软考中级

1.计算机网络概论

> 硬件基本组成

运算器、控制器、存储器、输入设备、输出设备

CPU

> 运算器

- 算术逻辑单元 ALU：负责处理数据，加法器
- **累加寄存器** AC：为ALU提供一个工作区,暂存源操作和结果
- 数据缓冲寄存器 DR：cpu和内存外设之间的数据传送中转站
- 状态条件寄存器 PSW：为指令建立条件码内容

> 控制器

- 指令寄存器 LR：指令 = 内存->缓冲寄存器->IR
- **程序计数器** PC：寄存信息；计数。保存下一条要执行的指令地址
- 地址寄存器 AR：保存当前cpu访问的内存地址
- 指令译码器 ID：对指令中的操作码字段进行分析，识别操作，发出具体控制信号

在中断响应中，CPU保护程序计数器的主要目的是为了CPU在执行完中断服务程序时能回到被中断程序的断点处

数据表示

原码：

反码：正数的反码与原码相同，负数的反码按位取反

补码：正数补码=原码=反码，负数补码=反码+1

移码：补码的基础上符号位取反

$N=2^E\times F$ E阶码，F尾码

浮点数的精度取决于尾数M的位数，范围取决于阶码E的位数

补码最适合进行数字加减运算，移码最适合表示浮点数阶码

±0 编码相同的是补码和移码

对阶：阶码小的向阶码大的对齐

校验码

> 奇校验

在编码中增加一位校验位来使编码中1的个数为奇数，（偶校验同理）

对于奇校验，若有奇数个数据位出错，可以检测出错误但无法纠正

> 海明码

$(2^k)-1 \geq n+k$ n数据位 k校验位

海明码每个数据位由确定关系的校验位来校验

海明码利用奇偶性进行检错和纠错

> 循环冗余校验码 CRC

求CRC编码，采用模2运算

计算机体系结构

> flynn分类法

根据计算机在执行程序的过程中指令流和数据流的不同组成，分为4类：

SISD、SIMD、MISD(无实际作用)、MIMD

指令系统

CISC：更复杂，指令直接对主存单元的数据进行处理，微码多，通用机，微程序控制器

RISC：配置更多寄存器，专用机，组合逻辑控制器，寻址方式单一，指令少，硬布线逻辑

> 流水线

流水线的执行周期为流水线上时间最长的一段？存疑

时间计算公式：一条指令的完整周期 + (指令数-1)*指令各段的最大执行时间

存储系统

SRAM静态随机存储器：工作速度快，需要一直供电，cache采用静态随机存储

DRAM动态随机存储器：使用电容存储，间隔一段时间刷新，不刷新数据丢失，内存使用动态存储器

虚拟存储体系由 主存-辅存 两级存储器构成

> Cache

Cache存储器用来存放主存的部分拷贝，不能扩大主存容量

Cache命中率必须很高，达到90%以上

主存与cache的地址映射方式：

- 直接映像：固定
- 全相联映像：允许调用cache的任何一个块
- 组相联映像

发生块冲突最小的是全相联映像，组相连，直接

> 磁表面存储器

非格式化容量 = 面数 × (磁道数/面) × 内圆周长 × 最大位密度

格式化容量 = 面数 × (磁道数/面) × (扇区数/道) × (字节数/扇区)

磁盘调度先移臂调度，然后旋转调度

固态硬盘一般采用闪存或者DRAM: 读写快速、质量轻、能耗低

> 廉价冗余磁盘阵列

RAID-0：不具备容错能力

RAID-1：镜像容错改善可靠性

RAID-2：海明码进行错误检测

RAID-3：减少磁盘存储器的个数

RAID-4：独立进行读写

RAID-5：同一个磁盘记录数据和检验信息

RAID-6：分别进行两个独立的校验运算形成两个独立的冗余数据

闪存：闪存掉电后信息不丢失，以块为单位进行删除，嵌入式系统中有flash代替ROM存储器，闪存不能代替主存

I/O系统工作方式

> 程序控制方式

- 无条件查询：I/O端总是准备好输入输出主机
- 程序查询：cpu轮询方式 测试I/O设备的端口，处于ready就进行读取操作

> 中断方式

程序控制I/O，发起中断指令或启动指令，cpu与I/O并行，提高cpu利用率

> DMA（直接内存存取）

允许主存和I/O设备之间通过DMA控制器直接进行数据交换，整个过程不需要cpu

每传送一个数据都需要占用一个存储周期

CPU是在一个总线周期结束时响应DMA请求

> 通道控制方式

在一定硬件基础上利用软件手段控制I/O，免去cpu接入

> I/O处理机 (通道的延伸)

