计算机系统 消息系统 ③

计算机科学1993 Vol.20 No.5

11—16

消息系统的研究

TP3

蔡希尧 王小民 边平定

(西安电子科技大学软件工程研究所 西安 710071)

摘 要：

本文讨论计算机系统中消息系统的主要问题，包括消息的传递、组织和管理等方面，并列举了并发程序设计、图形用户界面、面向对象数据库、分布系统、以及PC机所涉及的消息问题所采取的策略和措施。文章最后提出了若干需要研究的问题。

一、前言

在计算机系统中，消息的地位和作用越来越引人注目。在并发程序中，进程间的通信方式，采用消息传递；在面向对象的系统中，对象和消息传递是构成系统的两个基本元素；在图形用户界面中，依靠多种消息把系统联系在一起，使之运转；新一代的分布式操作系统中，有的是以消息为基础构成的；还可以举出更多的例子。因此，在计算机系统中，消息往往自成一个子系统，有关消息的组织、消息的传递、消息的管理等，是消息子系统的设计和运行中所关注的问题，而这些问题则关系到整个计算机系统的正常运行。因此，对消息系统的理解和掌握，就成为计算机技术的一个重要方面。

消息这个词，是新闻界和通信系统中沿用已久的。信息论创立以后，也使用消息这个词，作为载有信息的某些文本、语句等的总称。例如一份电报就是一个消息。当接收者收到电报时，事先对它的内容一无所知，则他得到了很多信息；如果已知一部分内容，则所得的信息少；如果电报的内容全部已知，那就没有得到信息。所以接收者收到消息时所获得的信息量，取决于对内容的无知程度的改变，这样，把信息和消息严格地区别开来。

在计算机技术中，信息和消息也有不同的涵义。信息是一种泛指的概念，而消息则有特定的意义。消息首先被应用于并发程序中，是一种程序构造，用以实现进程间的通信。在图形用户界面（GUI）中，广泛地使用消息，因为GUI的工作是事件驱动的，而每一事件总是与相应的消息联系在一起，是消息的到达引起事件的发生。在面向对象系统（OOS）中，消息传递是程序运行的基本机制，对象只在消息到达时才予以响应，显现其行为。GUI和OOS等系统中，消息的数量往往比较大，这就需要很好的组织和管理。

本文将就消息系统有关的三个主要问题：消息传递、消息组织和消息管理展开讨论。

二、消息传递

消息传递是对象（进程）和对象（进程）之间的通信机制。在早期的并发程序中，进程之间是通过共享变量来实现通信的。消息传递则不依靠共享变量，通过某种形式的信道，以消息交换来实现通信。在OOS中，对

蔡希尧 教授，博士生导师。王小民、边平定 博士生

象之间的通信，一律采用消息传递的方式。

为实现消息传递，语言所提供的基本设施是两个原语：send和receive。当对象A要向对象B发送消息时，执行send操作，语句的形式是：

send expression-list to destination-designator

语句中给出了接收者B的名。B则执行receive操作，并指出消息源A的名，语句的形式是：

receive variable-list from source-designator

这是目前一些语言中所采用的主要方式，在这里，给参加通信的对象命名是很重要的。

在消息传递的系统中，通信双方直接经过信道通信，还是在信道中设置缓存(信箱)，有明显的区别。首先是信道的容量有所不同，设置了缓存以后，信道中可以存在多个消息；执行send语句时，只要缓存未满，可以立即发送消息而不会延时；接收者获取消息的时间灵活。有了缓存以后，发送者到达send语句，不必等待接收者是否到达receive语句，在发送以后，可以继续其他的工作。但互相指名的直接通信也有优点，这就是可靠性较高。现在，多数消息传递系统中都设置缓存，对缓存命名，并采取适当的管理。

