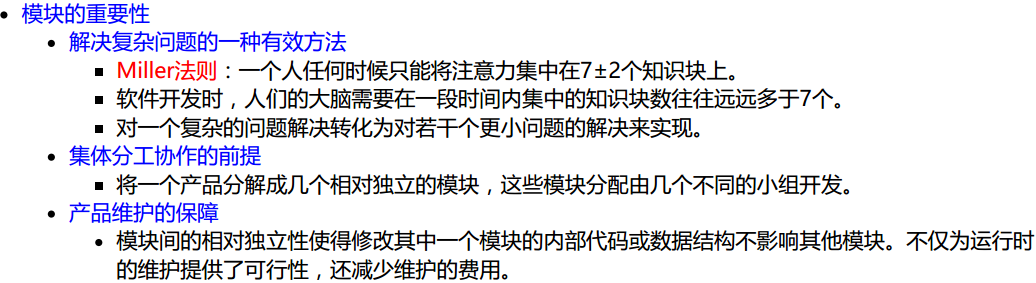
1. 什么是软件体系结构？软件产品在实现时通常分解成一些较小的模块，这种做法的意义是什么？

一个软件系统的体系结构是指构成这个系统的计算部件（computational components）、部件间的相互作用关系（interactions）。部件间的相互作用关系称为连接（connector）

软件产品在实现时通常分解成一些较小的模块，这种做法的意义是：解决复杂问题的一种有效方法；集体分工协作的前提；产品维护的保障。（详细内容在第一讲里）



1. 在C/S结构中，有状态信息服务器和无状态信息服务器相比，哪一种服务器对同时使用的用户有限制？为什么？

有状态信息服务器对同时使用的用户数有限制。因为有状态信息服务器需要设计一个状态信息表，用于登记当前的用户操作请求，当状态信息表填满时，新用户的操作被推迟，意味着同时使用的用户数受限制。（详见第五讲层次结构分析中的二部分）

1. 总结P/F结构风格的缺点（P/F：Pipes and Filters 管道/过滤器，第二讲的第二部分）

通常导致进程成为批处理的结构。这是因为虽然过滤器可增量式地处理数据，但它们是独立的，所以设计者必须将每个过滤器看成一个完整的从输入到输出的转换。

不适合于需要共享大量数据的应用设计。

不适合处理交互的应用。当需要增量地显示改变时，这个问题尤为严重。

过滤器之间通过特定格式的数据进行工作，数据格式的设计和转换是系统设计的主要方面，为了确保过滤器的正确性，必须对数据的句法和语义进行分析，这增加了过滤器设计的复杂性。

并行运行获得高效率往往行不通。原因，第一，独立运行的过滤器之间的数据传送的效率会很低，在网络传送时尤其如此；第二，过滤器通常是在消耗了所有输入后才产生输出；第三，在单处理器的机器上进程的切换代价是很高的；第四，通过管道对过滤器的同步控制可导致频繁启动和停止过滤器工作。

1. 在分布是令牌环互斥算法中，一个进程进入临界区执行需要令牌消息传递的个数是[1,∞]，解释这里∞含义。（第四讲第四部分分布式互斥算法）

分布式在令牌环互斥算法中，算法保持一个令牌在环中传递，一个进程只有在得到令牌时才能进入临界区执行，以便实现互斥，技术当前所有进程都没有申请进入临界区运行，令牌也要沿着环不断地传递；因此当一个进程要进入临界区执行时，令牌消息传递的个数是不确定的，可能有任意多次，即∞。

1. 解释RPC透明性的含义，RPC中客户代理（client stub）实现什么主要功能？

含义：RPC透明性的思想使得远程过程调用尽可能象本地调用一样，即调用进程应该不知道被调用过程是在另一台计算机上执行，反过来也是如此，被调用过程也应该不知道是由哪台机器上的进程调用。客户代理(client stub)主要功能：参数整理、查找服务器地址；向服务器发送请求；阻塞等待接收结果，分析结果返回并唤醒client进程。

（RPC文档里的内容）将参数封装成消息，请求内核将消息发送给服务器；调用receive原语进入阻塞状态；当消息返回到客户时，内核找到客户进程，消息被复制到缓冲区，并且客户进程解除阻塞；客户代理检查消息，从中取出结果，并将结果复制给它的调用进程

1. RPC中服务器代理（server stub）实现什么主要功能？

调用receive原语，处于阻塞状态，并等待消息的到来当消息到达后，代理被唤醒；将消息分解，并取出参数；以通常方式调用服务器的过程；调用结束并获得控制后，把结果封装成消息；调用send原语发送给客户重新调用receive等待下一个消息