专门负责输入输出的处理机，可以有独立的存储器、运算器、指令控制部件

总线

总线是指计算机设备和设备之间传输信息的公共数据通道

- 地址总线
- 数据总线
- 控制总线

总线上的多个部件只能分时向总线发送数据，但可同时从总线接收数据

寻址

- 立即寻址：直接指出操作数本身的寻址方式
- 直接寻址：地址码字段给出有效地址
- 间接寻址：给出存放操作数地址的地址
- 寄存器寻址：

计算机中CPU的中断响应时间指从发出中断请求到开始进入中断处理程序

指令周期大于时钟周期

管道过滤器体系结构的优点：

- 软件构建具有良好的高内聚、低耦合的特点

- 支持重用
- 支持并行执行

加密技术

数字签名采用私钥进行签名，公钥进行验证，是对真实性的保护

> 对称加密

采用对称密码编码技术，特点是文件加密和解密使用相同的密钥

- 数据加密标准算法 **DES**：共享密钥加密，用56位密钥对64位加密，输出64位
- 三重DES：两个56位密钥，112位
- RC-5：可以对明文消息进行加密传输
- RC-4：报文认证算法
- 国际数据加密算法IDEA：只能用来进行数据加密，128位密钥
- AES：128、192、256位密钥，用128位解密
- **MD5**：用来生成消息摘要,长度128

> 非对称加密

公钥、私钥

- RSA：主要用来数据签名和验签

> 认证技术

- SSL：https

SHA-1是一种针对不同输入生成160位固定长度摘要的算法

计算机可靠性

串联 R = R1R2...

并联 R = 1-(1-R1)(1-R2)

MTTF/(1+MTTF) 平均故障时间

1/(1+MTTR) 可维护性

计算机性能评价

- 时钟频率
- 指令执行速度
- 等效指令速度法
- 数据处理速率
- 核心程序法

> 计算机系统性能

- 吞吐率：最长流水段操作时间的倒数
- 响应时间
- 资源利用率

媒体

- 表示媒体：语言编码、电报码、条形码
- 表现媒体：输入输出设备
- 感觉媒体：人的感觉，声音
- 传输媒体：传输数据的物理载体，电缆、光缆

MPEG-1：VCD、MPEG-2：DVD

MPG是视频文件格式

色调指的是一幅画中色彩的总体倾向；亮度是指发光体表面发光强弱的物理量；饱和度指颜色的鲜艳程度(纯度)

采样频率定义为8kHz，这是因为信号定义的频率最高值为4kHz

人耳能听到的信号频率范围是20Hz~20kHz

改变数字载波频率可以改变乐音的音调

微程序一般由硬件执行

是否可简化看是否形成阻塞，不阻塞可以简化

VLIW是超长指令字的简称

IPv4地址长度 32位

IPv6地址长度128位

2.程序设计语言基础知识

语法：基本字符构成的符号

语义：静态语义指编译时可以确定的语法成分的含义，动态语义是运行时刻才能确定的语义

解释：直接解释执行源程序

编译：将源程序翻译成目标函数

值调用、引用调用

编译过程

- 词法分析：记号，字符
- **语法分析**：
    - 对源程序的逻辑**结构**进行分析
    - 语句形式是否正确
    - 移进归约分析法是自下而上的语法分析方法
    - 句子结构是否符合语法规则
- 语义分析：
    - 类型检查
    - 是否包含静态语义错误
    - 语法制导翻译是一种静态语义分析
    - 程序运行时陷入死循环，可能是动态语义错误
- 中间代码生成：后缀式、三地址码、树
- 代码优化：
- 目标代码生成：
    - 解释方式不包含，分配寄存器的工作在目标代码生成阶段进行
    - 表达式的除数是否为零
- 符号表管理：
- 出错处理：

