计算机基础与综合知识

内容

计算机组成、操作系统、网络、嵌入式，以及数学与专业英语等公共基础知识。

CPU组成与体系结构

内容

CPU由运算器、控制器、寄存器和高速缓存组成。运算器负责算术和逻辑运算；控制器负责指令的读取、译码和执行；寄存器提供高速数据存储。CISC和RIBB是两种主要的指令集架构。流水线技术通过将指令执行过程分成多个阶段来提高CPU利用率。

考点

1.运算器包含ALU、累加器、状态寄存器
2.控制器包含程序计数器PC、指令寄存器IR、时序发生器
3.流水线相关：吞吐率=指令数/(流水线级数+n-1) x 时钟周期
4.CPI=时钟周期数/指令数，MIPS=主频/CPI x 10^6

易错点

1.PC存放的是下一条指令地址，不是当前指令
2.流水线冲突包括资源冲突、数据冲突、控制冲突
3.超标量和超流水线是不同的概念

存储系统

内容

存储系统采用多级层次结构：寄存器-Cache-主存-辅存。Cache-主存地址映射有直接映射、全相联映射和组相联映射三种方式。虚拟存储器通过页式、段式或段页式管理实现逻辑地址到物理地址的转换。

考点

1.Cache命中率公式：h=命中次数/(命中次数+缺失次数)
2.平均访问时间=h x tc+(1-h) x tm
3.虚拟地址=虚页号+页内偏移
4.位示图大小=磁盘块数/字长
5.页面置换算法：OPT、FIFO、LRU、Clock

易错点

1.直接映射每个主存块只能映射到一个Cache行
2.全相联映射每个主存块可映射到任意Cache行
3.组相联是直接映射和全相联的折中
4.LRU是最近最少使用，不是最近最久未使用

数学与经济管理

内容

系统架构师考试涉及的数学知识包括：概率统计、线性代数、离散数学、算法分析。经济管理知识包括：盈亏平衡分析、投资回报率、成本效益分析。

考点

1.概率：条件概率P(A|B)=P(AB)/P(B)，贝叶斯公式
2.排列组合：P(n,k)=n!/(n-k)!，C(n,k)=n!/(k!(n-k)!)
3.算法复杂度：O(1)<O(logn)<O(n)<O(nlogn)<O(n^2)<O(2^n)
4.盈亏平衡点：固定成本/(单价-单位可变成本-单位税金)
5.投资回报率ROI=(收益-成本)/成本x100%

易错点

1.贝叶斯公式：P(A|B)=P(B|A)xP(A)/P(B)
2.马尔科夫链：当前状态只与前一状态有关
3.快速排序平均O(nlogn)，最坏O(n^2)
4.动态规划最优子结构和重叠子问题

通信与网络

内容

网络通信基础：数据传输方式(串行/并行)、同步方式(同步/异步)、通信模式(单工/半双工/全双工)。网络设备：集线器、交换机、路由器、网关。

考点

1.集线器(Hub)：物理层设备，共享带宽
2.交换机(Switch)：数据链路层设备，MAC地址转发
3.路由器(Router)：网络层设备，IP地址路由
4.网关(Gateway)：高层协议转换
5.网络地址转换NAT：私有IP转公网IP
6.VLAN：虚拟局域网，逻辑隔离广播域
7.SDN软件定义网络：控制平面与数据平面分离

易错点

1.交换机通过MAC地址表转发帧
2.路由器通过路由表转发IP包
3.NAT解决IPv4地址不足问题
4.VLAN隔离广播域但不能隔离碰撞域
5.三层交换机兼具路由和交换功能

指令流水线技术

内容

流水线将指令执行划分为取指、译码、执行、访存、写回等阶段并重叠执行，以提高指令吞吐率，理想情况下每个时钟周期完成一条指令。流水线会因数据相关、控制相关和结构相关产生冒险而停顿。

考点

1.流水线周期=执行时间最长的一段
2.吞吐率=指令数/总时间；加速比=顺序时间/流水线时间
3.流水线时间=(t1+t2+...+tk)+(n-1)*流水线周期
4.三类冒险：结构相关、数据相关、控制相关
5.解决：转发/旁路、分支预测、乱序执行

易错点

1.流水线周期取各段中的最大值，不是平均值
2.超标量一个周期发射多条指令；超流水细分更多段
3.分支跳转会导致流水线清空(控制冒险)

校验与纠错码

内容

校验码通过增加冗余位检测或纠正数据传输中的错误。奇偶校验只能检测奇数位错误，CRC 循环冗余校验广泛用于通信和存储，海明码可检测并纠正一位错误。码距决定检错纠错能力。

考点

1.码距 d：可检测 d-1 位错，纠正 (d-1)/2 位错
2.奇偶校验：码距 2，只检错不纠错
3.CRC：模 2 除法生成校验位，检错能力强，不纠错
4.海明码：校验位满足 2^r>=m+r+1，可纠 1 位错
5.应用：内存 ECC 用海明码，网络帧用 CRC

易错点

1.海明码校验位插在 2 的幂次位置(1,2,4,8)
2.CRC 只能检错不能纠错
3.码距越大检错纠错能力越强

操作系统

操作系统概述

内容

操作系统的定义：能有效地组织和管理系统中的各种软/硬件资源，合理地组织计算机系统工作流程，控制程序的执行，并向用户提供一个良好的工作环境和友好接口。

进程管理

进程的组成及状态转换

内容

进程的组成

进程由程序、数据和进程控制块（PCB）组成。

三态模型

运行态、就绪态和阻塞态。

五态模型

运行态、就绪态、阻塞态、新建态和终止态。

图 1.四态和五态模型

进程间的通信

内容

多个并发进程存在资源共享和相互合作，因此进程之间需要进行通信。

进程调度

内容

- 先来先服务（FCFS）：按先后次序分配CPU，有利于长作业，不利于短作业；有利于CPU繁忙型作业，不利于I/O繁忙型作业。
- 时间片轮转：分配给每个进程时间片，轮流占用CPU，通过时间片轮转能高效利用资源和响应时间特性，从而提高资源利用率。
- 优先级调度：系统选择优先级大的进程占用CPU。
- 多级反馈调度：时间片轮转和优先级调度结合形成。设置多个就绪队列，1，2，3...n，每个队列分别赋予不同的优先级，队列1最高，队列n最低，分配不同的时间片长度，新进程先进入队列1的末尾，按FCFS原则，执行队列1的时间片；若未期间内完成，则输入队列2的末尾，依次类推。

死锁

内容

死锁：指两个以上的进程因相互争夺对方占用的资源而陷入无限等待的现象。

死锁产生的4个必要条件：

- 互斥（资源互斥）
- 请求保持（进程占有资源并等待其他资源）
- 不可剥夺（系统不能剥夺进程资源）
- 环路（进程资源周期是一个环路）

死锁的处理策略：

- 死锁预防：破坏死锁产生的4个必要条件。
- 死锁避免：银行家算法。
- 死锁检测：检测死锁并解决以后，记法加以解除。
- 死锁解除：死锁发生以后解除的方法，如资源剥夺、撤销进程法等。

系统内有n个进程，每个进程都需要m个资源，不发生死锁的最小资源数为 `m * (n - 1) + 1`。

线程

内容

线程是进程中的一个实体，是可独立分配和调度的基本单位，线程可以共享进程的公共数据，全局变量，代码和一个进程拥有的资源（如打开文件和信号）等，但不能共享线程独有的资源，如线程的栈指针等标示数据。

存储管理

页面置换算法

内容

- 最佳置换算法：理想化的算法，选择最长时间不再被访问的页面置换。
- 先进先出置换算法：淘汰最先进入主存的页面，即选择在主存中驻留时间最久的页面予以淘汰。
- 最近最少未使用置换算法：选择最近最少未使用的页面予以淘汰，根据局部性原理，这种方式效率较高。
- 最近未用置换算法：优先淘汰最近未访问的，而近海决最近未被修改的页面。

分页存储管理

内容

段页式存储管理：

先将进程分为若干个大小相等的分块，将用户程序按程序的逻辑关系分为若干个段，再将每个段划分成若干个页。

段页式的地址结构：段号+页内页号+页内地址w

段页式存储的优缺点：

- 优点：空间浪费小、存储共享容易、存储保护容易、易动态连接。
- 缺点：由于管理软件的增加，易导致程序易随之加长，需要的硬件以及占用的内存也相对增加，使得执行速度大大下降。

设备管理

内容

- 通道技术：CPU只需向通道发出I/O命令，通道收到命令后，从主存中取出本次I/O要执行的通道程序并执行，在完成要求的I/O任务后才向CPU发出中断信号。目的：使数据传输独立于CPU，和CPU从前端的I/O工作中解脱出来。
- DMA技术(直接内存存取（DMA）：是指数据在主存与I/O设备之间直接成块传送，即在主存与设备间传送一个数据块的过程不需要CPU的任何干涉。
- 缓冲技术：包括硬件缓冲（硬件寄存器）和软件缓冲（软件系统），是为了缓和许业利用率，尽可能使许业处于繁忙状态。引入缓冲技术的原因：①缓和CPU与I/O设备之间速度不匹配的矛盾。②减少对CPU的中断频率。放宽对中断响应时间的限制。③提高CPU和I/O设备之间的并行性。
- Spooling技术：假脱机技术，用一块物理设备模拟另一类物理设备的技术。是使独占使用的设备变成多台虚拟设备的一种技术。

