13.1 什么是控制驱动部分

控制驱动部分是**OOD**模型的外围组成部分之一，由系统中全体主动类构成。这些主动类描述了整个系统中所有的主动对象，每个主动对象是系统中一个控制流的驱动者。

控制流（**control flow**）

——进程（**process**）和线程（**thread**）的总称

有多个控制流并发执行的系统称作并发系统（多任务系统）

为什么需要控制驱动部分

并发行为是现实中固有的

当前大量的系统都是并发系统（多任务系统），例如：

- 外围设备与主机并发工作的系统
- 有多个窗口进行人机交互的系统
- 多用户系统
- 多个子系统并发工作的系统
- 单处理机上的多任务系统
- 多处理机系统
- ……

多任务的设置

- 描述问题域固有的并发行为
- 表达实现所需的设计决策

隔离硬件、操作系统、网络的变化对整个系统的影响

13.2 相关技术问题

（1）由系统总体方案决定的实现条件：

计算机硬件

性能、容量和**CPU**数目

操作系统

对并发和通讯的支持

网络方案

网络软硬件设施、网络拓扑结构、通讯速率、网络协议等

软件体系结构（详后）

编程语言

对进程和线程的描述能力

其它商品软件

如数据管理系统、界面支持系统、构件库等

——对共享和并发访问的支持

（2）软件体系结构

抽象地说，软件体系结构描述了构成系统的元素、这些元素之间的相互作用、指导其组合的模式以及对这些模式的约束 ——Mary Shaw

几种典型的软件体系结构风格

管道与过滤器风格（pipe and filter style）
数据抽象风格（data abstraction style）
面向对象风格（object-oriented style）
隐式调用风格（implicit invocation style）
层次风格（layered style）
仓库风格（repository style）
黑板风格（blackboard style）
解释器模型（interpreter model）
进程控制风格（process control style）
客户-服务器风格（client-server style）

（3）网络环境中的软件体系结构

客户-服务器体系结构

- 一层客户-服务器软件体系结构
- 文件共享软件体系结构
- 二层客户-服务器体系结构
- 三层客户-服务器体系结构

P2P软件体系结构

- 集中目录式P2P-第一代P2P软件体系结构
- 纯P2P-第二代P2P软件体系结构
- 非结构化的层次纯P2P-第三代P2P软件体系结构

一层客户-服务器软件体系结构

- 一层客户端-服务器软件体系结构是在20世纪70年代开始使用的基于大型机的体系结构，如图7.1所示。用户分时使用主机，使用终端通过键盘输入数据到主机。其特点是计算机体积庞大、价格昂贵、维护成本高，等等。典型的大型机包括IBM的Unisys等。
- 一层的客户端-服务器软件体系结构的明显缺点是，所有的业务逻辑与数据文件均在主机上，实际上没有客户端程序，因此实现起来价格昂贵。另外，因为技术限制，当时的大型机基本上没有图形界面，而只能使用键盘输入。由于大型机需要分时操作，因此只能供很少的客户使用。
- 随着时间的推移，企业中需要使用计算机的用户越来越多。为了适应新的需要，公司开始考虑将大型机的应用分成客户端与服务器两部分，这就产生了下节将要介绍的文件共享软件体系结构。

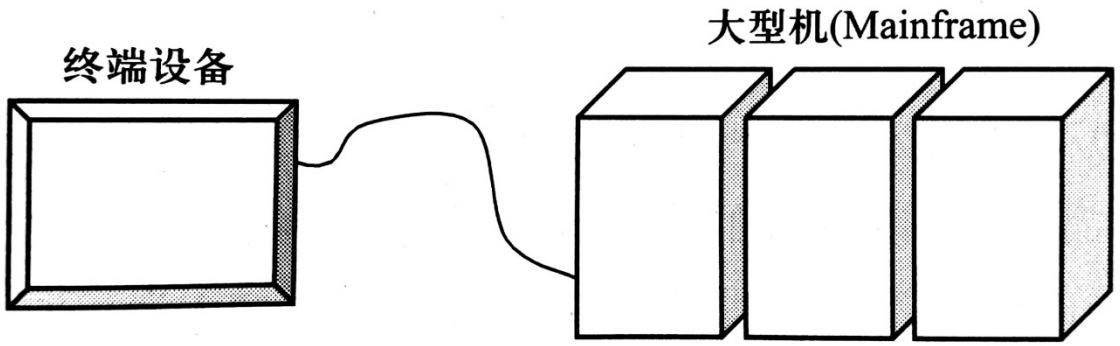


图 7.1　一层的客户端－服务器软件体系结构

文件共享软件体系结构

- 较早的个人计算机网络基于文件共享原理，所有的数据都集中存储在一个文件服务器中，供授权用户访问。该架构图如图7.2所示。
- 工作原理如下

① 客户端发送文件请求，该请求被发送到文件服务器。
② 文件服务器的文件管理系统帮助找到所需要文件。
③ 客户端下载所需要的文件。
④ 客户端更新文件，例如写入新的商业信息等。
⑤ 客户端将更新以后的文件发送给文件服务器。