编译正确的程序不包含语法错误

对于大多数通用程序设计语言，用**上下文无关文法**描述即可

编译器将代码转换为机器码执行，速度快；解释器参与运行控制，程序执行速度慢

编译正确的程序不包含语法错误

中间代码生成和代码优化 不是每个编译器必须的

语法分析技术

语法分析器接受以单词为单位输入

- LR分析法
  - LR(1)分析能力最强
  - LR(0)分析能力最弱
- 算符优先法
- 递归下降法

C语言 需要通过 预处理、编译、汇编、链接 才能执行

语句结构：顺序语句、选择语句、循环语句

3.数据结构

线性结构

线性表，顺序存储；链式存储

栈，后进先出

队列，先进先出

串

循环链表的优点是从表中任一结点出发都能访问到整个链表

散列存储结构将结点按其关键字的散列地址存储到散列中

树

> 基本概念

- 结点的度：一个结点的子树的个数
- 叶子结点：终端结点
- 内部结点：度不为0的结点
- 结点层次：
- 树的高度：一颗树的最大层数
- 有序树：

高度为k的二叉树 有 2^k - 1个结点

> 遍历

- 先序遍历：根左右
- 中序遍历：左根右
- 后序遍历：左右根

任何一颗二叉树的叶结点在前序、中序、后序中的相对次序不发生改变。

二叉树

- 完全二叉树
- 最优二叉树：哈夫曼树，带权路径长度最短的树
- 平衡二叉树，时间复杂度O(log2n)，左子树和右子树高度绝对值之差不超过1
- 满二叉树
- 二叉查找树（二叉排序树）：左子树非空，结点小于根；右子树非空，结点大于根；左右子树本身是二叉排序树

完全二叉树中任意一个结点的左、右子树的高度之差的绝对值

> 哈夫曼树

哈夫曼树中权值最小的两个结点互为兄弟结点

权值越大的叶子离根结点越近

不存在只有一个子树的结点

结点总数一定为奇数

图

> **遍历**

使用链表进行深度优先遍历 DFS

使用队列对图进行广度优先遍历 BFS

邻接矩阵的无向图进行深度优先遍历

深度优先算法类似于二叉树的先序遍历

> 最小生成树

Prim算法适合求网较稠密的生成树

Kruskal算法适合求稀疏图。

都采用贪心算法

哈希函数

如何构造哈希函数，如何处理冲突

- 开放定址法
- 链地址法
- 再hash法
- 建立一个公共溢出区

排序

稳定性：两个元素在排序前后的相对位置不变

排序方式	时间复杂度	稳定性	备注
直接插入排序	O(n^2)	稳定	
简单选择排序	O(n^2)	不稳定	
冒泡排序	O(n^2)	稳定	
希尔排序	O(n^1.3)	不稳定	
快速排序	O(nlogn)	不稳定	
堆排序	O(nlogn)	不稳定	是一种插入排序
归并排序	O(nlogn)	稳定	最坏情况下复杂度最低
基数排序	O(d(n+rd))	稳定	

对**有序的数组**进行排序，最适宜采用**插入排序**

优先队列通常采用堆数据结构，向优先队列中插入一个元素的时间复杂度为O(lgn)