下面讨论消息传递中的几个特殊问题：

1. 消息传递的类型

设A和B是互相通信的两个对象，A发送消息给B。我们按发送发对象A是否等待B的接收，以及是否等待B返回结果，可以把消息传递分成以下三种类型：

类型1：A发送消息M给B，在发送M以后，不等待B是否收到M，A立即继续以下的工作。

类型2：A发送消息M给B，不仅等待M为B所接收，还等待B返回结果。

类型3：A发送消息M给B，希望得到某种结果，但并不立即需要这一结果，于是A在发送M以后，去做其他的事，B在处理M以后，将结果放在缓存中，待A需要时取用。

这三种类型的消息传递，在一些系统中已经采用。除了这些不同的类型以外，有的语言，如ABCL/1，仿照信件的传递有平信和快信之分，把消息传递分为普通和快速两种模式[1]，以普通模式传递的消息，被发往接收者时，排在队的末尾，按先到先处理的顺序被接收。以快速模式传递的消息，另设专用的缓存，它们的处理，优先于以普通模式传送的消息。

2. 命名和联编

在同一系统中，消息的名应当是唯一的。在没有设置缓存的信道中，发送对象和接收对象显式地互相指明对方的名，一对一地进行通信，这时候的命名方式叫做“直接命名”。这种命名的方式直观易懂，但当一个对象的名改变时，所有和它通信的其他对象的通信语句，都要做相应的改变。从分别编译的角度来看，显式命名的方式也不是所希望的。

在设置缓存（信箱）的信道中，通信是间接的，收发双方都要使用信箱，要给信箱命名。这时候，通信除了一对一以外，还可以一对多或多对一。当一个对象发送消息给一个信箱，允许多个对象取用消息时，可能发生竞争而引起冲突，要有互斥措施以解决冲突。

命名的过程中要进行联编，有静态和动态两种联编方式。静态联编时通信双方的名在编译时是已知的，动态联编则在运行过程中进行。静态联编应用较多，但固定了通信双方，不能用于动态分配通信资源的场合，排除了一个对象沿着一条在编译时还不知道的信道实现通信的可能性，限制了在变化环境中消息传递的能力。动态联编可以克服这些缺点，比较灵活，但实现比较困难。

3. 基于消息的操作系统

现代的分布式操作系统中的一类，是以消息传递为基础构成的，被称之为“基于消息的操作系统”，操作系统V[2]，Amoeba[3]，Nexus[4]等均属于这一类，它们使用显式的

消息以支持进程间通信。而通常的操作系统如UNIX，对系统的服务是采用受保护的过程调用提出请求的，不是显式的消息传递。这些操作系统的共同特点是：(1) 一个比较小的核，在分布式系统的每个结点上都配以核的副本；(2)服务进程和核分离；(3)核主要管理进程间的通信和一些必需的控制；(4) 进程间的通信用消息传递；(5) 结点间的通信用远方过程调用 (RPC)。这类分布式系统，都是以顾主-服务员方式工作的。RPC就是前面提到第二种类型的消息传递，它事实上已经成为顾主-服务员工作方式的通信标准。

三、消息的组织

在复杂的系统中，消息的数目很多，为了便于管理，把消息加以分类，是组织消息的基本方法。

现代的图形用户界面 (GUI)，往往含有数量很多的消息，例如Microsoft公司的"窗口"系统和1988年宣布的和OS/2配合使用的PM (Presentation Manager)，它们的结构都是基于消息的，并支持类构造。它们定义了许多通用的消息，并能生成新的消息。例如在PM中，设置了100多条通用消息，和150多条专用消息，分成四大类：