文件管理

文件的结构和组织

内容

三级索引文件结构：0-9为直接索引，即每个索引结点存放的是数据；10为一级间接索引；11为二级间接索引。

文件目录

内容

文件控制块（FCB）：用来存放控制文件需要的各种信息的数据结构，以实现"按名存取"。FCB的有序集合称为文件目录。一个FCB就是一个文件目录项。

FCB主要包含以下信息：

- 基本信息：如文件名、文件的物理地址、文件的长度和文件种类。
- 存储控制信息：如文件读取权限等。
- 使用信息：如文件建立时间、修改时间等。

存储方法和存储空间的管理

空闲区表

内容

分区号	第一个空闲盘块号	空闲盘块数	状态
1	14	4	可用
2	20	8	可用
3	100	19	可用
...	...	...	未用

位示图

内容

在许多大建立一张位示图，0标示空闲，1标示占用。假设系统字长为32位，位示图中的每个二进制对应一个物理块，因此可以用（字号，位号）对应一个物理块。

空闲块链

内容

每个空闲物理块有指向下一个空闲物理块的指针，所有空闲物理块构成一个链表。

空闲物理块
空闲物理块
空闲物理块

数据库设计

软考通：系统架构设计师 知识导图：共 310 个知识点，第 3/15 页

数据库设计

内容

数据库系统原理、规范化、事务开发、分布式数据库、数据仓库与大数据存储等。

数据库基本概念

数据库三级模式

内容

数据库的三级模式：

- 外模式：物理结构和存储方式
- 模式：表
- 外模式：视图

数据库三级模式

数据库设计

内容

数据库设计6个阶段：

- 用户需求分析
- 概念结构设计
- 逻辑结构设计
- 物理结构设计
- 数据库实施阶段
- 数据库运行和维护阶段

用户需求分析
用户需求
概念结构设计
概念模型
逻辑结构设计
逻辑模型
物理结构设计
物理模型
数据库实施阶段
数据库
数据库管理系统
数据库管理员
数据库运行和维护阶段
数据库管理员
用户使用数据库
数据库管理员
维护数据库
数据库管理员
数据库管理员
数据库管理员
数据库管理员
数据库管理员

嵌入式系统及软件

内容

嵌入式系统是以应用为中心、以计算机技术为基础的专用计算机系统。特点：专用性、实时性、可靠性、资源受限。实时操作系统(RTOS)提供任务调度、中断管理、内存管理等功能。

考点

1.嵌入式处理器：MCU、MPU、DSP、SoC
2.实时操作系统特点：确定性、可抢占、低延迟
3.任务调度：优先级调度、时间片轮转、Rate Monotonic
4.交叉编译：在宿主机上编译目标平台的程序
5.BSP板级支持包：硬件抽象层
6.BootLoader启动流程：初始化硬件-加载内核-启动系统

易错点

1.硬实时和软实时的区别
2.优先级反转问题及优先级继承解决方案
3.嵌入式系统功耗管理
4.看门狗定时器的作用

嵌入式系统的组成及特点

内容

嵌入式系统以应用为中心、以计算机技术为基础，需要满足应用对**功能、可靠性、成本、体积和功耗**等方面的严格要求。

嵌入式系统的特点

1. 专用性强。嵌入式系统面向特定应用需求，能够把通用CPU中许多由板卡完成的任务集成在芯片内部，从而有利于嵌入式系统的小型化。
2. 技术融合。嵌入式系统将先进的计算机技术、通信技术、半导体技术和电子技术与各个行业的具体应用相结合，是一个技术密集、资金密集、高度分散、不断创新的知识集成系统。
3. 软硬一体软件为主。软件是嵌入式系统的主体，有IP核。嵌入式系统的硬件和软件都可以高效地设计，量体裁衣，去除余，可以在同样的硅片面积上实现更高的性能。
4. 比通用计算机资源少。由于嵌入式系统通常只完成少数几个任务。设计时考虑到其经济性，不能使用通用CPU，这就意味着管理的资源少，成本低，结构更简单。
5. 程序代码固化在非易失存储器中。为了提高执行速度和系统可靠性，嵌入式系统中的软件一般都固化在存储器芯片或单片机本身中，而不是存在磁盘中。
6. 需专门开发工具和环境。嵌入式系统本身不具备开发能力，即使设计完成以后，用户通常也不能对其中的程序功能进行修改，必须有一套开发工具和环境才能进行开发。
7. 体积小、价格低、工艺先进、性能价格比高、系统配置要求低、实时性强。
8. 对安全性和可靠性的要求高。

嵌入式系统的分类

内容

嵌入式操作系统分类：

- 嵌入式实时操作系统
  - 强实时操作系统
  - 弱实时操作系统
- 嵌入式非实时系统

从安全性要求来看，也可以分为：

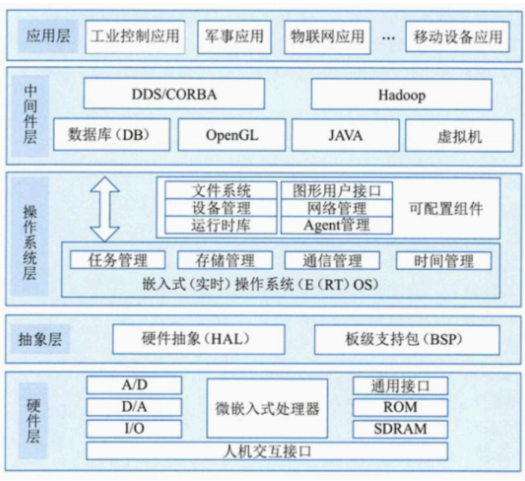
- 安全攸关系统
- 非安全攸关系统

实时系统：能够在指定或者确定的时间内完成系统功能和外部或内部、同步或异步时间做出响应的系统。
安全攸关系统：指不正确的功能或失效会导致人员伤亡、财产损失等严重后果的计算机系统。

嵌入式软件的组成及特点

内容

嵌入式系统组成架构



BootLoader一般包括以下功能：

- 片级初始化：完成微处理器初始化，包括设置微处理器的核心寄存器和控制寄存器、微处理器和核心工作模式及其局部总线模式等。
- 板级初始化：正确地设计各种寄存器地内容来完成微处理器以外的其他硬件设备的初始化。例如初始化LED显示设备。
- 加载内核（系统级初始化）：将操作系统和应用程序的映像从Flash存储器复制到系统的内存中，然后跳转到系统内核的第一条指令处继续执行。

考点

BSP是介于主板硬件和操作系统中驱动层程序之间的一层。
BSP主要包括两个方面的内容：引导加载程序BootLoader和设备驱动程序。

嵌入式操作系统的特点：

- 可裁剪性
- 可配置性
- 强实时性
- 安全性
- 可靠性
- 高确定性

嵌入式系统软件架构原理与特征

内容

嵌入式操作系统的典型架构：

- 层次化架构
- 递归模式架构

计算机网络

内容

计算机网络按OSI七层模型或TCP/IP四层模型组织。核心协议包括：物理层（编码）、数据链路层（MAC、PPP）、网络层（IP、ICMP、ARP）、传输层（TCP、UDP）、应用层（HTTP、DNS、SMTP）。

考点

1.TCP三次握手：SYN-SYN+ACK-ACK
2.IP地址分类：A(1-126)、B(128-191)、C(192-223)、D(224-239)、E(240-255)
3.子网掩码和CIDR
4.路由协议：RIP(距离向量)、OSPF(链路状态)、BGP(路径向量)
5.网络安全：对称加密(AES)、非对称加密(RSA)、数字签名、PKI

易错点

1.ARP是IP到MAC地址解析
2.RIP最大跳数15，OSPF使用链路状态算法
3.TCP是面向连接的可靠传输，UDP是无连接的不可靠传输
4.VLAN通过标签隔离广播域
5.子网划分：主机位借位给网络位

网络概述

网络的分类

内容

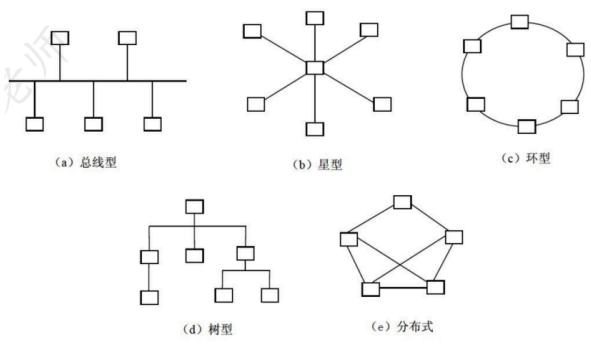
按通信距离可分为：局域网、城域网和广域网。

网络分类	缩写	分布距离	计算机分布范围	传输速率范围
局域网	LAN	10m左右	房间	4Mbps~1Gbps
		100m左右	楼宇	
		1000m左右	校园	
城域网	MAN	10km	城市	50Kbps~100Mbps
广域网	WAN	100km以上	国家或全球	9.6Kbps~45Mbps

网络拓扑结构

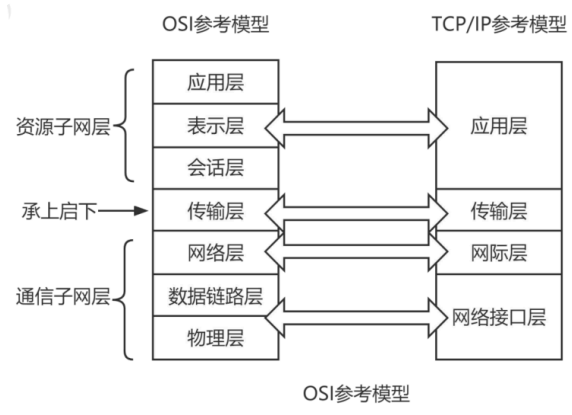
内容

常用拓扑结构



ISO/OSI网络体系结构

内容



- 应用层：面向用户服务，提供网络和用户应用软件之间的接口服务，如文件传输、电子邮件、远程登录等。
- 表示层：提供格式化的表示和转换数据服务，如数据的压缩、解压缩、加密和解密等。
- 会话层：提供会话管理功能。传输的单位是报文。
- 传输层：提供端到端的数据传输服务。功能有：数据分段、数据传输、数据组装、差错控制、流量/拥塞控制等，传输的数据单位是报文。
- 网络层：将数据链路层的帧数据组成数据包，提供路由选择、差错检测和恢复、流量/拥塞控制、激活和终止网络连接等功能。传输的数据单位是数据包。
- 数据链路层：传输帧数据，传输的数据单位是帧。
- 物理层：为数据链路层提供物理连接。传输的数据单位是比特。