实现二分查找，要求查找表顺序存储，关键码有序排列

4.操作系统知识

吞吐量：计算机系统在单位时间内处理工作的能力。

操作系统4大特征：并发性、共享性、虚拟性、不确定性

操作系统5大部分：进程管理、文件管理、存储管理、设备管理、作业管理

进程管理

PCB是进程存在的唯一标志

运行、就绪、阻塞

进程同步：在系统中一些需要互相合作，协同工作的进程

进程互斥：多个进程因争用临界资源而互斥执行

> 信号量 pv

p操作表示申请一个资源，v操作表示释放一个资源，S表示某资源的可用数

> 调度算法

- 先来先服务 FCFS
- 时间片轮转
  - 固定时间片
  - 可变时间片
- 优先级调度
- 多级反馈调度

> 死锁

4个必要条件：互斥条件、请求保持条件、不可剥夺条件、环路条件

I/O设备管理

与设备无关的系统软件、设备驱动程序、中断处理程序、硬件

> 作业调度

R = 1 + 作业等待时间/作业执行时间，响应比高优先

嵌入式系统初始化过程：（自底向上）

- 片级初始化
- 板级初始化
- 系统初始化：以软件初始化为主，主要进行操作系统的初始化

嵌入式操作系统的特点：

微型化、可定制、实时性、可靠性、易移植性

嵌入式系统设计中，高速缓存对程序员是透明的

linux只有一个根目录，用"/"表示

BIOS保存在主板上的ROM中

5.软件工程基础知识

开发小组人员为N，通信信道为 N(N-1)/2

> 软件生命周期

可信性分析与项目开发计划、需求分析、概要设计、详细设计、编码、测试、维护

软件过程

> CMM 指软件过程能力成熟度模型

- 初始级
- 可重复级：可重复类似的开发
- 已定义级：已将软件过程文档化，标准化
- 定量管理级：利用统计工具分析并采取改进措施
- 优化级：持续改进

> CMMI

- CL0(未完成的)
- CL1(已执行的)：共性目标是过程将可标识的输入工作产品转换成可标识的输出工作产品
- CL2(已管理的)
- CL3(已定义级的)
- CL4(定量管理的)
- CL5(优化的): 使用量化(统计学)

> 软件活动

软件描述、软件开发、软件有效性验证和软件进化

软件过程模型

答案解析： 开发模型的特点如下：

类型	特征
瀑布模型	结构化方法。开发阶段性、**需求明确**、文档齐全、风险控制弱。
原型模型	迭代方法。分为原先开发与目标软件开发。**需求不明确**。
螺旋模型	迭代方法。瀑布与原型（演化）模型结合体。适用于**大型**、**复杂**、**风险**项目。
喷泉模型	面向对象方法。**复用好**、开发过程**无间隙**、节省时间。
V模型	开发与**测试**结合。
变换模型	适用于**形式化开发**。
智能模型	适用于基于规则的**专家系统**。
快速应用开发RAD	基于构件的开发方法。**用户参与**、开发或复用构件、**模块化要求高**，不适用新技术。
RUP/UP	**用例驱动**、**架构为中心**、**迭代**、**增量**。
可重用构建模型	基于构件的开发方法。**开发或复用构件**。

瀑布模型

以文档为驱动，适合于软件需求很明确的软件项目模型

容易理解、管理成本低，开发费用高

增量模型

第一个交付版本所需要的成本和时间很少

需求不清晰对开发进度有影响

管理成本低，容易理解

演化模型

迭代的过程模型，web项目

- **原型模型**：需求不完整、不准确；适用于小系统规模，适合用户需求不清晰且经常发生变化

  用来探索特殊的软件解决方案、支持用户界面设计

  - 水平原型（行为原型）：功能的导航
  - 垂直原型（结构化）：实现一部分功能，用在复杂算法实现上
  - 抛弃式原型：不确定、不完整的项目
  - **演化原型**：开发增量式，
- **螺旋模型**：综合了瀑布和演化模型，适用于庞大、复杂并且具有高风险的系统，适用于大型软件开发，明确了开发中的**风险**

喷泉模型

以用户**需求为动力**，以对象作为驱动的模型，适合于面向对象的开发方法

迭代性、**无间隙性**，**支持重用**，要求严格管理文档

基于构件的开发模型

领域工程和应用系统工程

形式化方法模型

统一过程模型UP

用例和风险驱动，以架构为中心，迭代并且增量

- 起始阶段：生命周期目标
- 精化阶段：生命周期架构
- 构建阶段：初始运作功能
- 移交阶段：产品发布
- 生产阶段：

> 敏捷方法

- 极限编程XP：高效、低风险；先测试、后编写
- 水晶法Crystal：不同项目，不同策略
- 并列争求法Scrum：迭代，30天一个迭代周期，不包括重构refactoring
- FDD功能驱动法：开发人员分类
- ASD自适应：预测-协作-学习
- 开放式源码：虚拟团队
- 敏捷统一过程AUP：

需求分析

> 需求工程

需求获取、需求分析与协商、系统建模、需求规约、需求验证、需求管理

系统测试

- 单元测试：侧重于模块中的内部处理逻辑和数据结构
  - 模块接口、局部数据结构、模块内路径、边界条件、错误处理
- 集成测试：模块组合起来测试
- 确认测试：集中于用户可见的动作和用户可识别的系统输出
  - α测试，β测试
- 系统测试：软硬件、外设、网络综合测试；
  - 恢复测试
  - 安全性测试
  - 压力测试
  - 性能测试
  - 部署测试
- 验收测试：