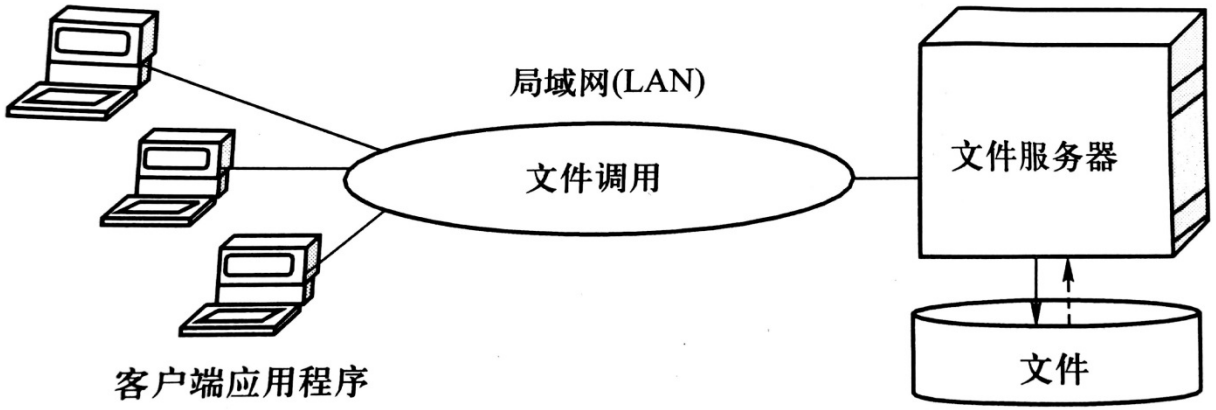


图 7.2　文件共享体系结构——所有的业务逻辑都在客户端

文件共享软件体系结构

- 文件共享软件体系结构的缺点如下。

① 该软件体系结构只有在共享用户使用率比较低，并且数据转移量也较低的情况下才能有效地工作，即该体系结构只能支持少量的用户。

② 业务逻辑完全在客户端，所以要修改业务逻辑时必须要修改每个客户端。

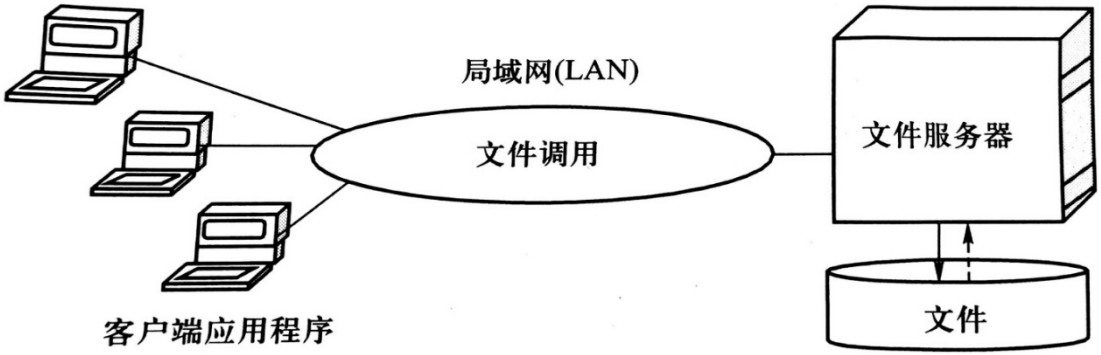


图 7.2　文件共享体系结构——所有的业务逻辑都在客户端

两层客户端-服务器软件体系结构

- 在两层客户端-服务器体系结构中，通常使用SQL实现客户端和服务器之间的通信，其中，服务器可能支持存储过程和触发器，这就意味着服务器可以被编程实现业务逻辑，而事实上，业务逻辑在服务器上运行要比在客户端上运行更合适。所以该体系结构是一个更有效的整体系统。两层客户端-服务器体系结构的结构图如图7.3所示。
- 在两层客户端-服务器体系结构的内容中，第一层包含用户图形界面、应用程序和部分业务逻辑；第二层包含部分业务逻辑、遗产数据与其他资源管理系统、数据库管理系统与共享资源等。
- 两层客户端-服务器的工作原理如下。

① 客户端发送SQL语句进行一项数据查询，或者要求更新数据库。

② 服务器端数据库管理系统可以协助用户查询，并且将查询结果返回给客户端；或者更新数据库，例如修改一个表中某一个域的值，或者写入新的商业记录。

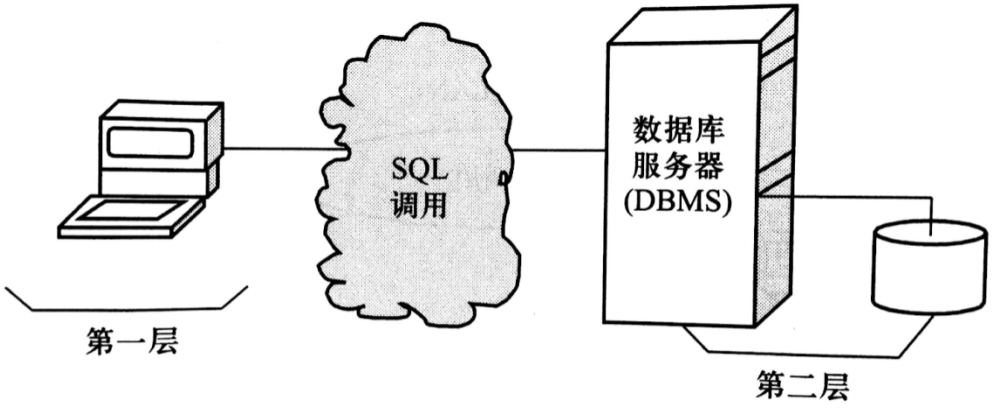


图 7.3　两层客户端－服务器软件体系结构

三层客户端-服务器软件体系结构

- 为了克服两层软件体系结构的局限性，人们在用户系统接口客户环境和数据库管理服务器环境之间，引入一个中间层。带有这样的中间层的体系结构被称为三层客户端-服务器体系结构(3-Tier Client/Server)，如图7.4所示。
- 三层客户端-服务器体系结构的内容为，第一层：包括运行在客户端操作系统中的图形用户界面或者是网页浏览器。第二层：通常包括所有的业务逻辑处理程序，也可能包括为浏览器生成网页内容的图形用户界面处理程序。第三层：包括数据库管理系统、遗产数据库管理程序与数据库。