1. 简述QDS的思想、作用及目标（第七讲： 三量化设计空间）。

QDS思想是基于设计空间（design space）和质量配置函数（quality function deployment，QFD）。QDS的作用是一种将系统需求转换为可选择的功能和结构设计，量化（quantified）分析各种的可能选择记忆组合。QDS的目标为一个期望的系统（desired system）制作（produce）一个设计模型，也可以用来分析和比对已有的设计，或对仙游的设计提出改进建议，作为自动系统设计（automated system design）的基础。

1. RPC动态联编（Binding）有哪些特点？定期转询的功能是什么？

动态联编的灵活性(flexible)；均衡工作量(load balancing):支持多服务器(support multiple servers)，把客户均衡地分布于各个服务器上；容错性(fault tolerance):定期转询服务器(poll the server perodically)，对无响应的过程unregister，到达一定程度的容错性；支持权限:服务器可以指定由哪些用户使用，这样联编对非授权的用户拒绝接受。动态联编的缺点(disadvantages)：花费系统时间:the extra overhead of exporting and importing interfaces costs time.客户进程往往执行时间短，但每次每个进程要重新import to binder；瓶颈(bottleneck):use multiple binders

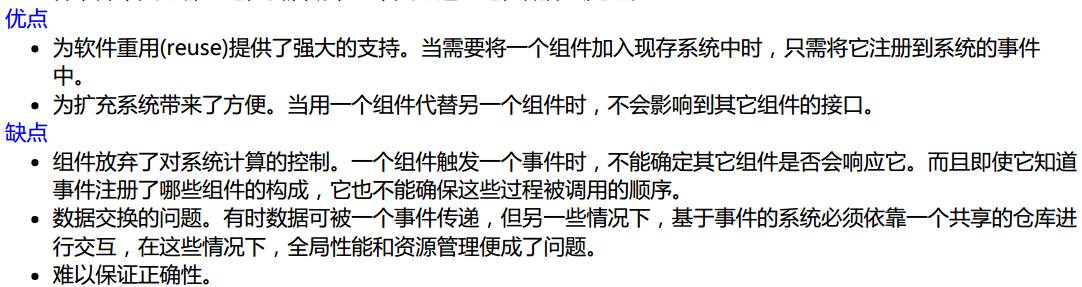
定期转询功能：定期转询服务器(poll the server perodically)，对无响应的过程unregister，到达一定程度的容错性，（客户代理定期轮询，对无响应的过程unregister。）

1. 在C/S结构中，有状态信息服务器和无状态信息服务器相比，哪一种服务器更具有可靠性？为什么？

无状态信息服务器更具有可靠性：因为无状态信息

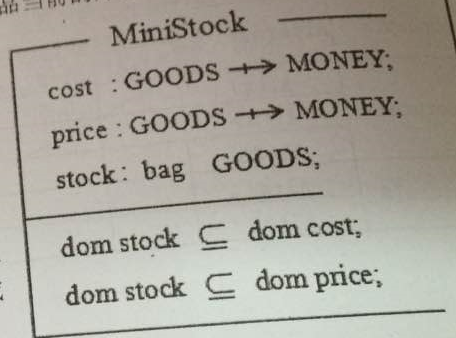
1. 总结基于事件隐含激活的结构风格特点（第二讲）。

在一个基于事件的隐含激活系统中，组件不直接调用一个过程，而是触发(announce)或广播(broadcast)一个或多个事件（event)。系统中的组件事先注册（register)它们感兴趣的事件及对应的过程，将来，当一个事件被触发（inovaation），系统自动调用在这个事件中注册的所有过程，这样，一个事件的触发就导致了另一模块中的过程的调用。从体系结构上说，这种隐含激活风格的组件是一些模块，这些模块是一些过程和一些事件的集合。过程可以用通常的方式调用，也可以在系统事件中注册一些过程，当发生这些事件时，过程被隐含调用。



1. 某简单账户管理系统主要维护用户姓名和当前账目月，提供账户新增加，查询，收入，和支出等操作，要求账目的月不小于0，请应用于Z Notation形式化如下的系统设计。
2. 定义一个状态模式（UserAccount），描述账户的基本属性。
3. 定义一个操作模式（NewUser），用于注册一个新的用户的用户账户和初始余额，规定同一账号不能重复使用
4. 定义一个操作模式（Import），向给定cur-user的账目中添加指定金额cur-money
5. 定义一个操作模式（Emport），向给定cur-user的账目中扣除指定金额cur-money
6. 定义一个操作模式（Query），查询系统中账目余额小于指定金额cur-money的所用用户

