑问或建议，欢迎随时通过微信 Deckardcain4 与我交流。愿这份讲义能在你的备考路上带来切实的帮助，愿你在每一次复习与训练中都更接近目标。祝各位同学学习顺利，考试成功。

知识产权备案盗版封号