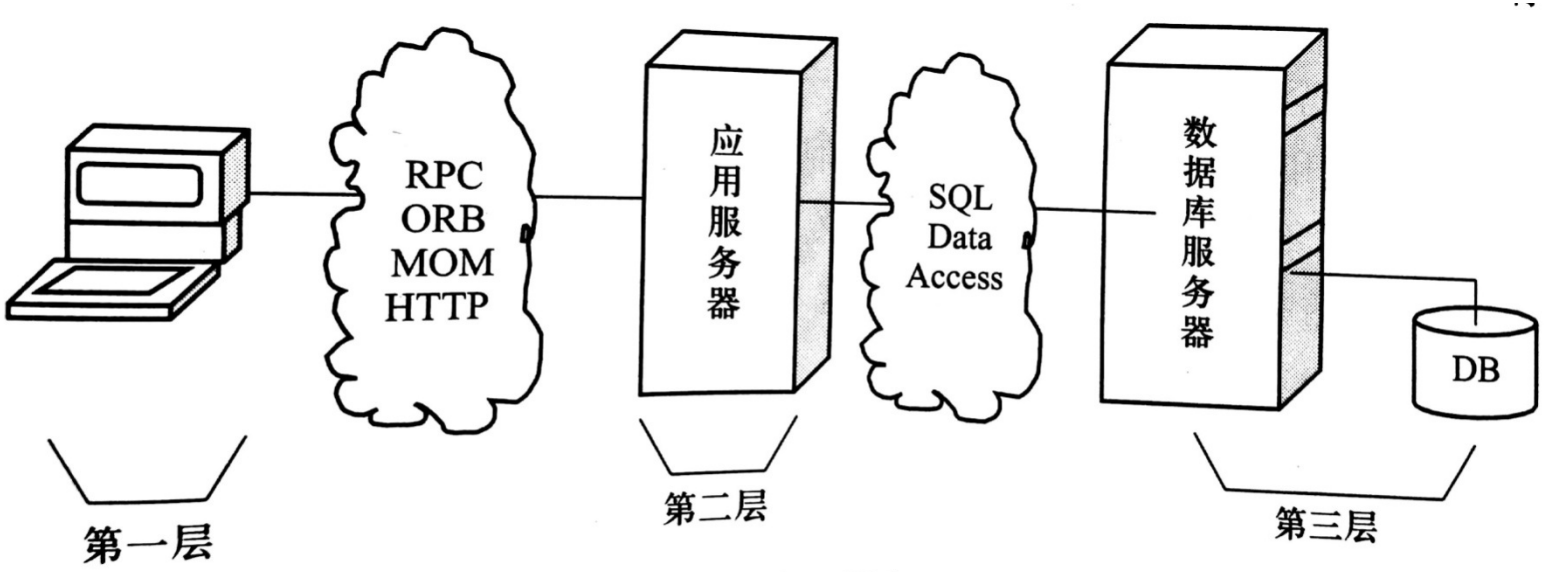


图 7.4　三层客户端－服务器软件体系结构

集中目录式P2P-第一代P2P软件体系结构

- 集中目录式P2P软件体系结构采用中央目录服务器管理P2P各结点，P2P结点向中央目录服务器注册关于名称、地址、资源、元数据、所能提供的服务等自身的信息，但所有内容存储在各结点中而非服务器上。当一个结点要查找某种服务的时候，该结点首先要连接到目录服务器，然后根据目录服务器中的信息查询以及网络流量和延迟等信息来选择与定位其他对等点。一旦找到了所要连接的结点，则直接与其建立连接，而不必经过中央目录服务器进行。

- 集中目录式软件体系结构的结构图如图7.9所示。在该图中，每个结点都知道目录服务器的地址，虚线代表每个结点随时准备连接到目录服务器，实线代表各对等结点之间的连接。在图中，除了目录服务器以外，其他每个对等结点之间都有连接，从而形成了一个网络。目录服务器形式上是该对等网络的“中心”。但是，实际上目录服务器除了提供各结点的基本的已注册信息之外，并不真正地承担任何其他的功能，因此，该体系结构与传统的客户端 - 服务器体系结构还是有本质的区别。