简单库存管理系统： 产品当前的库存量（num）、成本（cost）和售价（price）；提供入库、出库、查询等操作。要求：库存中的产品都应定义了成本和售价，库存量非负。已知自由类别[GOODS]为产品集合，[MONEY]为价格集合。系统的状态模式（MiniStock）如右图所示。



请应用Z Notation形式化下列的各操作模式：

1. 定义一个入口操作模式（import）：实现一批产品及数量的入库操作，要求每一入库产品都应定义了成本和售价。
2. 定义一个出库操作模式（Export）：指定一个产品及出库数量，实现出库操作。
3. 定义一个成本设置操作模式（SetCost）：实现一批产品及成本的设置操作。
4. 定义一个调价操作模式（ModiPrice），将系统中的售价提高x%。
5. 定义一个利润预算模式（SumProfit），计算系统中指定一个产品的库存的利润。一个产品的利润=（该产品的售价-该产品的成本）\*该产品的库存量。

第一讲 软件体系结构概述

一、什么是软件体系结构（Software Architecture）

1.软件体系结构是软件工程的一门新兴学科

2.在系统科学中，系统(system)的定义

系统的特性(集合性,相关性, 整体性, 功能性, 环境适应性

系统的体系结构

3.软件体系结构的定义（Mary Shaw, David Garlan)

4.软件设计的层次性(结构级, 代码级, 执行级)

5.体系结构的类别(概略型, 需求型, 设计型)

6.体系结构的重要性(规划, 需求, 设计, 实施, 测评, 维护阶段)

二、模块及其设计

1.什么是模块(定义, 模块的重要性)

2.模块内聚性（module cohesion),含义

内聚性的层次（偶然，逻辑，暂时，过程，通信，信息，功能）

3.模块耦合(module coupling)

耦合的级别（内容，共用，控制，特征，数据耦合）

4.重用

重用的两种类型，偶然，计划重用

关于构件重用

重用的障碍（自负，经济利益，检索，代价，版权）

三、软件工程（Software Engineering)

1.软件危机（software crisis)

计算机硬件技术的发展迅速，软件危机，The Mythical Man-Month

2.软件工程的复杂性

建筑工程的经验对软件工程的启发

软件工程的复杂性

软件在执行时处于离散状态。

软件运行环境具有不可再现性

硬件的复杂性

3.软件工程的内容 软件工程的生命周期

四、软件体系结构的意义与目标

1.软件体系结构的意义

2.软件体系结构的目标

五、软件体系结构的研究现状

1.软件体系统结构的发展

程序抽象，软件工程，体系结构

2.软件体系统结构的研究现状

体系统结构的研究现状其现状具体表现在：

缺乏系统统一的概念和坚实的理论基础

缺乏工程知识的系统化和标准化

缺乏形式化，没有建立统一的体系结构的工程描述方法

目前，软件体系结构已经成为软件工程的从业者一个重要研究领域。

软件体系结构的研究内容（风格(styles)，设计模式(design patterns) ，结构描述语言（ADL）

第二讲　软件体系结构风格

一、什么是软件体系结构风格（Architecture Styles）

1.什么是结构风格（Architecture Styles）

2.体系结构风格的分类

Garlan和Shaw对通用体系结构风格的分类（数据流系统（Dataflow systems)，调用/返回系统(Call-return systems)，独立组件(Independent components)，虚拟机(Virtual machines) 仓库(Repositories)）

二、常用结构风格（Architecture Styles)

1.管道/过滤器（Pipes and Filters,P/F) 系统组织，优缺点

2.数据抽象与面向对象组织（Data abstraction and Object-oriented Organization)系统组织

3.事件及隐含激活（Event-Based, Implicit Inovaations）系统组织

4.层次系统（Layers Systems)

5.仓库(Repositories)

6.解释器（Interpreters)系统组织

三、其他结构风格

1.分布式处理（Distributed Processes)

2.主程序/子过程调用（Main program/subroutine organiztions)

3.确定域结构(Domain-specific)

4.状态变迁系统（State transition system)

四、案例分析（Case Studies)

1.KWIC问题描述

2.方法1：主程序/子过程调用（Main program/subroutine）

3.方法2：数据抽象与面向对象组织（Data abstraction and Object-oriented Organization)