测试方法

软件测试是为了发现错误

软件测试对象：程序，数据，文档

> 黑盒测试

- 等价类划分
- 边界值分析
- 错误推测
- 因果图

> 白盒测试（结构测试）

- **逻辑覆盖**
  - 语句覆盖：每条语句至少执行一次
  - 判定覆盖(分支覆盖)：每个判断真和假至少执行一次
  - 条件覆盖：每一判定语句逻辑条件的每种可能取值至少满足一次
  - 判定/条件覆盖：每个条件所有可能的取值组合至少出现一次
  - 条件组合覆盖：每个判断的所有逻辑条件的可能取值的组合至少执行一次
  - 路径覆盖：覆盖所有可能路径，属于动态测试方法
- 循环覆盖
- 基本路径测试

- 静态测试
  - 代码检测法
  - 静态结构分析法
  - 静态质量度量法

软件维护

- 改正性：改正在系统开发阶段已发生而系统测试阶段尚未发现的错误
- 适应性：应用软件适应信息技术变化和管理需求变化而进行的修改
- 改善性：为扩充功能和改善性能
- 预防性：为了适应未来的软/硬件环境的变化
- 数据维护

软件项目估算

> 成本估算方法

自顶向下估算方法

自底向上估算方法

差别估算方法

COCOMO估算模型

COCOMOII 估算模型：对象点(体系结构阶段模型)、功能点、代码行

Putnam估算模型：源代码行数、开发时间、工作量、技术状态

进度管理

Gantt图：能清晰地描述每个任务从何时开始，到何时结束，并行性。不能反应任务之间的依赖关系，难以确定里程碑

PERT图：松弛时间，给出开始结束时间，给出任务之间的关系，不能反应并行关系

> 软件质量

特征：功能性、可靠性、易使用性、效率、可维护性、可移植性

> McCabe度量法

$V(G) = m - n + 2p$ m是弧的个数，n是结点个数，p强连通分量个数

快捷算法：闭合区域+1

> 软件维护工具

版本控制工具，文档分析工具，开发信息库工具，逆向工具，再工程工具

> QFD

期望需求、基础需求、意外需求

冗余附加技术：独立设计的相同功能冗余备份程序的存储及调用；实现纠错检测及恢复的程序；为实现容错软件所需的固化程序

正式技术评审的目标是发现软件中的错误

> 概要设计

系统架构、模块划分、系统接口、数据设计

6.结构化开发方法

系统规模不太大且不复杂，最适宜采用结构化开发方法

耦合

耦合类型	描述
非直接耦合	两模块无直接关系，联系完全通过主模块的控制和调用
数据耦合	借助参数表传递简单数据
标记耦合	通过参数表传递记录信息(数据结构)
控制耦合	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	访问同一全局变量(非全局数据结构)，不是通过参数表传递
公共耦合	访问同一个公共数据环境(如全局数据结构、共享通信去、公共内存)
内容耦合	不通过正常入口直接访问另模块的内部数据，代码重叠，模块有多个入口

耦合是模块之间的相对独立性

- 无直接耦合
- 数据耦合：有调用关系，传递数据参数
- 标记耦合：传数据结构
- 控制耦合：一个模块通过传送开关等控制信息，控制另一个模块
- 外部耦合：
- 公共耦合：通过一个公共数据环境相互作用
- 内容耦合：一个模块直接使用另一个模块的数据，通过非正常入口转入另一个模块内部

内聚

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下表所示。

内聚类型	描述
功能内聚	完成单一功能，各部分协同工作，缺一不可
顺序内聚	处理元素相关，且顺序执行
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，必须按特定的次序执行
瞬时内聚	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	完成一组没有关系或松散关系的任务

对一个模块内部各个元素彼此结合的紧密程度的度量

- 功能内聚：完成单一功能，各个部分协同工作，最强的内聚
- 顺序内聚：处理元素相同，顺序执行
- **通信内聚**：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，按待定的次序执行
- 巧合内聚：完成一组没有关系的任务
- 逻辑内聚：把几种功能组合在一起，由传送给模块的判定参数确定执行哪种功能
- 时间内聚：同时执行的动作组合在一起
- 标记内聚：一组模块通过参数表传递信息