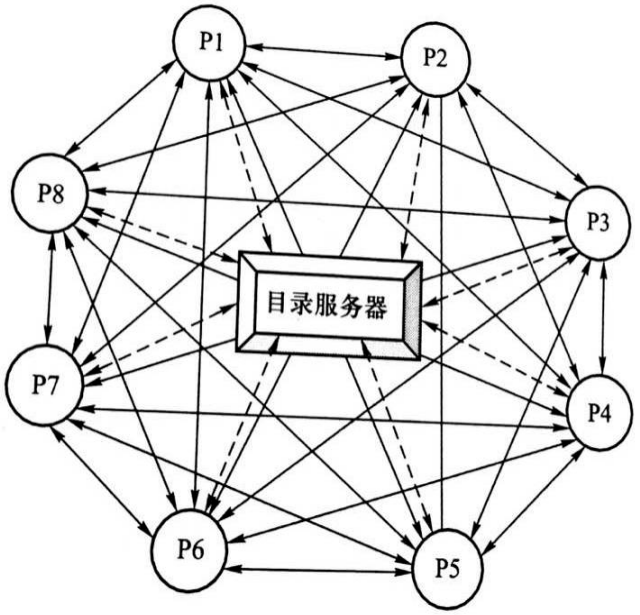


图 7.9　集中目录式 P2P 的软件体系结构图

集中目录式P2P-第一代P2P软件体系结构

- 第二页：发行服务与执行服务过程+缺点+图7.10
- 集中目录式P2P软件体系结构中，发现服务与执行服务的过程如图7.10所示。

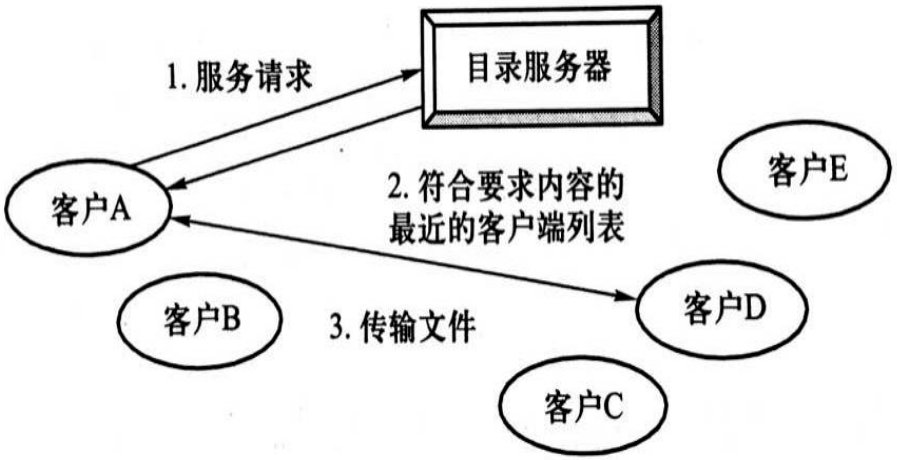


图 7.10　集中目录式 P2P 软件体系结构中发现服务与执行服务的过程

- 在图7.10中，客户A想要在对等网络中发现某种服务，则：
  （1）客户A发出请求给目录服务器。
  （2）目录服务器从列表中找出A所需要的对等点。
  （3）客户A与新对等点直接交流。本图中是从客户D下载文件到客户A。
- 缺点：网络的稳定性较差。一旦服务器失效，则该服务器下的对等结点可能全部失效。

纯P2P-第二代P2P软件体系结构

- 纯P2P体系结构也被称为广播式的P2P模型，它没有集中的中央目录服务器，每个用户随机接入网络，并与自己相邻的一组邻居结点通过端到端连接构成一个逻辑覆盖的网络。每个对等结点都可以既作为客户端又作为服务器，且与它们的邻居都偶遇相同的能力。每个对等结点都作为网络中的一个结点，没有中心路由器。原始的纯P2P体系结构图如图7.11所示。
- 图7.11

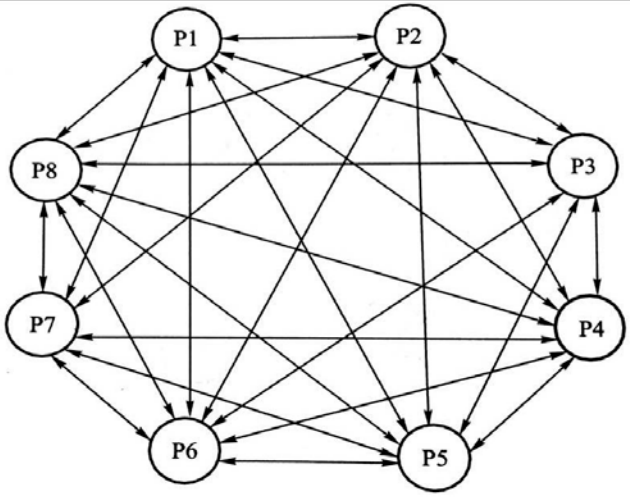


图 7.11　纯 P2P 软件体系结构图

- 对等结点之间的内容查询和内容共享都是直接通过相邻结点广播接力传递，同时每个结点还可以记录搜索轨迹，以防止搜索环路的产生。纯P2P网络结构解决了网络结构中心化的问题，扩展性和容错性较好。

纯P2P-第二代P2P软件体系结构

- 纯P2P体系结构发现服务与执行服务的过程如图7.12所示，具体的询问步骤如下：

（1）结点P1首先连接到网络。

（2）结点P1用ping数据包以广播的方式发送消息给其他的临近结点，以便寻找的网络中的在线结点。

（3）在线结点用pong数据包回答，提供在线结点的信息，包括IP地址、端口号、可共享的文件数目等。

（4）发送query查询消息，一个询问消息被用于搜寻在网络中的其他结点共享的文件，它包括含询问字符串与最小要求的链接速度。

（5）得到query hit回答消息。P1询问回复消息包含匹配一次特定询问的所有文件的名单、每个文件的大小和反应结点的链接速度。

（6）使用push消息上载文件到客户。

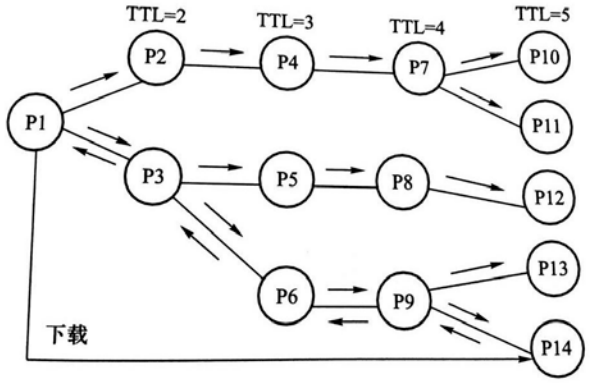


图 7.12　纯 P2P 软件体系结构中发现服务与执行服务的过程

纯P2P-第二代P2P软件体系结构

- 纯P2P体系结构具有如下缺点：

（1）服务发现限制与困难。在大型P2P网络中，查询会遇到查询范围小的困难。

（2）由于没有一个对等结点知道整个网络的结构，网络中的搜索算法以泛洪的方式进行，控制信息的泛滥消耗了大量带宽并且可能很快造成网络拥塞甚至网络的不稳定，从而导致整个网络的可用性较差。

（3）系统容易受到垃圾信息，甚至是病毒的恶意攻击。

非结构化的层次纯P2P-第三代P2P软件体系结构

- 传统的纯P2P体系结构在运行中存在很多问题，主要体现在基于泛洪的第二代P2P网络中当网络变得很大时，系统的搜索/回答率会变得很低。为了解决该问题，代表基于层次的第三代P2P系统在2001年应运而生。通过将对等结点分为超级结点与叶子结点两大类的方法，让超级结点形成一个建立在原来的对等网络之上的覆盖网络的方式，引入了层次的概念。这种改良体系结构为非结构化的层次纯P2P体系结构，其网络拓扑结构如图7.13所示。