网络互连硬件

网络的设备

内容

协议层	互连设备	作用
物理层	中继器 集线器	①中继器：信号在介质里传输的过程中会有衰减和噪声，使用中继器进行放大和去噪； ②集线器：是一种特殊的多路中继器。
数据链路层	网桥 交换机	①网桥：连接两个局域网，检查帧的源地址和目的地址，若两者在同一网络段上，则不转发，否则转发到另一个网络段上； ②交换机：检查以太网帧的目的地址MAC，在自己的地址表（端口号-MAC）中进行查找并转发。
网络层	路由器	用于连接多个逻辑上分开的网络，最主要的功能是选择路由路径。
应用层	网关	将协议进行转换，将数据重新分组，以便在两个不同类型的网络系统之间进行通信。

网络的传输介质

内容

介质类型	介质	说明
有线介质	双绞线 同轴电缆 光纤	①双绞线：分为3类、4类和5类、6类、7类双绞线，常用的是5类非屏蔽双绞线，频率带宽为100MHz，最大长度为100m； ②同轴电缆：分为基带同轴电缆和宽带同轴电缆； ③光纤：光导纤维，多模光纤使用发光二极管作为光源，价格便宜，传播距离短，带宽小；而单模光纤使用激光作为光源，价格昂贵、传播距离长、带宽大。
无线介质	微波 红外线和激光 卫星	①微波：使用无线电波通信，受天气、环境、地形影响大； ②红外线和激光：把传输信号转换成红外光信号和激光信号，在空间传播，受环境气候影响大； ③卫星：以人造卫星作为微波中继站，容量大、距离远，但延迟高。

网络的协议与标准

内容

ISO/OSI参考模型　TCP/IP协议　TCP/IP模型

应用层	文件传输协议(FTP)	远程登录协议(Telnet)	电子邮件协议(SMTP)	网络文件服务协议(NFS)	网络管理协议(SNMP)	应用层
表示层						
会话层						
传输层	TCP		UDP			传输层
网络层	IP	ICMP	ARP	RARP		网际层
数据链路层	Ethernet IEEE 802.3	FDDI	Token-Ring/IEEE 802.5	ARCnet	PPP/SLIP	网络接口层
物理层						硬件层

网网际层协议

内容

网际层协议	功能
IP	提供无连接、不可靠的数据传送服务
ICMP（Internet控制信息协议）	ICMP就是一个专门用于发送差错报文的协议。ICMP定义了五种差错报文（源抑制、超时、目的不可达、重定向和要求分段）和四种信息报文（回应请求、回应应答、地址屏蔽码请求和地址屏蔽码应答）。
IGMP（网络组管理协议）	主要用于组播，主机可以通过IGMP告诉路由器想接收或离开某个网络组播的信息。
ARP、RARP	地址解析协议ARP的作用是将IP地址转换为物理地址（MAC地址）； 反地址解析协议RARP的作用是将物理地址（MAC地址）转换为IP地址。

传输层协议

内容

TCP：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。
UDP：是一种不可靠、无连接的协议。

应用层协议

内容

Internet服务	说明
DNS域名服务	DNS域名解析协议，域名服务器将域名转换为IP地址，使用UDP端口，端口号为53
远程登录服务	Telnet协议，使用TCP端口，端口号为23
电子邮件(E-mail)服务	简单邮件传输协议SMTP（简单邮件发送协议，使用TCP端口，端口号为25）， POP3（邮件接收协议，使用TCP端口，端口号为110）， 多用途邮件扩充协议MIME、增强私密邮件保护协议PEM
WWW服务	超文本传输协议HTTP，使用TCP端口，端口号为80。 Web地址的URL：scheme://host.Domain[:port]Upath/filename 示例：http://www.dailnews.com.cn/channel/welcome.html
文件传输服务	FTP协议，使用TCP端口，有两种TCP连接： ①控制连接，传输口令和参数，端口号为21； ②数据连接，传输文件，端口号为20。

Internet服务	说明
网络文件服务	网络文件服务协议NFS，使用UDP端口，端口号为2049，允许客户端可以像本地文件系统一样，访问网络上计算机之间共享的文件和目录。
网络管理服务	简单网络管理协议SNMP，使用UDP端口，端口号：①用于数据传送与接收的161端口；②用于报警信息接收的162端口。它允许网络管理员监视和管理网络设备、服务器、路由器和其他网络设备的运行状态和性能。
DHCP服务	动态主机配置协议DHCP，主要用于在局域网内为设备动态分配IP地址，它基于UDP协议，并使用67和68端口进行通信。

	说明	信息传输	端口
HTTP	超文本传输协议	明文传输	80
HTTPS	HTTPS协议是由HTTP协议+SSL证书构建的可进行加密传输、身份认证的网络协议，要比HTTP协议安全。	具有安全性的SSL加密传输协议	443

组网技术

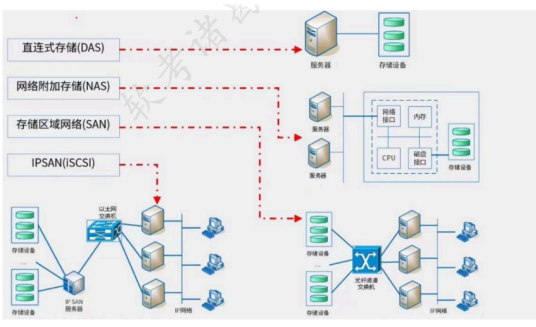
内容

路由原理：路由器工作在OSI七层协议中的第3层，即网络层。

网络存储技术

内容

- 直连式存储（DAS）
- 网络连接存储（NAS）
- 存储区域网络（SAN）



系统配置与性能评价

性能指标

计算机的性能指标

内容

时钟频率(主频)；运算速度；运算精度；内存的存储容量；存储器的存取周期；数据处理速率PDR(processingdatarate)；吞吐率；各种响应时间；各种利用率；RASIS特性(即：可靠性Reliability、可用性Availability、可维护性Sericeability、完整性和安全性Integraity and Security)；平均故障响应时间；兼容性；可扩充性；性能价格比。

操作系统的性能指标

内容

系统的可靠性、系统上下文切换、系统响应时间、系统的吞吐率、系统资源利用率、可靠性和可移植性。

数据库管理系统的性能的指标

内容

数据库本身和管理系统两部分，有数据库大小、数据库中表的数量、单个表的大小、表中允许的记录（行）数量、单个记录（行）的大小、表上所允许的索引数量、数据库所允许的索引数量、最大并发事务处理能力、负载均衡能力、最大连接数等。

WEB服务的性能指标

内容

最大并发连接数、响应延迟和吞吐量。

性能计算

内容

- 时钟频率：一般来讲，主频越高，速度越快。
- 指令执行速度：计量单位KIPS、MIPS。
- 等效指令速度法：统计各类指令在程序中所占比例，并进行折算，是一种固定比例法。
- 数据处理速率法(Processing Data Rate，PDR)：采用计算PDR值的方法来衡量机器性能，PDR值越大，机器性能越好。PDR与每条指令和每个操作数的平均位数以及每条指令的平均运算速度有关。

性能设计

内容

阿姆达尔定律：

- 加速比 = 不使用增强部件时完成整个任务的时间 / 不使用增强部件时完成整个任务的时间
- 总加速比 = 原来的执行时间 / 新的执行时间 - 1/（（1 - 增强比例） + 增强比例 / 增强加速比）

性能评估

内容

基准测试程序

把应用程序中用得最多、最频繁的那部分核心程序作为评价计算机性能的标准程序，称为基准测试程序。

基准程序，包括:

- 整数测试程序。同一厂家的机器，采用相同的体系结构，用相同的基准程序测试，得到的MIPS值越大，一般说明机器速度越快。
- 浮点测试程序。指标MFLOPS(理论峰值浮点速度)。
- SPEC基准程序(SPEC Benchmark)。重点面向处理器性能的基准程序集，将被测计算机的执行时间标准化，即将被测计算机的执行时间除以一个参考处理器的执行时间。
- TPC基准程序。用于评测计算机在事务处理、数据库处理、企业管理与决策支持系统等方面的性能。其中，TPC-C是在线事务处理(On-line Transaction Processing，OLTP)的基准程序，TPC-D是决策支持的基准程序。TPC-E作为大型企业信息服务的基准程序。

WEB服务器的性能评估

WEB服务器的指标有：最大并发连接数、响应延迟和吞吐量等。

WEB服务器性能评测方法有：基准性能测试、压力测试和可靠性测试。

考点

性能测试的准确程序依次递减：真实的程序、核心程序、小型基准程序、合成基准程序。

系统监视的方法：

- 一是通过系统本身提供的命令，如UNIX/Linux中的W、ps、last，Windows中的netstat等；
- 二是通过系统记录文件查阅系统在特定时间内的运行状态；
- 三是集成命令、文件记录和可视化技术，如Windows的Perfmon应用程序。

软件工程基础知识

作者：Xmind中文官方社区 | 来源：Xmind中文官方社区 | 最后更新：2019-12-31

软件工程概述

软件开发的基础概念

定义

软件是计算机硬件、程序和支持软件开发和维护的文件的集合。

- 软件开发模型是指软件开发过程的一种抽象表示，它定义了软件开发过程中各个阶段的顺序和活动。
- 软件开发方法是一种系统化的方法论，它为软件开发提供了一系列的原则、技术和工具。
- 软件开发过程是一系列活动的集合，用于将需求转化为可运行的软件产品。

软件开发过程

定义

- 软件开发产品：指软件开发过程中，由开发人员、编辑和测试人员生成的文件。
- 软件开发活动：指软件开发过程中，为生成软件开发产品所采取的一系列行动。
- 软件开发模型的作用：是一种抽象概念，表现软件开发过程，强调软件开发过程的顺序、步骤和活动，以及产品的特征。

软件工具和环境

- 软件开发工具
- 软件开发环境
- 软件工程
- 软件开发过程

软件生命周期

定义

软件生命周期是指软件从开始规划、设计、开发、测试、部署、维护、升级直到最终退役的整个过程。这个过程通常包含多个阶段，每个阶段都有特定的目标和活动，以确保软件能够满足用户需求并保持其价值。