1) 输入消息：响应键盘和鼠标的输入而生成的消息。

2) 系统消息：响应程序中的事件或一个系统中断而生成的消息。

3) 控制消息：用于同一个子窗口的双路通信。

4) 用户消息：由程序员定义，在某种应用中用以对一个预定义的事件发生时进行数据传递。

MIT所设计的X窗口，是以网络协议来定义的，这些协议说明了消息传送的格式和意义，并把应用和工作站之间通信的消息分成四类：

1) 单路请求消息：这是请求工作站做某一事情的消息，其中含有用户给出的信息。由于消息是单路传递的，请求发出以后，就可以去做以后的工作。信道中设有缓存。

2) 来回请求消息：应用向工作站发出请求，等待回答。在等待期间，应用程序不能做其他的工作。

3) 事件消息：这是和事件联系在一起的消息。X窗口是事件驱动的，系统提供33种不同的事件类型，并把它们分成五个种类，和每一事件联系的消息也随之分类。

4) 错误事件消息：在单路请求的情况下，可能会出现这样的问题：工作站上的服务程序发现请求消息中有错误，但这时候应用程序已经在做其他的事，为解决这种问题，可使用错误事件消息，由工作站发给应用程序。

在面向对象的数据库 (OODB) 中，也往往设置许多消息，并分成类型。例如在ORION数据库中，消息被分成四种类型：

1) 实例属性消息；

2) 类属性消息；

3) 实例方法消息；

4) 类方法消息。

(1) 和(2)两类消息通称为"属性消息"，(3)和(4)两类消息通称为"方法消息"。

把消息分类予以组织，显然有利于管理，但能够这样做的系统，消息往往是预先设定的，前面列举的GUI和OODB中的消息分类，属于这一情况。有的系统，消息是在运行中生成的，只能伴随着生成它们的对象或/和这些消息同时生成的对象来加以组织管理，演员系统 (actor system) 就是属于这一情况的典型例子[5]。在演员系统中，每个演员是一个对象，都能够发送或接收消息，并在接收消息后执行动作。一个正在执行动作的演员所生成的作业包括一个消息和一个目标演员。在这个系统中，消息本身也被看做演员，是由某个演员生成的非串行化的演员 (只有简单功能描述的演员)。这样，一方面是消息和演员的动作联系在一起，另

一方面消息也是演员，消息和演员混为一体，系统的配置是一种瞬时描述，是变化的。

四、消息的管理

消息的管理包括对消息的存贮，消息流的控制，消息的处理等方面，和信道的构造、系统结构有密切的关系，往往需要硬件的支持，其中缓冲存贮（或称信箱、队）的安排，是管理中的重要一环。

在GUI中，消息数量很多，往往设置不同层次的缓存，例如"窗口"和PM系统中，设有多个缓存，称之为队，并分称为系统队和应用队。PM的消息队以及消息流的情况如图1所示。每一个输入消息，首先进入系统队，然后加以分路，送到各自的应用队去。原始的输入在进入系统队以前，要转变成标准的消息格式。分路则是按照输入的当前状态，或程序中某一事件的要求来决定的。

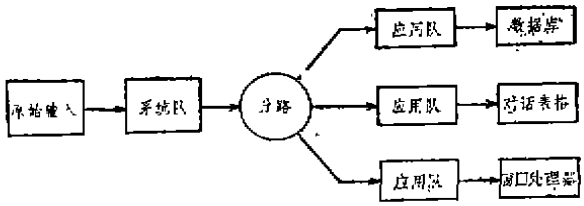

图1 PM中的消息队和消息流

在面向对象的数据库系统中，如前面提到的ORION，提供了相当数量的消息，使用户可用以控制数据库的资源和完整性，控制的范围包括对象的组合、模式的变换、事务的管理等。为此，ORION专门设置了一个消息管理部件，接收发送给ORION的所有消息，并加以处理。这个部件的主要数据结构是快速缓存，分成两部分：一部分是"实例消息快速缓存"，存放实例的属性和方法的消息；另一部分是"类消息快速缓存"，存放类的属性和方法的消息。图2是ORION的结构[6]，消息管理部件和其他三个子系统都有联系，这表明用户对于数据库中的各个子系统，都可以通过消息管理部件加以控制。