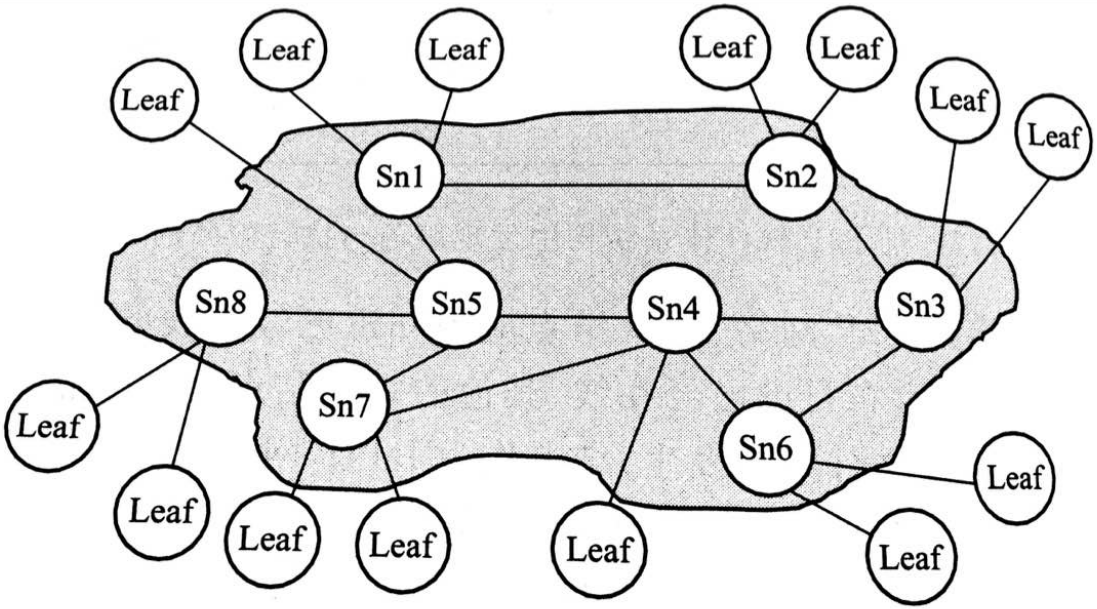


图 7.13　非结构化的层次纯 P2P 体系结构网络拓扑结构图

（4）系统的并发性

进程（**process**）概念出现之前，并发程序设计困难重重

主要原因：

并发行为彼此交织，理不出头绪

与时间有关的错误不可重现

进程概念的提出使这个问题得到根本解决

进程的全称是顺序进程（**sequential process**），其基本思想是把并发程序分解成一些顺序执行的进程，使得：

每个进程内部不再包含并发行为

所以叫做顺序进程，其设计避免了并发问题

多个进程之间是并发（异步）执行的

所以能够构成并发程序

线程（Thread）

由于并行计算的需要，要求人为地在顺序程序内部定义和识别可并发执行的单位。
因此后来的操作系统大多支持线程概念。

线程与进程的区别：