划分软件系统时，应尽量做到高内聚低耦合

结构化分析方法

抽象和分解是两个基本手段

> 数据流图

- 外部实体 E
- 数据存储 D
- 数据加工
- 数据流

数据流图中的元素在数据字典里定义

> 数据字典

数据流、数据项、数据存储、基本加工

> 模块结构图

模块、调用、数据、控制信息和转接符号

用户界面设计

> 原则

用户操纵原则、减轻用户记忆负担、保持界面一致

7.面向对象技术

常见的命名对象有：变量、函数、数据类型

一个对象通常使用对象**标识、属性和方法**组成

边界对象表示了系统与参与者之间的接口

过载多态：指操作具有相同的名称，且在不同的上下文中所代表的含义不同

面向对象的类：

- 实体类：
- 控制类：用于控制用例工作
- 边界类（接口类）：可以实现界面控制、外部接口和环境隔离

面向对象设计原则：（前5大原则）

- 里式替换原则：子类可替换父类
- 依赖倒置原则：要依赖于抽象，不是具体实践
- 接口分离原则：依赖于抽象
- **开放-封闭**原则：可扩展，不可修改
- 共同封闭原则：包中所有类对于同一类性质的变化应该是共同封闭的
- 单一责任原则：
- 迪米特原则：一个对象应当对其他对象尽可能少的了解

行为型对象模式使多个对象都有机会处理请求

Java语言采用即时编译，对象在堆空间分配

UML

- 类图：展示了一组对象、接口、协作和它们之间的关系
  - 依赖：一个事物发生变化会影响另一个事物
  - 关联：对象之间的链接关系，聚集：整体与部分
  - 泛化：子类共享父类的结构和行为
  - 实现：一个类制定了另一个类保证执行的契约
  - 组合：
- 对象图：展现了某一时刻一组对象以及它们之间的关系，实例的静态快照
- 用例图：用例、参与者以及它们的关系
- 交互图：传递消息，包含 对象、链和消息
  - 序列图：时间顺序组织的对象之间的交互活动，一个用例多个对象
  - 通信图：强调收发消息的对象，或者参与者的结构
  - 交互概览图：活动图和顺序图的混合物
  - 计时图：关注沿着线性时间轴、生命线内部和生命线之间的条件该表
- 状态图：状态、转换、事件和活动组成
- 活动图：特殊的状态图，展现了在系统内部从一个活动到另一个活动的流程，复杂用例中的业务流程进行进一步建模
- 构件图：展现了一组构件之间的组织和依赖
- 组合结构图：用于描述一个分类器的内部**结构**
- 部署图：软件和硬件之间的物理关系以及处理节点的分布情况
- 包图：描述由模型本身分解而成的组织单元，以及他们的依赖关系

定时图强调消息跨越不同对象或参与者的实际时间

设计模式

> 创建型设计模式：创建对象

- **抽象工厂 Abstract Factory**：提供一个接口，可以创建一系列相关或相互依赖的对象，而无需指定它们具体的类。

- 生成器 Builder：将一个复杂类的表示与其构造相分离，使得相同的构建过程能够得出不同的类，允许被构造的对象有不同的表示
- 工厂方法 Factory Method：定义一个用于创建对象的接口，让**子类**决定实例化哪一个类
- 原型 Prototype：用原型实例指定创建对象的类型，并且通过拷贝这个原型来创建新的对象
- 单例 Singleton：保证一个类只有一个实例，并提供一个访问它的全局访问点

> 结构型设计模式：处理类或对象的组合

- **适配器 Adapter**：将一个类的接口转换成客户希望的另外一个接口，使得原本由于接口不兼容而不能一起工作的那些类可以一起工作
- **桥接** Bridge：**将抽象部分与其实现部分分离**，使它们都可以独立变化，适用于不希望在抽象和它的实现部分之间有一个固定的判定关系
- 组合 Composite：将对象组合成树型结构以表示“整体-部分”的层次结构
- **装饰 Decorator**：动态地给一个对象添加一些额外的职责
- 外观 Facade：定义一个高层接口，为子系统中的一组接口提供一个一致的外观，从而简化该子系统的使用
    - GUI
- 享元 Flyweight：提供支持大量细粒度对象共享的有效方法
- 代理 Proxy：为其他对象提供一种代理以控制这个对象的访问