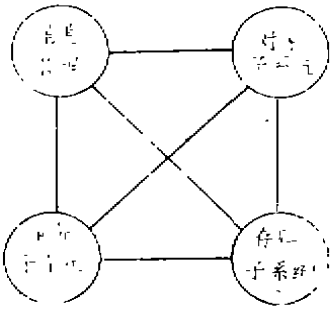

图2 ORION的结构

在分布式系统中，为了使系统能够符合实用的要求，基本消息的交换必须是快速的。基本消息是长度比较短的、使用次数多的消息，经验表明，限制这些消息交换速度的因素不是网络的带宽，而是处理开销，因此，常常设置一个专用的消息处理部件，以满足结点之间的通信要求。图3是一个面向对象的分布式计算机DOOM的结点组成[7]，这个计算机执行面向对象程序设计语言POOL-T的程序。系统的每一个结点，除了有数据处理部件以外，还有一个通信部件，其他结点发来的消息，先进入通信部件，然

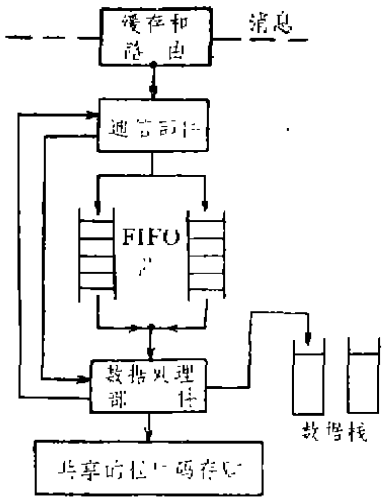

图3 DOOM的结点组成

后进入先进先出（FIFO）队中，这些队是和数据处理部件连接的。每个方法有一个对应的队。通信部件到数据部件还有直接通路，前者到后者的直接通路用以传送其他结点发

来的应答消息，后者到前者的直接通路用以传送向其他结点发送的消息，包括应答消息。

目前，消息管理已经发展到PC机系统，特别在联网的情况下。八十年代微机的蓬勃发展，使得各种各样的应用必须考虑使用微机，才能够推广，才能够开拓市场，即使是很大的计算，在用户方面仍然只是一个微机，真正的计算则是通过网络，在某个或多个主机上进行。用户通过自己桌子上的微机（配以必要的图形用户界面），对计算实现控制。这样，必须让用户对应用消息有清楚的了解，有更大的控制。例如在用户干预计算时，需要了解消息被发送时的状态；当外来的消息触发某种事件时，用户要知道发生了什么，应当去做什么。显然，这种种情况，都需要消息的参与，对消息进行必要的管理。

在IBM的PC/DSNX系统中，采用了以下的消息管理策略[8]：把消息文本从应用码中分离出来，并把管理消息的码从应用码中分离出来，应用码和消息管理码之间的通信经过一个接口，为每一应用模块提供一个唯一的消息标识。此外，所有的消息还被分成类（Class）和分割（Severities），类是按消息内容的类型来划分的，分割则按用户对于应用的干预等级来划分。有关类和分割的消息均包含在消息的标识之中。

五、讨论

前面就消息的传递、组织和管理的问题作了讨论，并且列举了并发程序设计、图形用户界面、面向对象数据库、分布系统和PC机中所采取的措施和策略，说明了消息的问题所涉及的范围和现有的解决办法。消息的传递、组织和管理也可以通称之为消息管理，其目的是为了达到最有效地利用消息，计算机硬件和软件如何最好地支持消息的有效利用，还没有定论。

影响消息使用效率的关键在于消息在缓存中的管理问题，特别是有些在缓存中排队的消息，是否符合与缓存相联系的对象的接收条件并不明确，更加影响效率。因此引出对消息管理上值得探讨的问题：

1. 通常强调的是对消息的处理。但检验消息是否符合接收条件也是重要的，并且处理和检验是不同的操作，应予以区别。当消息准备进入缓存时，应有专门的设施对它们进行检验，只有符合接收条件的消息才允许进入缓存，这是提高效率的一种措施。