进程既是处理机分配单位，也是存储空间、设备等资源的分配单位（重量级的控制流）；

线程只是处理机分配单位（轻量级的控制流）；

一个进程可以包含多个线程，也可以是单线程的。

控制流是进程和线程的总称。

应用系统的并发性

从网络、硬件平台的角度看：

- 分布在不同计算机上的进程之间的并发
- 在多**CPU**的计算机上运行的进程或线程之间的并发
- 在一个**CPU**上运行的多个进程或线程之间的并发

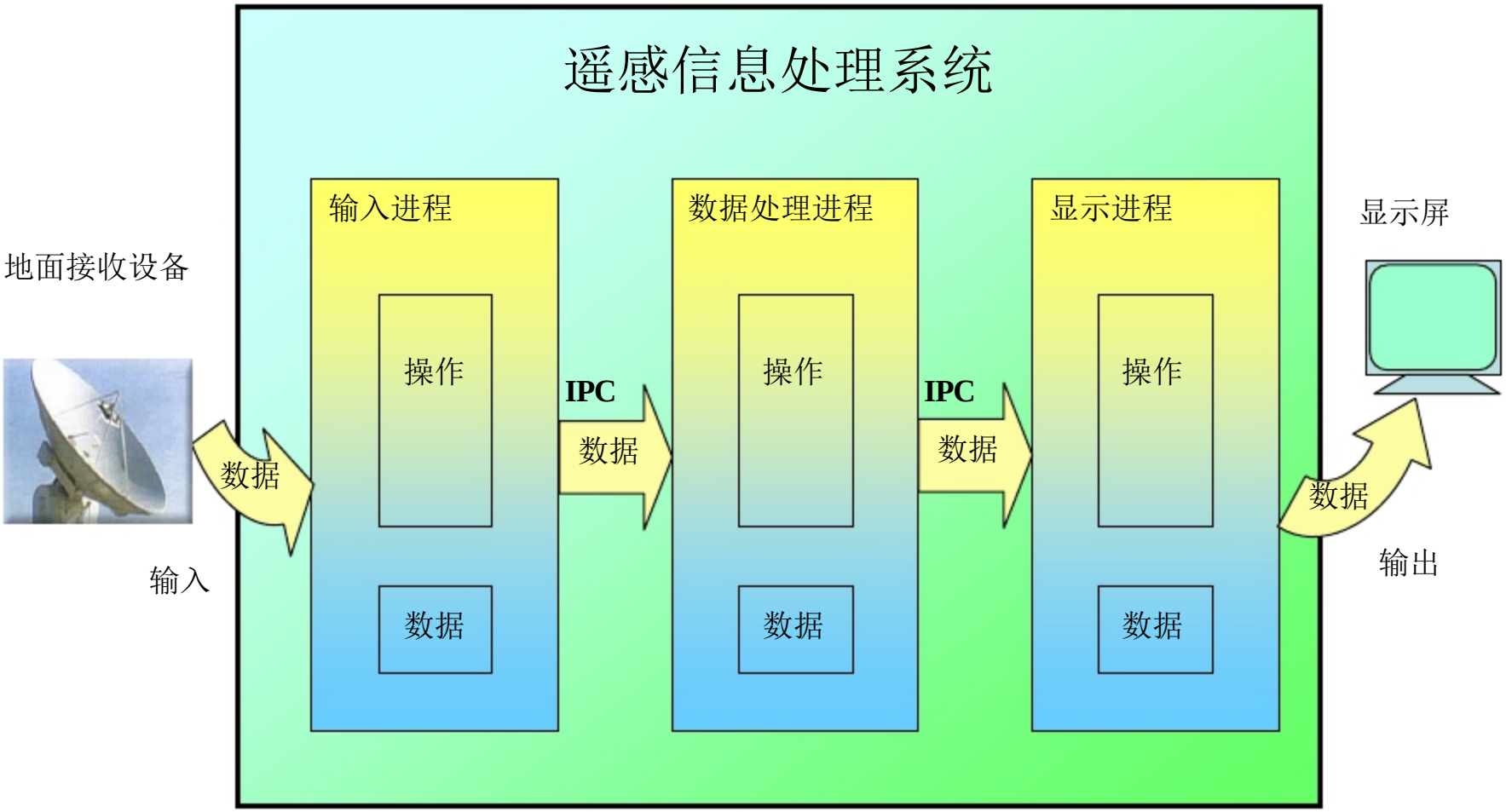
从应用系统的需求看：

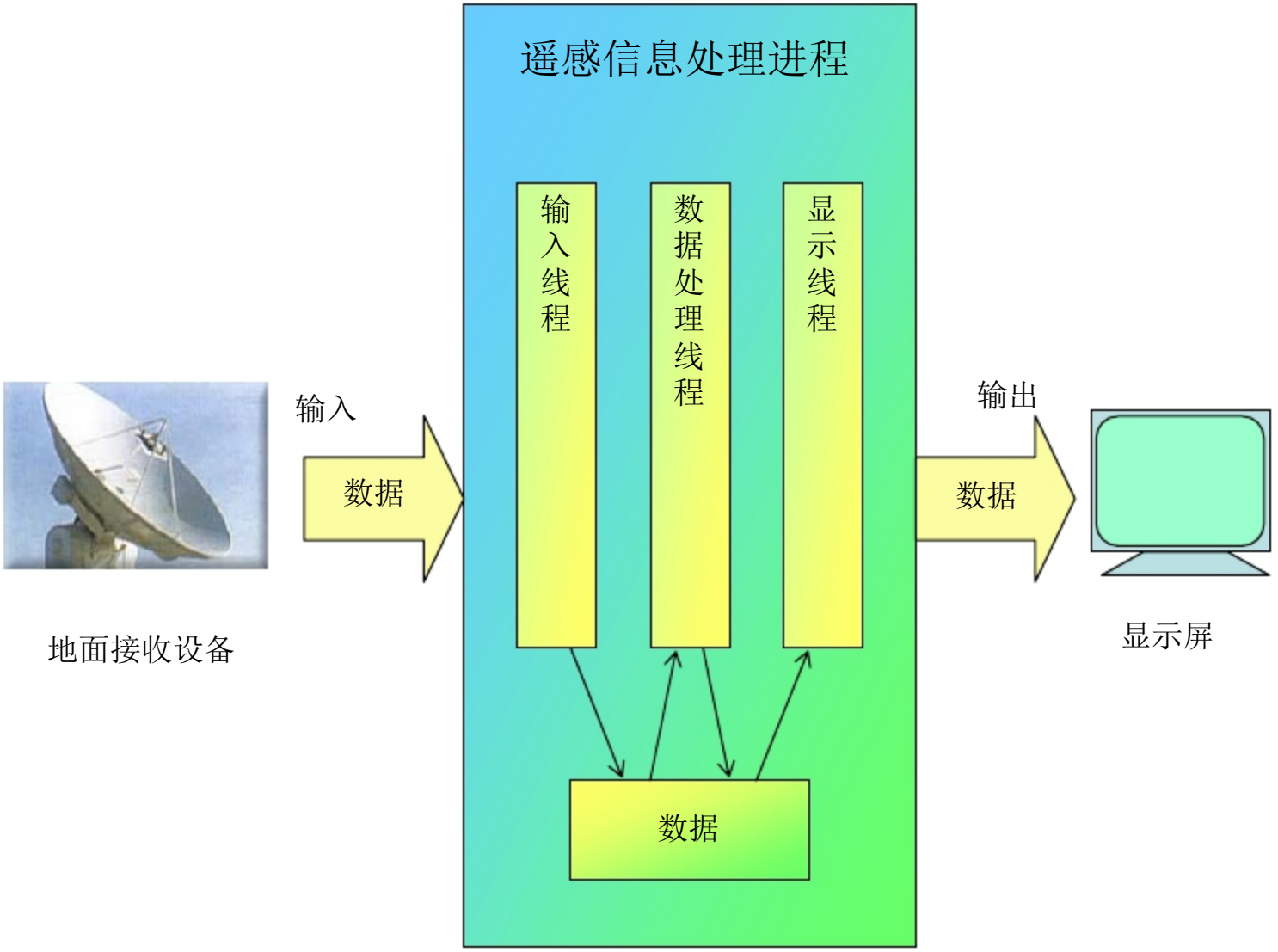
- 需要跨地域进行业务处理的系统
- 需要同时使用多台计算机或多个**CPU**进行处理的系统
- 需要同时供多个用户或操作者使用的系统
- 需要在同一时间提供多项功能的系统
- 需要与系统外部多个参与者同时进行交互的系统