- 软件规划：确定项目目标、范围、成本和资源。
- 软件需求分析：收集和分析用户需求，确定软件应该做什么。
- 软件设计：根据需求分析结果，设计软件的架构、模块、接口和数据结构。
- 软件编码：将设计转化为可执行的代码。
- 软件测试：对软件进行各种测试，以发现和修复错误。
- 软件部署：将软件安装到生产环境，使用户能够使用。
- 软件维护：对软件进行修复、改进和升级，以适应新的需求和环境。
- 软件退役：当软件不再有价值或无法维护时，将其从生产环境中移除。

软件开发过程模型

定义

软件开发过程模型是软件开发过程的抽象表示，它定义了软件开发过程中各个阶段的顺序、活动和交付物。不同的模型适用于不同的项目类型和开发环境，以确保软件开发过程的有效性和可控性。

- 瀑布模型：线性、顺序的开发模型，适用于需求明确、变化较小的项目。
- 迭代模型：将软件开发过程分解为多个迭代，每个迭代都包含需求分析、设计、编码、测试和评审。适用于需求不明确、变化较大的项目。
- 螺旋模型：结合了瀑布模型和迭代模型的优点，强调风险管理。适用于大型、复杂的项目。
- 敏捷模型：一种迭代模型的变体，强调快速响应变化、客户参与和团队协作。适用于需求变化频繁、需要快速交付价值的项目。
- 原型模型：通过构建原型来验证需求和设计，适用于需求不明确、需要用户反馈的项目。
- 喷泉模型：一种迭代模型的变体，强调软件的持续演化和维护。适用于需求稳定、但需要持续改进和维护的项目。
- 大爆炸模型：一种极端的模型，将所有需求一次性实现，风险极高。适用于需求极其明确、风险可控的项目。

瀑布模型

定义

瀑布模型是一种线性、顺序的软件开发模型，它将软件开发过程分解为一系列阶段，每个阶段完成后才能进入下一个阶段。这种模型强调文档和计划，适用于需求明确、变化较小的项目。

瀑布模型

螺旋模型

定义

螺旋模型是一种结合了瀑布模型和迭代模型特点的软件开发模型，它强调风险管理。适用于大型、复杂的项目，尤其是需求不明确、风险较高的项目。

螺旋模型

敏捷模型

定义

敏捷模型是一种迭代模型的变体，它强调快速响应变化、客户参与和团队协作。适用于需求变化频繁、需要快速交付价值的项目。

敏捷模型

原型模型

定义

原型模型是一种迭代模型，它通过构建原型来验证需求和设计。适用于需求不明确、需要用户反馈的项目。

原型模型

喷泉模型

定义

喷泉模型是一种迭代模型的变体，它强调软件的持续演化和维护。适用于需求稳定、但需要持续改进和维护的项目。

喷泉模型

大爆炸模型

定义

大爆炸模型是一种极端的模型，将所有需求一次性实现，风险极高。适用于需求极其明确、风险可控的项目。

大爆炸模型

软件开发过程模型的选择

定义

选择软件开发过程模型时，需要考虑项目的规模、复杂度、风险、资源、时间、质量要求、团队经验、客户参与度、变化频率、技术成熟度、法规要求、预算、维护成本、可扩展性、可重用性、可移植性、可维护性、可测试性、可理解性、可交付性、可监控性、可管理性、可度量性、可证明性、可验证性、可信性、可靠性、可用性、可安全性、可保密性、可隐私性、可合规性、可审计性、可追溯性、可记录性、可文档性、可沟通性、可协作性、可学习性、可教学性、可推广性、可商业性、可盈利性、可持续性、可社会性、可伦理性、可责任性、可问责性、可信任性、可声誉性、可品牌性、可文化性、可艺术性、可美学性、可人性化、可用户体验性、可满意性、可忠诚性、可口碑性、可推荐性、可传播性、可扩散性、可影响性、可改变性、可颠覆性、可创新性、可发明性、可专利性、可知识产权性、可版权性、可商标性、可设计性、可工业性、可制造性、可生产性、可供应性、可分销性、可零售性、可消费性、可使用性、可维护性、可升级性、可淘汰性、可回收性、可再利用性、可再制性、可再生性、可可持续性、可绿色性、可环保性、可节能性、可节水性、可节材性、可节地性、可节人性、可节时性、可节钱

面向对象技术

面向对象基础

面向对象的基本概念

内容

- 封装：将数据和行为组合在一个类中，并且对外隐藏了实现的细节
- 继承：是一种实现代码重用的方式
- 多态：允许相同的消息传递给不同的对象得到不同的行为结果
- 重载：一个类中定义了相同名称但参数列表不同的方法

面向对象分析

内容

面向对象（OOD）的目的是获得对应用问题的理解，确定问题域。

在OOD中，类可以分为3种类型：实体类、控制类、边界类。

- 实体类：保存需要存储在数据库中的信息
- 控制类：控制用例工作的类，例如：身份验证器
- 边界类：系统或门与系统外部进行交互

在用例模型中，每个参与者和用例交互至少要有一个边界类。

面向对象设计

内容

面向对象的分析模型主要由顶层架构图、用例与用例图、领域概念模型构成。

设计模式包含以包图标识的软件体系结构图，以及互表示的用例实现图、完整精确的类图、针对复杂对象的状态图和用以描述流程状处理过程的活动图。

面向对象设计

内容

面向对象设计原则：

- 单一职责
- 开放封闭原则
- 替换原则
- 依赖倒置原则
- 接口分离原则
- 迪米特法则

UML

内容

UML是面向对象软件的标准化建模语言

由三个要素构成：

- 基本构造快
- 图
- 公用机制

UML中的图

内容

UML图总览

类图	描述一组对象、接口、协作和它们之间关系的静态结构图。类图中通常包含各种类、接口、协作、关系、依赖、泛化、关联、实现等。UML 1.x 中，类图是最复杂的图，UML 2.x 中，类图仍然是最复杂的图。
对象图	描述一组对象及它们之间关系的图，对象图描述了在类图中所建立的事物的实例的静态快照。对象图从实践角度来说，是从实际的角度来考虑问题，是类图的一个实例。
组合结构图	描述一个组合结构的内部结构，包括在此结构内部的部分，以及它们之间的关系，以及这些部分可以与外部世界交互的端口。组合结构图是UML 2.x 中新增的图。
包图	描述了系统在包层面上的结构设计，用来表示包和包之间的依赖关系。包图是UML 2.x 中新增的图。
用例图	描述了一组用例、参与者以及它们之间的关系。用例图是用来对系统的功能进行建模，是从用户的角度来看待系统，是从外部来观察系统的一个视图。
顺序图	描述对象之间的交互（消息的发送与接收）。顺序图是用来对系统的动态方面进行建模，是从时间的角度来考虑问题，是一个交互图。
通信图	描述了对象之间的交互，强调对象的结构组织。通信图是用来对系统的动态方面进行建模，是从空间的角度来考虑问题，是一个交互图。
交互概览图	描述了一组顺序图、通信图、活动图之间的关系。交互概览图是用来对系统的动态方面进行建模，是从时间的角度来考虑问题，是一个交互图。
时序图	描述了系统对象的状态随时间变化的情况。时序图是用来对系统的动态方面进行建模，是从时间的角度来考虑问题，是一个交互图。
活动图	描述了一组顺序图、通信图、交互概览图之间的关系。活动图是用来对系统的动态方面进行建模，是从时间的角度来考虑问题，是一个交互图。
状态图	描述了一个对象在其生命周期中的状态变化情况。状态图是用来对系统的动态方面进行建模，是从时间的角度来考虑问题，是一个交互图。
部署图	描述了系统的硬件拓扑结构以及在这个结构上运行的软件。部署图是用来对系统的实现方面进行建模，是从空间的角度来考虑问题，是一个实现图。
构件图	描述了系统的物理结构，包括可执行文件、库、文件、数据表、文档等。构件图是用来对系统的实现方面进行建模，是从空间的角度来考虑问题，是一个实现图。

类图

内容

描述类、类的特性以及类之间的关系

用例图

内容

描述了一组用例、参与者以及它们之间的关系。

用例之间包含三种关系：

- 包含关系：当可以从两个或两个以上的用例中提取公共行为，用 <<include>> 表示
- 扩展关系：如果一个用例明显地混合或两种或两种以上的不同场景，即根据情况可能发生多种分支，则可以将这个用例分为一个基本用例和若干个扩展用例，用 <<extend>> 表示
- 泛化关系：当多个用例共同拥有一种类似的结构和行为的时候，可以将它们的共性抽象成父用例

顺序图

内容

描述对象之间的交互（消息的发送与接收）。

分为三种消息：

- 同步消息（实心三角箭头）
- 异步消息（异步消息）
- 返回消息（返回消息）

活动图

内容

描述过程行为和并行行为。

是一种特殊的状态图，展现了在系统内从一个活动到另外一个活动的流程。

状态图

内容

描述了某一时刻一组对象和它们之间的关系，是类图的一个快照。

对象图

内容

描述了某一时刻一组对象和它们之间的关系，是类图的一个快照。

通信图

内容

分为：

- 交互图
- 通信图

用来显示不同对象的关系

交互概览图

内容

活动图的变体，描述业务过程中的控制流概览。软件过程中的详细度编概观图以及多个图进行拼接，抽象掉信息和注由线，属于顺序图与活动图的混合。

与活动图不同的是，它的节点是交互图。

时序图

内容

用例显示随时间变化，一个或多个元素的值或状态的变更。

构件图

内容

构件图又称为组件图。

组合结构图

内容

描述一个组合结构的内部结构，以及它们之间的关系，用来表示系统中逻辑上的“组合结构”。

包图

内容

描述了系统在包层面上的结构设计，用来表示包和包之间的依赖关系。

UML中的5种视图

内容

- 用例视图：描述系统的功能需求，主要包含用例图。
- 逻辑视图：描述如何实现系统内部的功能，也称为设计视图，主要包括类图、对象图、状态图、顺序图、合作图和活动图。
- 进程视图：描述系统的并发性，主要包括状态图、顺序图、合作图、活动图、构件图和配置图。
- 实现视图：描述系统代码构件组织和实现模块及它们之间的依赖关系，主要包括构件图。
- 部署视图：定义系统种软硬件的物理体系结构及连接、哪个程序或对象主流在哪台计算机上执行，主要包括配置图。