2. 让消息具有主动性，在缓存中排队的消息能够主动地提出对接收者的请求，这样做的得失如何？

3. 当一个消息所联系的对象处于忙的状态，不能接收消息，那么，它在队中等待，还是找另一个相似的对象呢？（例如找一个具有消息中所指明的方法的对象）。如果第二种作法更有利，那么，在消息中指明的不应是对象，而是对象的类。

4. 设置一个对象，能够为在缓存中排队的消息分配一个合适的对象，或类的一个实例。

用消息传递进行通信，用消息触发事件，都带有模拟人的通信的性质。凡是拟人的方式，往往是自然简单的，但欲形式地予以描述和实现，达到有效的利用，却又是不容易的。消息系统中的这些有待探讨的问题，也是如此。

参考文献

[1] Yonegawa, A. and et al., Modeling and Programming in an Object-Oriented Concurrent Language ABCL/1, in<Object-Oriented Concurrent Programming>, ed. by A. Yonegawa and M. Tokoro, The MIT Press, 1987

[2] Cheriton, D. R., The V Distributed System, CACM, Vol.31, No.3 1988

[3] Mullender, S.J. and et al., Amoeba: A Distributed Operating System for the 1990s, IEEE Computer, Vol.23, No.5 1990

计算机科学1993 Vol.20 No.5

对象标识的语义及构成

李天柱 （河北大学 河北保定071002）

摘 要

Based on the three database schema in ANSI/X3/SPARC, the full object identity, including objective object identity, conceptual object identity, internal object identity, is introduced. The objective object identity is a special primary key, the internal object identity corresponds to the object identity oid in usual object-oriented systems. The object equality based on the full object identity is also discussed. The fall object identity can be used not only to identify the objects in user interface, but also to manage the objects in computer systems, makes object identity further perfect. And it is also available in combining object-oriented model with value-oriented model.

1. 引言

在面向对象模型中，对象标识是一个重要概念，文〔3，9〕均将其做为O-O模型的必备特征。在NF^2关系模型中，也有人引入了对象标识(元组标识和关系标识)。现在人们之所以重视对象标识，是因为在描述嵌套结构的复杂事物、处理数据共享、描述历史数据和版本特征等方面，对象标识具有重要作用。文〔10〕强调了由系统管理、独立于内容、独立于结构、独立于地址的强标识；文〔15〕指出了对对象标识的错误理解，提出了怎样使强标识与唯一访问（unique reference）协同存在的问题；文〔7〕指出，在扩展的关系模型中，只有不存在不变的key的情况下，才引入对象标识；文〔9〕指出，标识的概念包括比较运算、逻辑标识、对象标识、对象生成。究竟对象标识的语义是什么，怎么构成，作用是什么，是值得深入研究的重要问题。本文主要就面向复杂对象数据库中的对象标识进行进一步讨论。

2. 不同级别模式下的对象及对象标识

数据库中的数据模型是对客观事物的描

李天柱 副教授. 主要研究方向为数据库与知识库. 收到日期：92-09-19.

〔4〕 Tripathi, A.P., An Overview of the Nexus Distributed Operating System Design, IEEE Trans. on SE, Vol.15, No.6 1989

〔5〕 Agha, G., An Overview of Actor Languages, SIGPLAN Notices, Vol.21, No.10 1986

〔6〕 Kim, W. and et al., Integrating an Object-Oriented Programming System with a Database System, in «Research Foundation in Object-Oriented and Semantic Database Systems», ed. by A. F. Córdenas and D. McLeod, Prentice-Hall, 1990

〔7〕 Bronnenberg, Win J.H.J. and et al., DOOM: A Decentralized Object-Oriented Machine, IEEE Micro, Vol.7 No.5, 1987

〔8〕 d'Arielli, L., A Message Management System for Personal Computers, IBM System Journal, Vol. 28, No.3, 1989