处理应用系统并发性的例子

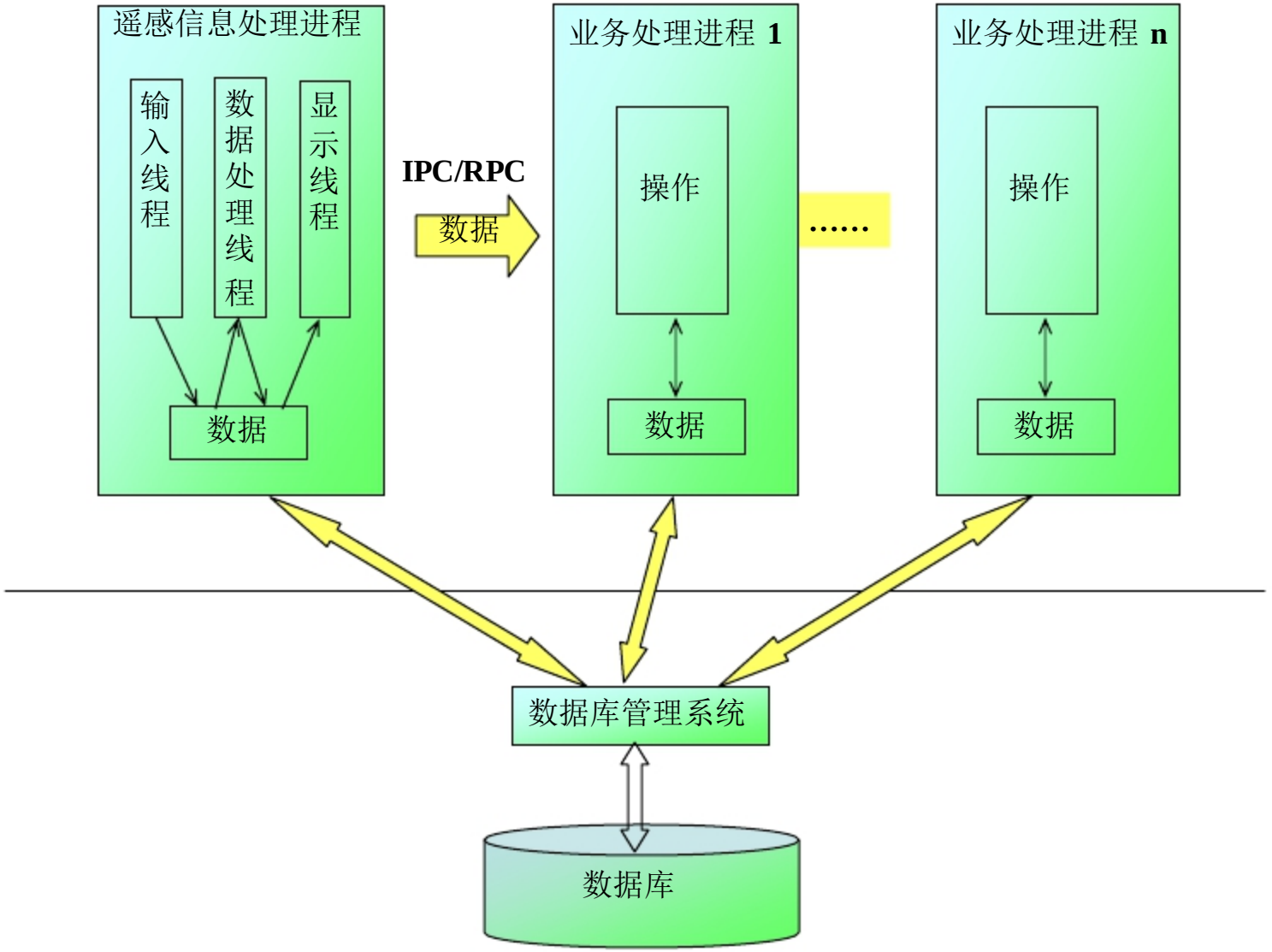
见《面向对象的分析与设计》**13.2.4.3**节

例**1 ~** 例**7**


遥感信息处理系统
地面接收设备
输入
数据
输入进程
操作
数据
IPC
数据
数据处理进程
操作
数据
IPC
数据
显示进程
操作
数据
数据
显示屏
输出


遥感信息处理进程
输入线程
数据处理线程
显示线程
数据
输入
数据
输出
数据
地面接收设备
显示屏

同时采用多进程和多线程



13.3 如何设计控制驱动部分

（1）选择软件体系结构风格

一层客户-服务器软件体系结构

文件共享软件体系结构

二层客户-服务器体系结构

三层客户-服务器体系结构

集中目录式P2P-第一代P2P软件体系结构

纯P2P-第二代P2P软件体系结构

非结构化的层次纯P2P-第三代P2P软件体系结构

（2）确定系统分布方案

系统分布——包括功能分布和数据分布

在面向对象系统中都体现于对象分布

原则：

减少远程传输，便于管理

对象分布

软件体系结构

系统功能在哪些结点提供

数据在哪些结点长期存储管理，在哪些结点临时使用

参照用况　把合作紧密的对象尽可能分布在同一结点

追踪消息　把一个控制流经历的对象分布在同一结点

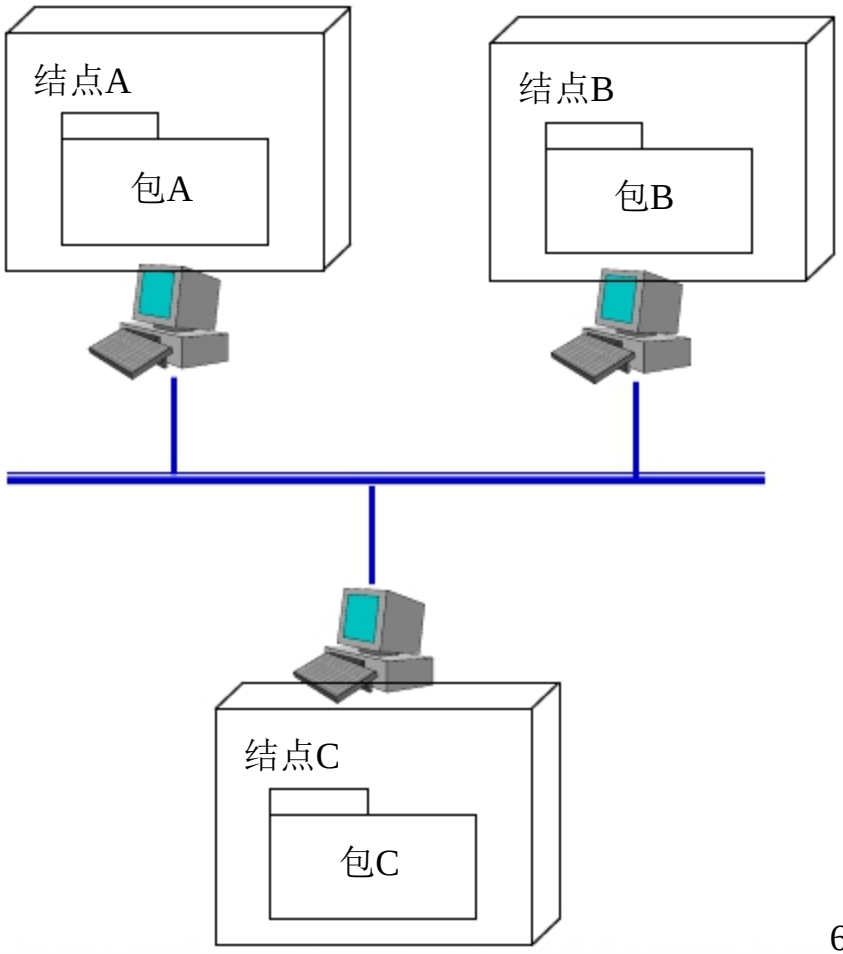
对象分布通过类的分布体现

类的分布：根据对象分布的需要

考虑分布方案之前，将系统暂时看作集中式的
确定分布方案之后，将对象分布到各个处理机上