> 行为设计模式：描述类与对象怎么交互，怎么样分配职责

- 责任链 Chain of Responsibility：使得多个对象有机会处理请求，从而避免请求的发生者和接受者之间的耦合
- 命令 Command：将一个请求封装为一个对象，适用于抽象出特执行的动作以参数化对象
- 解释器 Interpreter：给定一个语言，定义它的文法
- 迭代器 Iterator：提供一种方法顺序访问一个聚合对象中的各个元素
- 中介者 Mediator：用一个中介对象来封装一系列的对象交互
- 备忘录 Memento：在不破坏封装性的前提下捕获一个对象的内部状态
- **观察者 Observer**：当一个对象状态发生改变，所有依赖于它的对象都得到通知并更新
- 状态 State：允许一个对象在其内部状态改变时改变它的行为
- 策略 Strategy：定义一系列算法，一个个封装，
- 模板方法 Template Method：定义一个操作中的算法骨架
- 访问者 Visitor：表示一个作用于某对象结构中的各元素的操作

8.算法设计与分析

算法设计原则

- 正确性
- 可读性
- 健壮性
- 高效率和低存储量

策略

- 分治：分解成子问题，快速排序，归并排序
- 贪心：局部最优解
    - 邻分背包
    - 策略与递归技术的联系最弱
- 动态规划：经分析得到的子问题不是独立的
    - 最优子结构和子问题重叠

    - 0-1背包
    - 重复求解
- 回溯：既有系统性又有跳跃性的搜索算法，以广度优先的方式搜索
    - 深度优先的方式搜索解空间
- 归纳：从测试所暴露的问题出发分析

9.数据库技术基础

> DBMS

把事务开始、事务结束以及数据库的插入删除修改的操作写入日志文件，

按一定时间间隔，把数据库缓冲区内容写入数据文件

概念数据模型 E-R

事务

- 原子性：要么都做，要么都不做
- 一致性：数据库只包含成功事务提交的结果
- 隔离性：事务相互隔离
- 持久性：一旦事务提交，即使数据库奔溃，对数据库的更新操作也将不会丢失

关系范式

- 1NF：属性不可拆分或无重复的列
- 2NF：完全函数依赖
- 3NF：消除传递依赖
- BC范式：所有非主属性对每一个码都是完全函数依赖

三级模式

- 外模式：描述用户看到的逻辑结构，视图
- 概念模式：用于描述整个数据库的逻辑结构，基本表
- 内模式：数据库底层表示，存储文件

有损联接且不保持函数依赖

数据模型三要素

- 数据结构
- 数据操作
- 完整性约束

分布式数据库

- 逻辑透明：
- 位置透明：不必知道数据存储哪个位置
- 分片透明：不必关心如何分片
- 复制透明：复制的数据由系统自动完成

10.网络与信息安全基础知识

ISO/OSI

层次	名称	主要功能	主要设备及协议
7	应用层	实现具体的应用功能	POP3、FTP、HTTP、Telnet、SMTP DHCP、TFTP、SNMP、DNS
6	表示层	数据的格式与表达、加密、压缩	
5	会话层	建立、管理和终止会话	
4	传输层	端到端的连接（端口）	TCP、UDP
3	网络层	分组传输和路由选择（IP）	三层交换机、路由器 ARP、RARP、IP、ICMP、IGMP
2	数据链路层	传送以帧为单位的信息（MAC）	网桥、交换机、网卡 PPTP、L2TP、SLIP、PPP
1	物理层	二进制传输（0/1）	中继器、集线器

- 应用层：应用软件
- 表示层：格式化和转化数据的服务
- 会话层：建立、维护、结束会话的功能
    - OSI/RM
- 传输层：传输服务，建立连接进行可靠通信
    - TCP: HTTP
    - UDP：SNMP协议的报文封装在UDP中
- 网络层：通信路径
    - 路由器
    - IP：无连接不可靠
    - ICMP：专门用于发送差错报文的协议
    - ARP、RARP：IP地址与物理地址互转
    - ARP协议常采用广播消息的方法来获取访问目标IP地址对应的MAC地址
- 数据链路层：数据帧
    - 网桥
    - 交换机
    - 网卡
- 物理层：物理链路
    - 中继器
    - 集线器