创建型设计模式

内容

创建型设计模式	定义	应用场景
单例（Singleton）模式	确保一个类只有一个实例，并为该实例提供一个全局访问点。	一个类只有一个实例
生成器（Builder）模式	将一个复杂对象的构造与它的表示分离，使得同样的构造过程可以创建不同的表示。	构造过程
工厂（Factory）方法（Factory Method）模式	定义一个用于创建对象的接口，让子类决定实例哪一个类。工厂方法使一个类的实例化延迟到其子类。	子类决定实例化
抽象工厂（Abstract Factory）模式	提供一个创建一系列相关或相互依赖对象的接口，而无须指定它们具体的类。	使用频繁
原型（Prototype）模式	用原型实例指定待创建对象的类型，并且通过复制这个原型来创建新的对象。	对象的复制

结构性设计模式

内容

用于处理类或对象的组合。

结构性模式主要是类的继承和实现。

结构型设计模式	定义	应用场景
适配器模式（Adapter Pattern）	将一个类的接口，转换成客户希望的另外一个接口。适配器模式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。	兼容原本不能一起工作的类
桥接模式（Bridge Pattern）	将抽象部分与它的实现部分分离，使它们都可以独立的变化。	抽象与实现分离
组合模式（Composite Pattern）	有时候会遇到这样的情况：“部分”和“整体”，或对象和一组对象，在程序中表现为一个对象。组合模式将对象组合成树状结构，以“部分-整体”的层次结构，使得客户端对单个对象和组合对象保持一致的方式处理。	部分-整体
装饰器模式（Decorator Pattern）	动态地给一个对象添加一些额外的职责。就增加功能来说，装饰器模式相比生成子类更为灵活。	增加职责
外观模式（Facade Pattern）	为一个复杂的子系统提供一个一致的接口，这个子系统的接口可以被其他的子系统或客户端调用。	简化调用
享元模式（Flyweight Pattern）	运用共享技术有效地支持大量细粒度的对象。	共享
代理模式（Proxy Pattern）	为其他对象提供一种代理以控制对这个对象的访问。	代理

行为型设计模式

内容

行为型设计模式	定义	应用场景
责任链模式（Chain of Responsibility）	为了避免请求发送者与多个请求处理者耦合在一起，将所有请求的处理者通过前一对象记住其下一个的引用而连成一条链；当有请求发生时，可将请求沿着这条链传递，直到有对象处理它为止。	责任链模式，多级
命令模式（Command）	将一个请求封装为一个对象，从而使得可以用不同的请求对客户进行参数化；对请求排队或记录请求日志，以及支持可撤销的操作。	参数化
解释器模式（Interpreter）	给定一个语言，定义它的文法的一种表示，并定义一个解释器，这个解释器使用该表示来解释该语言中的句子。	解释器
迭代器模式（Iterator）	提供一种方法顺序访问一个聚合对象中的各个元素，而又不暴露该对象的内部表示。	顺序访问
中介者模式（Mediator）	用一个中介对象来封装一系列的对象交互。中介者使各对象不需要显式地相互作用，从而使其耦合松散，而且可以独立地改变它们之间的交互。	中介者
备忘录模式（Memento）	在不破坏封装性的前提下，捕获一个对象的内部状态，并在该对象之外保存这个状态，以便以后当需要时能将该对象恢复到原先保存的状态。	保存状态
观察者模式（Observer）	定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新。	观察者
状态模式（State）	允许一个对象在其内部状态改变的时候，会改变其行为。这个对象看上去似乎是由其所属的类中做了修改。	状态改变
模板方法模式（Template Method）	定义一个操作中的算法的骨架，而将一些步骤的执行延迟到子类中。模板方法使得子类可以不改变一个算法的结构即可重定义该算法的某些特定步骤。	模板方法
访问者模式（Visitor）	表示一个作用于某个对象结构中的各元素的操作。它使你可以在不改变这些元素的类的前提下，定义作用于这些元素的新操作。	访问者

抽象工厂模式

构建模式

工厂方法模式

原型模式

单例模式

适配器模式

桥接模式

内容

将抽象部分与其实现部分分离，使它们都可以独立的变化

组合模式

装饰器模式

内容

动态地给一个对象添加一些额外的职责

外观模式

享元蝇域

内容

运用共享技术有效地支持大量细粒度的对象

代理模式

策略模式

责任链模式

命令模式

内容

抽象出待执行的动作以参数化某对象

解释器模式

迭代器模式

中介者模式

备忘录模式

观察者模式

状态模式

模板方法模式

访问者模式

信息安全基础

内容

信息安全包含5个要素：

- 机密性：确保信息不暴露给未授权的实体或进程
- 完整性：只有得到别人允许的人才能修改数据，并且能够判别出数据是否已被篡改
- 可用性：得到授权的实体在需要时可访问数据，即攻击者不能占用所有的资源而阻碍授权者的工作
- 可控性：可以控制授权范围内的信息流向及行为方式
- 可审查性：对出现的信息安全问题提供调查的依据和手段

信息安全基础知识

信息安全的概念

内容

信息安全的范围包括4个安全：

- 设备安全
- 数据安全
- 内容安全
- 行为安全

考点

信息安全的5个要素：

- 机密性：确保信息不暴露给未授权的实体或进程
- 完整性：只有得到允许的人才能修改数据，并且能够判别出数据是否已被篡改
- 可用性：得到授权的实体在需要时可以访问数据，即攻击者不能占用所有的资源而阻碍授权者的工作
- 可控性：可以控制授权范围内的信息流向及行为方式
- 可审查性：对出现的信息安全问题提供调查的依据和手段=

信息存储安全

内容

信息存储安全包括信息使用的安全、系统安全监控、计算机病毒防治、数据的加密和防止非法的攻击等。

网络安全

内容

网络安全威胁表现在：非授权的访问、信息泄露或丢失、破坏数据完整性、拒绝服务攻击、利用网络传播病毒。

安全措施的目标：访问控制、认证、完整性、审计、保密。

信息安全系统的组成框架

内容

信息安全系统涉及以下技术：

- 基础安全设备
- 计算机网络安全
- 操作系统安全
- 数据库安全

信息加解密技术

对称加解密技术

内容

DES、3DES、AES、RC-5、IDEA

公钥基础设施PKI

内容

PKI是以非对称加密技术为基础，以数据机密性、完整性、身份认证和行为不可抵赖性为安全目的，来实施和提供安全服务的具有普适性的安全基础设施。

数字证书：是一种由一个可信任的权威机构签署的信息集合。

考点

签证机构CA：

负责签发证书、管理和撤销证书。是注册用户所信赖的权威机构，CA在给用户签发证书的时候要加上自己的数字签名，以保证证书信息的真实性。任何击击可以用CA的公钥来验证该证书的合法性。

非对称加密技术

内容

非对称加密算法的优点：保密性好。缺点时仅适用于少量数据加密，加密速度慢，时间长。常用来加密对称加密算法的密钥。（数字信封的原理）

考点

常见非对称加密算法：RSA、DSA、ECC

数字签名

内容

数字加密的主要过程:

1. 当信息发送者需要发送信息时,首先生成一个对称密钥,用该对称密钥加密要发送的报文;
2. 信息发送者用信息接收者的公钥加密上述对称的密钥;
3. 信息发送者将上述两个步骤的结果集合在一起传给信息接收者,称为数字信封;
4. 信息接收者使用自己的私钥解密被加密的对称密钥,再用此对称密钥解密被发送方加密的密文,最后得到真正的原文。

数字签名的主要过程:

1. 信息发送者使用一个单项散列函数(Hash函数)对信息生成信息摘要;
2. 信息发送者使用自己的私钥签名信息摘要;
3. 信息发送者把信息本身与已签名的信息摘要一起发送出去;
4. 信息接受者使用与发送者相同的单项散列函数(Hash函数)对接收的信息生成新的信息摘要,再使用发送者的公钥对信息摘要进行验证,以确认信息发送者的身份和信息是否被修改过。

考点

数字信封运用了对称加密技术和非对称加密技术，本质是使用对称密钥加密数据，非对称密钥加密对称密钥，解决了对称密钥的传输问题。

信息摘要的特点：无论数据多长，都会产生固定长度的信息摘要；任何不同的输入数据，都会产生不同的信息摘要；单向性，即只能由数据生成信息摘要，不能由信息摘要还原数据。

数字签名，用发送放的私钥签名，用发送方的公钥验证消息的真实性，发送方不可否认。

访问控制技术

内容

访问控制矩阵，是通过矩阵形式表示访问控制规则和授权用户权限的方法。主体作为行，客体作为列。

	file1	file2	file3
User1	rw		rw
User2	r	rwx	x
User3	x	r	

信息安全的抗攻击技术

密钥的选择

内容

密钥生成要考虑3个方面的因素：

- 增大密钥空间
- 选择强钥（复杂密钥）
- 密钥的随机性（使用随机数）

拒绝服务攻击与防御

内容

传统拒绝服务攻击

内外部用户都可以发动这种攻击。内部用户可以通过长时间占用系统的内容、CPU处理时间使其他用户不能及时得到这些资源，而引起拒绝服务工具；

外部用户：消耗资源、破坏或更改配置信息、物理破坏或改变网络部件、利用服务程序中的处理错误使服务失效。

分布式拒绝服务攻击DDos

攻击者首先侵入并控制一些计算机，然后控制这些计算机同时向一个特定的目标发起拒绝服务工具，克服了传统DOS受网络资源的限制和隐蔽性两大缺点。

欺骗与防御

内容

- ARP欺骗：在ARP请求时，攻击者模拟返回分组格式，构造正确的IP地址和MAC地址映射，当在此发送信息时，攻击者就能监听到发送者的信息
- DNS欺骗：篡改域名服务器，修改了域名和IP地址的对应关系
- IP欺骗：修改主机host文件，达到欺骗目的