4.风格3：事件及隐含激活（Event-Based, Implicit Inovaations）

5.风格4：管道/过滤器（Pipes and Filters,P/F)

第三讲　分布式体系结构分析

一、分布式处理(Distributed process)结构风格概述

1.分布式处理结构风格的系统组织

2.分布式处理结构风格的优缺点

3.构造分布式处理结构风格的主要技术（层次结构，技术基础，RPC，组件技术，云计算技术）

二、分布式系统（Distributed System)

1.分布式系统是计算机发展的必然产物

集中式系统和分布式系统

一个常见的分布式系统的例子

分布式系统的优点，不足

三、客户/服务器（Client/Server)结构

1.C/S结构

2.C/S模型的软件结构的特点（简化软件产品的设计，提高软件的可靠性，适合分布式计算环境）

3.C/S模型的工作过程

4.数据共享的方式

5.C/S的连接

6.C/S的服务器设计、实现关键（服务器的调度任务和调度方式，线程技术，服务器缓冲技术）

四、C/S案例（一个C/S模型的文件服务器的设计）

第四讲　分布式体系结构关键技术

一、基于消息传递的通信

分布式进程的通信（发送原语(send primitive)，接发原语(receive primitive)，阻塞(Blocking)原语，非阻塞原语(Nonblocking primitive)，非缓冲原语(unbuffered primitives)，缓冲通信原语－信箱(mailbox)，不可靠(Unreliable)原语，可靠(Reliable)原语

二、远程过程调用（RPC）文档

三、分布式同步算法

逻辑时钟(Logical Clocks)（Lamport 算法）

四、分布式互斥算法

集中式算法

Ricart & Agrawala's算法

令牌环算法(Token Ring)

五、分布式系统的可靠性

1.选择算法(Elect Algoriathms)（Bull算法，环算法(Ring Algorithm)）

2.k-容错技术“两支军队”问题(two-army problem)，Byzantine generals problem）

3.表决算法(voting algorithm)表决技术(votings)

第五讲　层次结构分析

一、线程技术

1-6.引入线程的目的，概念，与进程的区别，分类，执行特性，应用

二、服务器缓冲技术

1.无状态信息(Stateless)服务器

2.有状态信息(State)服务器

3.无状态服务器与有状态服务器的比较

4.无状态服务器与有状态服务器的比较总结

三、N层结构的特性

1.层次结构设计

2.软件系统的层次结构

3.N层结构

四、N层结构的实现

1.一个产品的软件体系结构风格如果采用N层结构。则可能按下面的步骤实施

2.Window NT的层次结构

五、N层结构的优缺点

六、一个用于构造分布式系统的层次结构设计

1.表示层－－用户界面技术

2.业务逻辑层－－应用系统的核心

3.数据逻辑访问层(DAL)

第六讲　CORBA技术及应用实例

一、CORBA概述

CORBA及其发展过程

解决分布式系统的应用程序开发问题的两条规则

对象管理组(OMG，Object Management Group) 对象模型(Object Model) 引用模型(Reference Model)

二、CORBA特性

1.OMG接口定义语言（IDL）

2.语言映射

3.操作调用和调度软件

4.对象适配器（Object Adapter)

5.内部ORB协议

三、CORBA应用程序的一般开发过程

四、CORBA的基本服务

1.命名服务(Naming Service)，命名图(naming graph)，名称解析(name resolve)

2.交易服务(Trading Service)

3.事件服务(Event Service)

五、BES：一个优秀的CORBA开发平台

第七讲　结构设计空间及其量化

一.设计空间和规则（design space and rule)

1.设计空间（design space)

2.维(Deimension)

3.设计规则（design rule)

4.设计知识库（design vocabulary)

5.进一步讨论

二、用户接口结构（user-interface architecture)设计

1.用户接口结构的设计空间

基本结构模型(a basic structure model)

功能维例子(sample functional dimensions) 结构维例子(sample structure dimensions)

三、量化设计空间(Quantified Design Space，QDS)

1.简介

质量配置函数（QFD)

2.量化设计空间(QDS) 通用功能/结构设计空间（Generalized Functional Design Space）

第八讲　软件体系结构描述

一、软件体系结构形式化的意义

1．The architecture of a specific system(指定系统的体系结构)

二、软件体系结构描述的方法

1.主程序和子过程

2．数据抽象和面向对象设计

3．Category Theory(类属理论)

三、Z Notation简介

1．什么是形式规范（formal specifications）

2．Z notation的思想

3．Declarations，4. Schema texts，5. Predicates

四、Z－Notation的应用例子- The birthday book