TCP/IP协议 合并应用、表示、会话为应用层

http请求

HTTP的一次请求过程 1.在浏览器中输入URL，并按下回车键； 2.浏览器向DNS服务器发出域名解析请求并获得结果； 3.浏览器向服务器发送数据请求； 4.根据目的IP地址和端口号， 与服务器建立TCP连接； 5.服务器将网页数据发送给浏览器； 6.通信完成，断开TCP连接； 7. 浏览器解析收到的数据并显示；

FTP服务器的控制端口为21（控制口），上传文件的端口为20（数据口），被动模式在1025-65535之间

利用漏洞扫描系统可以获取ftp服务器中是否存在可写目录的信息

DNS是一种分布式地址信息数据库系统，端口53

主域名服务器在接收到域名请求后：本地缓存->域名服务器

- ipconfig

    - release：释放
    - flushdns：清除
    - displaydns：显示
    - registerdns：注册
- netstat：获取某个网络开放端口所对应的应用程序信息，不能诊断DNS故障
- nslookup：查询DNS的记录，诊断网络问题

HTTPS默认端口443

Telnet远程登录服务

电子邮件协议SMTP:25,发信服务器的协议

POP3:110, 收信服务器

IMAP

提供安全电子邮件服务S/MIME,PGP

包过滤技术对应用和用户是透明的

主动攻击：拒绝服务器攻击，分布式拒绝服务，信息篡改，资源使用，重放，伪装

被动攻击：窃听，业务流分析，电磁或射频截获，系统干涉

TCP/IP通信协议分为4层，自下而上：网络接口层->互联网层->传输层->应用层

实现VPN的技术有隧道技术、加解密技术、密钥管理技术和身份认证技术

使用IPSec为IP数据报文进行加密

SQL注入攻击的首要目标是获得数据库权限

网络系统中，通常把Web服务器置于DMZ区

11.标准化和软件知识产权基础知识

计算机软件著作权的保护对象是指计算机程序及其文档

商标权的保护期可以延长,可能无限拥有。

软著产生的时间是软件**开发完成时**

利用商业秘密权可以对软件的技术信息、经营信息提供保护

12.软件系统分析与设计

13.专业英语

Only when one writes do the gaps appear and the (`inconsistencies`) protrude. 不一致

Since his fundamental job is to keep everybody going in the (`same`) direction,his chief daily task will be communication.

Fail to capture the (`dynamics`) of the system`s behavior. 动力

To a user,a program with just the right features presented through an intuitive and (`simple`) interface is beautiful.

This way,even if the more ambitious research efforts should fail,there will be at least (`partial`) positive outcomes.

The complexity of software is an (`essential`) property,not an accidental one.

The object constructed in the requirement analysis shows the (`static structure`) of the real-world system and organizes it into workable pieces.

The elements interact with each other in some (`nonlinear`) fashion.

They think that beauty is (`impractical`).

Please planning is creating a game plan for your Web project (`outlining`) what you think you want your Web site to be.概述

It also (`defines`) how long the project will take and how much it will cost.

The (`tree-structure`) allows distribution lists to be maintained by subtree,if that is desirable. 树形结构

Once a (`standard`) has been established,many more groups and companies will adopt it.

Although this may sound simple,the fact is that (`conventional`) requirement capture approaches. 依照惯例

The data (`management`) includes the function of updating data on a database,and retrieving data from a database.

(`domain analysis`) focuses on real-world things whose semantics the application captures.

But,like anything,use cases come with their own problems,and as useful as they are,they can be (`misapplied`) 误用

Control of the distribution of (`information`).

Software systems have orders of magnitude more (`states`) than computers do.

we make the two similar parts into one,a (`subroutine`),open or closed. 子程序

The three types of the analysis model are (class model,interaction model and state model).类模型、交互模型、状态模型
The ability to run a program or application.

Systems development is a team (`activity`),and the effectiveness of the team largely determines the (`quality`) of the engineering.

You use a release plan to do an initial sanity check of the project's (`feasibility`) and worthiness. 可行性

This (`competition`) of ideas is a major driving force for scientific progress. 竞争

Add-on security packages can add to the (`automated`) access control capabilities of the OS. 自动化

The infrastructure layer serves as the (`hardware`) for building the platform layer of the cloud.

This small set of critical documents (`decides`) much of the manager`s work.