端口扫描

内容

端口扫描原理

端口扫描时尝试与目标主机某些端口建立连接。

端口扫描原理分类

- 全TCP连接
- 半打开式扫描（SYN扫描）
- FIN扫描
- 第三方扫描

强化TCP/IP堆栈以抵御拒绝服务攻击

考点

同步包风暴（SYN Flooding）

利用TCP协议缺陷发送大量伪造的TCP请求，使得被攻击者资源耗尽。三次握手，进行了两次，不进行第三次握手，连接队列处于等待状态，大量这样的等待，会占满全部对接空间，使得系统挂起，可以通过修改注册表防御SYN Flooding攻击。

ICMP攻击

操作系统ICMP数据包的最大尺寸不超过64KB，“Ping of Death”攻击利用这一点，达到TCP/IP堆栈崩溃、主机死机的效果。可以通过修改注册表方式ICMP攻击。

SNMP攻击

SNMP还能被用于控制这些设备和产品，重定向通信流，改变通信数据包的优先级，甚至断开通信连接。可以通过修改注册表项防御。

信息安全的保障体系与评估方法

内容

计算机系统安全保护能力的五个等级。

1. 用户自主保护级
2. 系统审计保护级
3. 安全标记保护级
4. 结构化保护级
5. 访问验证保护附

系统架构与设计

- 系统架构与设计
  - 软件架构概念
    - 内容
      软件架构是对系统整体结构的高层描述，关注组件、连接件和约束。架构风格包括：分层、管道-过滤器、仓库、C/S、B/S、事件驱动、微服务等。架构描述语言ADL用于形式化描述架构。
    - 考点
      1.架构风格分类：数据流(管道-过滤器)、调用/返回(主程序/子程序)、独立构件(事件驱动、虚拟机(解释器)、仓库(数据库))
      2.4+1视图模型：逻辑视图、开发视图、进程视图、物理视图+场景视图
      3.MVC架构：Model-View-Controller
      4.微服务特征：独立部署、自治、轻量级通信
      5.SOA vs 微服务：ESB vs API网关、粗粒度 vs 细粒度
    - 易错点
      1.管道-过滤器适合数据流处理，不适合交互式应用
      2.仓库风格适合数据密集型应用
      3.微服务的挑战：分布式事务、服务发现、链路追踪
      4.架构决策记录(ADR)的重要性
  - 软件架构设计与生命周期
    - 需求分析阶段
    - 实现阶段
    - 构件组装阶段
      - 内容
        - 如何支持可复用构件的互联
        - 在组装过程中，如何检测并调解体系结构失配的问题
        失配问题主要包括：
        - 由构件引起的失配
        - 由连接子引起的失配
        - 由于系统或分对全局体系结构的假设存在冲突引起的失配等
    - 部署阶段
      - 内容
        SA对软件部署的作用如下：
        - 提供高层的体系结构视图来描述部署阶段的软硬件模型
        - 基于SA模型可以以分析部署方案的质量属性，从而选择合理的部署方案
    - 后开发阶段
      - 内容
        这一阶段的SA研究主要围绕维护、演化和复用等方面来进行。
    - 设计阶段
      - 内容
        逻辑视图：描述系统的功能需求
        进程视图：关注非功能需求，例如系统的性能和可用性，进程视图强调并发性、分布性、系统集成性和容错能力，以及逻辑视图中的主要抽象的进程结构
        开发视图：指重于软件模块的组织和管理
        物理视图：如何把软件映射到硬件上
        统一的场景：重要系统活动的抽象。
      - 考点
        4+1视图：
        - 逻辑视图
        - 进程视图
        - 开发视图
        - 物理视图
        - 统一的场景
  - 软件构件的定义
    - 内容
      软件架构的定义：一个程序和计算系统软件体系结构是指系统的一个或者多个结构。
      结构中包括软件的构件，构件的外部可见属性以及它们的相互关系。
      构件的特性：
      - 独立部署单元，具有原子性，是不可拆分的
      - 作为第三方的组装单元
      - 没有外部的可见状态
      国际上常用的构件标准主要由三大流派：
      - EJB
      - CORBA
      - DCOM、COM+、组件对象模型
      - COBRA组件
- 软件架构风格
  - 数据流风格
    - 内容
      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，即必须前一个执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，前面执行到部分可以开始下一个执行。
      典型实例：传统编译器、网络报文处理。

    - 批处理架构风格
      - 内容
        构件为一系列固定顺序的计算单元，构件之间只通过数据传递交互，每个处理步骤是一个独立的程序，每一步必须在其前一步结束后才能开始，数据必须是完整的，以整体的方式传递，构件是独立的应用程序，连接件是某种类型的缓存。
    - 管道过滤器风格
      - 内容
        过滤器从管道（数据传输）中读取输入的数据流，经过内部处理，产生输出数据流并写入管道中。前一个过滤器的输出作为后一个过滤器的输入。前后数据流关联。构件就是过滤器，连接件就是管道。
  - 调用/返回风格
    - 主程序/子程序风格
      - 内容
        把问题划分为若干个处理步骤，构件即为主程序和子程序，子程序通常可作成为模块，过程调用作为交互机制，充当连接件的角色。
    - 面向对象体系结构风格
      - 内容
        构件是对象，连接件是对象间交互的方式。对象是通过函数和过程的调用来交互的。
    - 层次性体系结构风格
      - 内容

    - 客户端/服务器体系结构风格
    - 浏览器/服务器体系结构风格
  - 以数据为中心的风格
    - 仓库体系结构风格
      - 内容
        由中央数据结构（说明当前数据状态）和一组独立构件（对中央数据进行操作）组成。连接件是仓库与独立构件之间的交互。
    - 黑板体系结构风格
      - 内容
        包括知识源、黑板和控制三部分。
  - 虚拟机风格
    - 解释器体系结构风格
      - 内容
        专家系统

    - 规则系统体系结构风格
      - 内容
        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，一般用在人工智能领域和决策支持系统DSS中。
  - 独立构件风格
    - 进程通信体系结构风格
    - 事件系统体系结构风格
  - C2风格
    - 内容
      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。

- 基于构件软件的开发
  - 概述
    - 内容
      ABSD的三个基础：
      - 功能的分解
      - 选择架构风格来实现质量和商业需求
      - 基于软件模板的使用
  - 概念和术语
    - 内容
      ABSD是一个自顶向下，通过细化的方法，软件系统的体系结构通过迭代法得到细化，直到能产生软件构件和类
    - 考点
      ABSD的产物：软件构件和类
  - 基于架构的开发模型
    - 内容
      ABSD把软件过程分为六个部分：
      - 架构需求
      - 架构设计
      - 架构文档化
      - 架构复审
      - 架构实现
      - 架构演化
      - 架构需求
        - 内容
          架构需求一般来自三个方面：
          - 系统的质量目标
          - 系统的商业目标
          - 系统开发人员的商业目标
      - 架构设计
        - 内容
          将需求阶段的标识构件映射成构件，进行分析
      - 架构文档化
        - 内容
          主要产出用例文档，即架构规则说明和测试架构需求的质量设计说明书
      - 架构复审
        - 内容
          由外部人员（独立于开发组织之外的人，如用户代表和领域专家）参加复审
        - 考点
          架构复审的目的是标识潜在的风险，及早发现架构设计中的缺陷和错误
      - 架构实现
        - 内容
          用实体来显示出架构。实现构件、构件组装成系统
      - 架构演化
        - 内容
          对架构进行改变，按需求增删构件，使架构可复用
- 特定领域软件架构DSSA
  - DSSA定义
    - 内容
      DSSA：用于某一类特定领域的标准软件构件的集合。
      - 垂直域：一个特定领域中的通用软件架构
      - 水平域：在多个系统和多个系统族中功能区域的共有部分，例如网络和数据都有收集系统、收集系统的基水平域
  - DSSA的建立过程
    - 内容
      - 定义领域范围
      - 定义领域特定的元素
      - 定义领域特定的设计和实现需求的约束
      - 定于领域特定架构
      - 产生、搜集可复用的产品单元
  - 领域分析
    - 内容
      领域分析的主要目标：获得领域模型（领域需求）
  - 领域设计
    - 内容
      这个阶段的主要目标是获得DSSA，DSSA描述在领域模型中标识的需求的解决方案
  - 领域实现
    - 内容
      依据领域模型和DSSA开发和组织可重用信息
  - 参与DSSA的人员
    - 内容
      - 领域专家：该领域中有经验的用户，从事该领域中系统的需求分析、设计、实现以及项目管理的有经验的软件工程师等
      - 领域分析人员：具有知识工程背景的有经验的系统分析员来担任
      - 领域设计人员：有经验的软件设计人员来担任
      - 领域实现人员：有经验的程序设计人员来担任
- 系统架构评估
  - 重要概念
  - 系统架构评估方法
    - 基于场景的架构分析方法SAAM
    - 架构权衡分析方法ATAM
      - 质量属性效用树
    - 成本效益分析法CBAM
    - SEAM方法（了解）
    - SAABNet方法（了解）
    - SACMM方法（了解）
    - SASAM方法（了解）
    - ALRRA方法（了解）
    - AHP方法
    - COSMIC + UML方法（了解）

系统质量属性与架构评估

内容

软件质量属性是系统的非功能性需求，包括：性能(响应时间、吞吐量)、可用性(故障恢复)、安全性(认证、授权、加密)、可修改性(变更成本)、可测试性、易用性等。每个质量属性都有对应的战术来实现。

考点

1.性能战术：资源需求(减少计算量)、资源管理(引入并发)、资源仲裁(调度策略)
2.可用性战术：错误检测(心跳)、错误恢复(冗余、回退)、错误预防(从服务中移除)
3.安全战术：抵抗攻击(认证、授权)、检测攻击(审计)、恢复(恢复状态)
4.可修改性战术：局部化修改(高内聚)、防止连锁反应(信息隐藏)、延迟绑定
5.质量属性场景六要素：刺激源、刺激、环境、制品、响应、响应度量

易错点

1.性能=响应时间+吞吐量+资源利用率
2.可用性用MTTF和MTTR衡量
3.安全三要素：CIA(机密性、完整性、可用性)
4.可修改性关注变更的成本
5.战术是实现质量属性的设计决策

概念

内容

软件系统质量属性是一个系统的可测量或者可测试的属性。

开发期质量属性：

- 易理解性：指设计被开发人员理解的难易程度
- 可扩展性：软件因使用新需求或者需求变化而增加新功能的能力
- 可重用性
- 可测试性
- 可维护性
- 可移植性