以每台处理机上的类作为一个包

集中式
类图

分布到不同结点上

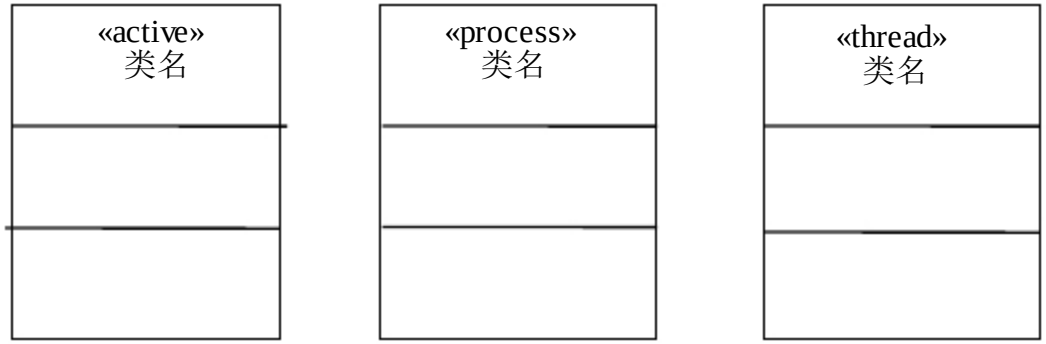


13.3.3 识别控制流

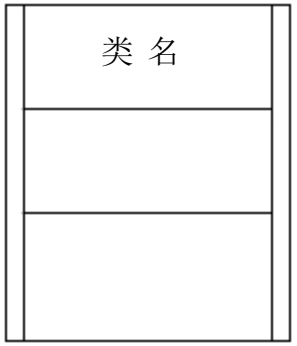
（1）以结点为单位识别控制流
不同结点上程序的并发问题已经解决
考虑在每个结点上运行的程序还需要如何并发
（2）从用户需求出发认识控制流
有哪些任务必须在同一台计算机上并发执行
（3）从用况认识控制流 关注描述如下三类功能的用况
要求与其他功能同时执行的功能
用户随时要求执行 的功能
处理系统异常事件功能
（4）参照**OOA**模型中的主动对象
（5）为改善性能而增设的控制流
高优先级任务、低优先级任务、紧急任务
（6）实现并行计算的控制流（线程）
（7）实现结点之间通讯的控制流（进程）
（8）对其它控制流进行协调的控制流

13.3.4 用主动对象表示控制流

控制流是主动对象中一个主动操作的一次执行。其间可能要调用其他对象的操作，后者又可能调用另外一些对象的操作，这就是一个控制流的运行轨迹。



UML1的主动类表示法

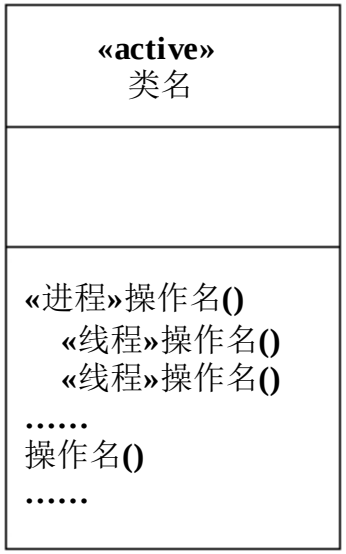


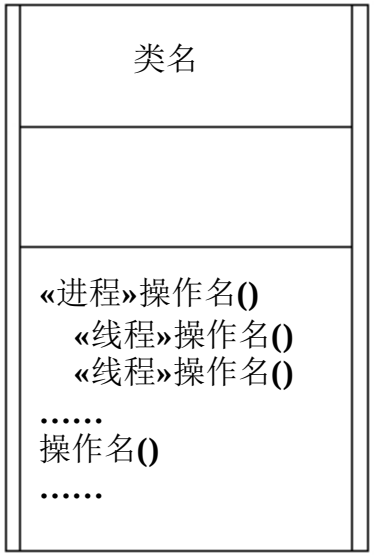
UML2的主动类表示法

问题：

一个主动类可以有多个主动操作和若干被动操作，这种表示法不能显式地表示哪个（哪些）操作是主动操作。

用关键词表示主动操作





显示地表示由进程创建线程