运行期质量属性：

- 性能
- 安全性
- 可伸缩性
- 互操作性
- 可靠性
- 可用性
- 鲁棒性

面向架构评估的质量属性

内容

- 性能
- 可靠性
- 可用性
- 安全性
- 可修改性
- 功能性
- 可变性
- 互操作性

质量属性场景描述

内容

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作为描述质量属性的手段。

- 刺激源：某个生成该刺激的实体（人、计算机系统或者任何其他刺激器）
- 刺激：当刺激到达系统时需要考虑的条件
- 环境：该刺激在某些条件内发生
- 制品：某个制品被激励，这可能是整个系统，也可能是系统的一部分
- 响应：在激励到达后所采取的行动
- 响应度量：当响应发生时，应当能够以某种方式对其进行度量，以对需求进行测试

以淘宝APP为例：
刺激源：淘宝APP进行更新，开发人员修改代码
刺激：淘宝APP暂时维护，增加部分功能
环境：APP平台
制品：新版本的淘宝APP
响应：增添部分功能或者修复部分BUG
响应度量：过去的BUG消失或者增加了新功能

软件可靠性基础知识

内容

系统可靠性指系统在规定条件下和规定时间内完成规定功能的能力。包括硬件可靠性和软件可靠性。可靠性设计技术包括：冗余技术、容错技术、故障检测与恢复。

软件可靠性基本概念

考点

平均失效时间（MTTF）：系统从开始运行到第一次故障所经历的平均时间。
平均恢复时间（MTTR）：平均修复时间，从出现故障到修复成功的时间。
平均故障间隔时间（MTBF）：MTTF + MTTR。

可靠性模型：串联系统R=R1xR2...Rn，并联系统R=1-(1-R1)(1-R2)...(1-Rn)

系统的可用性 = MTTF/(MTTF + MTTR) * 100%。

软件可靠性管理

考点

工程各个阶段可能进行的主要软件可靠性活动如下：

- 需求分析阶段：确定可靠性目标、分析影响因素、确定验收标准、执行管理框架、制订文档编写规范、制订活动初步计划、确定数据收集规范。
- 概要设计：确定可靠性度量、制订详细验收方案、可靠性设计。
- 详细设计阶段：可靠性设计、可靠性预测、调整活动计划。
- 编码阶段：可靠性测试。
- 测试阶段：可靠性建模、可靠性评价。
- 实施阶段：调整模型，可靠性评价。

软件可靠性设计

容错技术设计

内容

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

前向恢复：使当前的计算继续下去，把系统恢复成连贯的正确状态，弥补当前状态的不连贯情况。
后向恢复：系统恢复到前一个正确状态，继续执行。

恢复块设计（动态冗余）

内容

选择一组操作作为容错设计单元，从而把普通的程序块变成恢复块。

N版本程序设计

内容

N个程序由不同人独立设计，使用不同的算法、编程语言、编译程序、设计工具、实现方法和测试方法，
N版本程序设计是一种静态的故障屏蔽技术，采用前向恢复的策略。

冗余设计

内容

设计一种不同路径、不同算法或不同实现方法的模块或系统作为备份，在出现故障时可以使用冗余的部分进行替换，从而维持软件系统的正常运行。

检错技术

内容

采用检错设计技术考虑要素：检测对象、检测延迟、实现方式和处理方式。

系统配置技术

内容

双机热备技术

双机热备模式：Active/Standby方式。
双机互备模式：彼此均为备机。
双机双工模式：两种服务器均处于活动状态，同时运行相同的应用。

服务器集群技术

一组相互独立的服务在网络中组合成为单一的系统工作，并以单一系统的模式加以管理。

负载均衡算法

常用的负载均衡实现技术：

- 基于特定软件的负载均衡（应用层）。很多网络协议都支持重定向功能，例如，基于HTTP重定向服务，其主要原理是服务器使用HTTP重定向指令，将一个客户端重新定位到另一个位置。服务器返回一个重定向响应，而不是返回请求的对象。客户端确认新地址然后重发请求，从而达到负载均衡的目的。
- 基于DNS的负载均衡（传输层），主要原理是在DNS服务器中为同一个主机名配置多个地址，在应答DNS查询时，DNS服务器对每个查询将以DNS文件中主机记录的IP地址按顺序返回不同的解析结果，将客户端的访问引导到不同的节点上去,使得不同的客户端访问不同的节点，从而达到负载均衡的目的。
- 基于NAT的负载均衡。将一个外部IP地址映射为多个内部IP地址，对每次连接需求，动态地转换为一个内部节点的地址，将外部连接请求引导给得到地址的那个节点上，从而达到负载均衡的目的。
- 反向代理负载均衡。将来自Internet上的连接请求以反向代理的方式动态地转发给内部网络上的多个节点进行处理，从而达到负载均衡的目的。
- 混合型负载均衡。

软件可靠性测试

内容

软件可靠性测试由可靠性目标的确定、运行剖面的开发、测试用例的设计、测试实施、测试结果的分析等主要活动组成。

软件可靠性评价

内容

软件可靠性评价的3个过程：选择可靠性模型，收集可靠性数据、可靠性评估和预测。

未来信息综合技术

区块链技术

内容

区块链是一种去中心化的分布式账本，通过密码学将数据块按时间顺序链接，借助共识机制在无信任节点间达成一致，具有不可篡改、可追溯的特性，智能合约支持自动执行的可编程业务逻辑。

考点

1.特征：去中心化、不可篡改、可追溯、集体维护
2.结构：区块头(前块哈希、时间戳、Merkle根)+区块体
3.共识机制：PoW 工作量证明、PoS 权益证明、PBFT
4.Merkle 树用于高效校验交易完整性
5.分类：公有链、联盟链、私有链

易错点

1.每个区块含前一区块哈希，篡改会断链
2.PoW 耗能高(挖矿)，PoS 按持币权重
3.智能合约部署后自动执行且难以随意更改

人工智能与机器学习

内容

人工智能是模拟人类智能的计算机系统。机器学习是AI的子领域，包括监督学习、无监督学习、强化学习。深度学习是机器学习的子领域，使用多层神经网络。

考点

1.监督学习：分类(决策树、SVM、神经网络)、回归
2.无监督学习：聚类(K-means)、降维(PCA)
3.强化学习：Q-learning、DQN
4.神经网络：感知机、CNN(图像)、RNN/LSTM(序列)、Transformer
5.评估指标：准确率、精确率、召回率、F1值
6.过拟合与欠拟合：正则化、交叉验证、Dropout

易错点

1.训练集/验证集/测试集的划分
2.过拟合：模型在训练集表现好但泛化能力差
3.偏差-方差权衡
4.交叉验证避免过拟合
5.集成学习(随机森林、XGBoost)提高泛化能力

大数据技术

内容

大数据具有海量、多样、高速、低价值密度(4V)等特征，传统数据库难以处理。Hadoop 通过 HDFS 分布式存储和 MapReduce 并行计算处理大规模数据，Spark 基于内存计算显著提升性能。

考点

1.大数据 4V：Volume、Variety、Velocity、Value
2.HDFS：主从架构(NameNode/DataNode)、数据分块多副本
3.MapReduce：Map 映射+Reduce 归约，移动计算而非数据
4.Spark：基于内存的 RDD，比 MapReduce 快，支持流批处理
5.生态：Hive、HBase、Kafka、YARN

易错点

1.Spark 比 MapReduce 快因中间结果在内存而非磁盘
2.HDFS 适合大文件顺序读写，不适合小文件和随机写
3.NameNode 存元数据，DataNode 存实际数据块

边缘计算与物联网架构

内容

物联网将物理设备通过网络互联实现感知与控制，典型采用感知层、网络层、应用层的分层架构。边缘计算在靠近数据源的边缘节点就近处理数据，降低时延和带宽压力，与云计算形成云边端协同。

考点

1.IoT 三层架构：感知层、网络层、应用层(可加平台层为四层)
2.感知层：传感器、RFID、二维码
3.边缘计算：就近处理，低时延、省带宽、保隐私
4.云边端协同：云做全局、边做实时、端做采集
5.应用：智能制造、智慧城市、车联网

易错点

1.边缘计算不取代云计算而是协同
2.边缘计算优势在低时延和减少回传
3.感知层负责采集，网络层负责传输

云计算与虚拟化

内容

云计算提供按需的计算资源服务。部署模型：公有云、私有云、混合云。服务模型：IaaS(基础设施即服务)、PaaS(平台即服务)、SaaS(软件即服务)。虚拟化技术包括：硬件虚拟化、操作系统虚拟化、应用虚拟化。

考点

1.IaaS：提供虚拟机、存储、网络(AWS EC2)
2.PaaS：提供开发运行环境(Heroku、GAE)
3.SaaS：提供应用软件(Gmail、Office 365)
4.容器技术：Docker(轻量级虚拟化)、Kubernetes(容器编排)
5.虚拟化类型：全虚拟化、半虚拟化、硬件辅助虚拟化
6.云原生：容器化+微服务+DevOps+持续交付

易错点

1.IaaS/PaaS/SaaS的区别
2.Docker vs 虚拟机：Docker共享宿主机内核
3.Kubernetes核心概念：Pod、Service、Deployment
4.弹性伸缩：根据负载自动增减实例
5.云安全：数据加密、访问控制、合规审计

数字孪生体技术

内容

数字孪生通过对物理实体建立实时同步的数字模型，实现仿真、监控、预测和优化。信息物理系统(CPS)将计算、通信与物理过程深度融合，是工业互联网和智能制造的核心支撑技术。

考点

1.数字孪生：物理实体的高保真数字镜像、实时映射
2.核心要素：物理空间、虚拟空间、数据连接
3.CPS：计算、通信、控制(3C)与物理过程融合
4.应用：智能制造、设备预测性维护、智慧城市
5.支撑技术：物联网、大数据、仿真建模、AI

易错点

1.数字孪生强调实时双向映射，区别于静态仿真
2.CPS 强调物理与信息深度融合
3.数字孪生依赖物联网提供实时数据

知识产权基础知识

标准化基础知识

标准分类

内容

分类	内容
国际标准（IS）	国际标准化组织（ISO）、国际电工委员会（IEC）制定的标准， 以及ISO出版的《国际标准题内关键词索引》所收录的其他国际组织制定的标准
	中国（GB）：中华人民共和国国家技术监督局

标准的代号和编号

内容

标准	编号规则
国际标准	ISO+标准号+[杠+分标准号]+冒号+发布年号，方括号中的内容可有可无

知识产权基础知识

计算机软件著作权

内容

计算机软件著作权的客体：计算机程序及其有关文档。

软件著作权自软件开发完成之日起产生。

知识产权的保护期限

内容

客体类型	权利类型	保护期限
公民作品	署名权、修改权、保护作品完整权	无限制
	发表权、使用权和获得报酬权	作者终生及其死后50年（第50年的12月31日）
单位作品	发表权、使用权和获得报酬权（属于集体，无署名权、修改权等）	50年（首次发表后的第50年的12月31日），若期间未发表，不保护
公民软件产品	署名权、修改权	无限制
	发表权、复制权、发行权、出租权、信息网络传播权、翻译权、使用许可权、获得报酬权、转让权	作者终生及其死后50年（第50年的12月31日），合作开发的，以最后死亡作者为准
[illegible]	发表权、复制权、发行权、出租权、信息网络传播	50年（首次发表后的第50年的12月31日），若期间未发

单位和个人知识产权的归属判定

内容

情况说明		判断说明	归属
[illegible]	[illegible]	利用单位的物质技术条件进行创作，并由单位承担责任的	除署名权外，其他著作权归单位
[illegible]	[illegible]	[illegible]	[illegible]

委托知识产权的归属判定

内容

分布式系统

分布式系统

内容

分布式系统是由多个通过网络互联的计算机组成的系统。核心问题：一致性(CAP定理)、分区容错、可用性。关键技术：分布式锁、分布式事务、消息队列、负载均衡、服务发现。

考点

1.CAP定理：一致性(C)、可用性(A)、分区容错(P)三者最多满足两个
2.BASE理论：基本可用、软状态、最终一致性
3.分布式事务：2PC(两阶段提交)、3PC、TCC、Saga
4.一致性算法：Paxos、Raft
5.负载均衡算法：轮询、加权轮询、最少连接、一致性哈希
6.RPC框架：gRPC、Dubbo、Thrift

易错点

1.CAP中P是必须的，所以只能在C和A之间选择
2.BASE是CAP的妥协方案
3.2PC存在阻塞和单点故障问题
4.Raft比Paxos更容易理解和实现
5.微服务间通信推荐异步消息队列

面向服务架构SOA

内容

面向服务架构(SOA)将系统功能封装为标准化、松耦合、可复用的服务，通过定义良好的契约交互。服务可被编排为业务流程，支持跨系统集成和对业务变化的敏捷响应。

考点

1.核心概念：服务、契约、松耦合、无状态、可复用
2.服务粒度：粗粒度，面向业务
3.实现基础设施：ESB 企业服务总线
4.服务编排 orchestration(集中)与协同 choreography(分布)
5.相关标准：SOAP、WSDL、UDDI、BPEL

易错点

1.SOA 强调松耦合和标准契约
2.BPEL 用于服务编排(流程定义)
3.SOA 粗粒度，微服务细粒度

云原生架构

内容

- 服务化原则
- 弹性原则
- 可观测原则
- 韧性原则
- 所有过程自动化原则
- 零信任原则
- 架构持续演进原则

微服务架构与服务治理

内容

微服务将单体应用拆分为一组小的、可独立部署的服务，围绕业务能力构建，通过轻量级通信协作。服务数量增多带来服务发现、容错、配置和链路追踪等治理挑战，常借助服务网格解耦治理逻辑。

考点

1.特征：独立部署、去中心化、按业务拆分、技术异构
2.基础设施：注册发现、API 网关、配置中心、链路追踪
3.容错：熔断、限流、降级、隔离(舱壁)
4.服务网格 Service Mesh：Sidecar 边车代理(Istio/Envoy)
5.数据去中心化：每个服务独立数据库

易错点

1.微服务挑战：分布式事务、服务依赖、运维复杂
2.Service Mesh 将治理逻辑下沉到基础设施(Sidecar)
3.微服务相较 SOA：细粒度、去 ESB、独立数据

Web Service与RESTful

内容

Web 服务是基于标准协议、可跨平台调用的分布式构件。传统 Web 服务采用 SOAP/WSDL/UDDI 技术栈，基于 XML 交互。REST 是一种架构风格，将一切视为资源，用 HTTP 方法无状态操作，轻量简洁。

考点

1.SOAP：基于 XML 的消息协议；WSDL 描述服务；UDDI 服务注册发现
2.REST 约束：客户端服务器、无状态、可缓存、统一接口、分层
3.REST 用 HTTP 方法 GET/POST/PUT/DELETE 操作资源
4.SOAP 重量级(标准完备)，REST 轻量级(简单高效)
5.幂等性：GET/PUT/DELETE 幂等，POST 非幂等

易错点

1.REST 是架构风格不是协议，SOAP 是协议
2.无状态指服务端不保存客户端会话状态
3.WSDL 描述接口，UDDI 是服务目录

Web架构

内容

现代Web架构采用分层设计：前端(HTML/CSS/JS)、Web服务器(Nginx)、应用服务器、数据库服务器。常用技术栈：前端框架(Vue/React)、后端框架(Gin/Spring)、数据库(MySQL/Redis)、中间件(MQ)。

考点

1.前后端分离架构：前端SPA+后端API
2.Nginx负载均衡：轮询、IP哈希、最少连接
3.缓存策略：本地缓存(进程内)、分布式缓存(Redis)
4.数据库优化：索引优化、SQL优化、读写分离、分库分表
5.CDN加速静态资源分发
6.API网关：路由、限流、鉴权、熔断

易错点

1.Nginx反向代理和正向代理的区别
2.Redis常用数据结构：String、Hash、List、Set、ZSet
3.数据库索引B+树比B树更适合范围查询
4.分库分表策略：水平分片、垂直分片
5.消息队列解耦异步处理

企业应用集成EAI与ESB

内容

企业应用集成(EAI)将企业内异构的应用、数据和流程集成为统一整体，实现互联互通。企业服务总线(ESB)作为 SOA 的基础设施，提供消息路由、协议转换和数据格式转换，实现服务的松耦合集成。

考点

1.集成层次：数据集成、控制/API 集成、界面集成、流程集成
2.ESB 功能：消息路由、协议转换、数据转换、服务注册
3.集成方式：点对点(紧耦合)、总线(松耦合)
4.适配器屏蔽异构系统差异
5.ESB 是 SOA 落地的核心基础设施

易错点

1.点对点集成连接数随系统数平方增长，难维护
2.ESB 实现松耦合，降低系统间直接依赖
3.界面集成是表示层集成，数据集成在数据层

DevOps与持续交付

内容

DevOps 打通开发与运维，通过自动化和协作缩短交付周期，提升发布质量。持续集成(CI)频繁合并并自动构建测试，持续交付/部署(CD)自动将变更发布到目标环境，形成端到端流水线。

考点

1.CI 持续集成：频繁提交、自动构建、自动测试
2.CD：持续交付(可随时发布)与持续部署(自动发布生产)
3.流水线：代码-构建-测试-部署-监控
4.基础设施即代码 IaC、自动化配置管理
5.文化：协作、自动化、度量、共享

易错点

1.持续交付需人工确认发布，持续部署全自动
2.DevOps 是文化和实践，非单一工具
3.CI/CD 依赖自动化测试保证质量

容器与编排技术

内容

容器通过操作系统级虚拟化打包应用及其依赖，共享宿主内核，比虚拟机更轻量、启动更快。Docker 是主流容器引擎，Kubernetes 提供容器编排，实现自动部署、扩缩容、服务发现和故障自愈。

考点

1.容器 vs 虚拟机：容器共享内核轻量，虚拟机含独立 OS 重量
2.Docker：镜像(分层只读)、容器(运行实例)、仓库 Registry
3.K8s 核心对象：Pod、Service、Deployment、Node
4.K8s 能力：自动扩缩容、滚动更新、自愈、负载均衡
5.镜像分层与联合文件系统

易错点

1.容器共享宿主机内核，隔离性弱于虚拟机
2.Pod 是 K8s 最小调度单元，可含多个容器
3.镜像不可变，容器是镜像的运行实例

事件驱动架构与CQRS

内容

事件驱动架构(EDA)以事件的产生、检测和消费为核心，组件通过异步事件解耦。消息队列实现发布订阅。事件溯源以事件序列记录状态变化。CQRS 将读写模型分离以优化性能和扩展性。

考点

1.EDA 组件：事件生产者、事件通道、事件消费者
2.发布订阅解耦生产者与消费者
3.消息中间件：Kafka、RabbitMQ、RocketMQ
4.事件溯源 Event Sourcing：存储事件而非当前状态
5.CQRS：命令(写)与查询(读)模型分离

易错点

1.消息队列作用：解耦、异步、削峰
2.CQRS 读写分离增加复杂度，适合读写差异大场景
3.事件溯源可回放事件重建状态

系统运维与交付

内容

云计算与虚拟化、容器编排、DevOps 与持续交付等运维交付技术。

系统容灾与备份

内容

容灾备份保障系统在故障或灾难后能恢复运行和数据。恢复时间目标(RTO)衡量恢复时长，恢复点目标(RPO)衡量可容忍的数据丢失量。通过异地备份、双活数据中心和两地三中心等方案提升容灾等级。

考点

1.RTO：恢复时间目标(多久恢复)；RPO：恢复点目标(丢多少数据)
2.备份方式：完全备份、增量备份、差异备份
3.备份中心：冷备、温备、热备
4.容灾方案：本地双活、异地灾备、两地三中心
5.灾备等级越高，RTO/RPO 越小，成本越高

易错点

1.RPO 关注数据丢失量，RTO 关注恢复时长
2.增量备份基于上次备份，差异备份基于上次完全备份
3.双活比主备利用率高但一致性要求更高