第4卷 第5期
1993年10月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 4, No. 5
Oct. 1993

面向对象系统软件的分层构造模型

章远阳　杨芙清　邵维忠

(北京大学计算机科学技术系,北京 100871)

THE HIERARCHICAL CONSTRUCTION MODEL OF OBJECT—ORIENTED SYSTEM SOFTWARE

Zhang Yuanyang, Yang Fuqing and Shao Weizhong

(*Department of Computer Science, Peking University, Beijing* 100871)

Abstract From the properties and limitation of traditional model in system software, we introduce the concept of persistent object memory, and then propose a hierarchical model of object—oriented software system to make its implementation universal and portable. At last, around this hierarchical model, we attempt to discuss the object—oriented system software in a uniform point.

摘要 本文从分析系统软件传统模型的特点和局限性出发,介绍了永久性对象存储的概念,在此基础上,从实现的通用性和可移植性考虑,讨论了面向对象系统软件的分层构造模型,最后,围绕着分层模型,我们试图以一种统一的观点,就面向对象系统软件各部分之间的功能分配及相互关系等方面,阐述了我们的有关认识.

§0. 引　言

80年代提出的面向对象(Object—Oriented)设计思想经过近年来的工作和探索,其特点和作用正日益为人们所接受,并且在一些应用软件系统(如VLSI/CAD,Hypertext等)的设计中得到了充分体现,随着面向对象技术及其应用的发展,人们意识到:目前的多数面向对象系统都是建立在传统的系统软件基础之上,由于两者之间在语义概念上的不匹配性,只好采用软件的方式来填补这段语义差距,并且这部分工作在每一个面向对象系统的开发中必须重复进行,这不但提高了系统开发成本,而且导致了面向对象系统内部的复杂性和效率问题. 系统软件如何从概念/本质上直接支持面向对象软件的开发,即如何构造一个直接提供对象语义的面向对象系统软件平台,成为将面向对象技术进一步推广应用所亟待解决的问题.

本文1991年1月23日收到,1991年6月12日定稿. 作者**章远阳**,1991年博士毕业于北京大学,现在深圳工作,主要研究领域为软件工程. **杨芙清**,教授,学部委员,系主任,主要研究领域为操作系统,软件工程,软件固化,软件工程环境. **邵维忠**,副教授,主要研究领域为操作系统,软件工程,人工智能.

在讨论面向对象系统模型前，我们先回顾一下传统模型.从静态看，传统模型是按功能划分的，即“数据库(DB)——语言(Lang)——操作系统(OS)”，实质上每部分在不同的层次上都提供了具有一定描述能力的存储平台及在其上的操作能力，只是侧重点不同而已；从动态看，传统模型可以概括成：具有一定生存期的进程在一维线性的虚存空间中活动，通过和二维树形文件空间的I/O界面来处理进程与外界的信息联系(如图1).

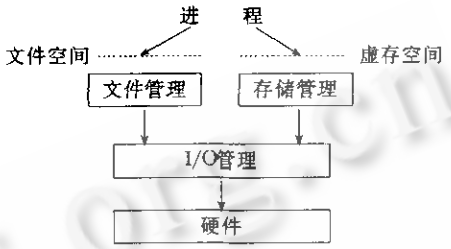

图1 传统模型

诚然，基于这种模式的OS(如UNIX)及其上的系统软件作为计算机传统软件技术的支柱，大大促进了计算技术的发展和应用，但也正是由于应用的广度、深度的发展，促使人们看到了其中的不足：

1.虚存是进程的运行空间，进程将其视为内部存储，对其上的信息(临时性信息)可以进行随机访问，利用指针构造复杂的数据结构等，但这些信息的生存期和进程相同，当进程终止(程序运行结束)后全部消失，即进程对虚存的操作能力强，但对其中的临时信息无法直接保留，必须通过文件的I/O界面进行处理.

2.所谓永久性(persistent)信息是指生存期超过进程，即程序运行结束后依然存在的信息，传统模型中对永久性信息的处理是利用文件系统，但进程对文件系统的接口是串行、顺序的，并且不能直接对其中的数据进行指针操作、进行数据类型的构造和抽象等，即进程对文件空间的直接操作能力较弱，但利用它可以将临时性信息转换成永久性信息保存起来.

3.这种将永久性信息和临时性信息的处理相互隔离的矛盾，常常使文件I/O部分成为整个软件系统的“瓶颈”，程序员要么将永久性信息看成是无类型的字符流，将其读入内存再进行语义解释；要么在文件系统上建立DB，按DB中的规则进行强制解释，这一矛盾随着应用领域的扩展(如AI，CAD等密集性数据处理(intensive data processing)中经常需要直接快速访问复杂的数据类型)将日益加剧，促使人们考虑能否将两者统一起来，使程序语言能透明、灵活地按访问临时信息的方式和能力，访问永久性对象.

将对象概念从应用软件扩展到系统软件直至底层，提供了解决这一问题的可能途径，这也正是研究永久性对象(persistent object)存储系统的背景所在.

下面我们先讨论永久性对象及其标识(identifier)，以永久性对象存储系统为基础，讨论面向对象系统软件的分层构造模型，并就对象概念的引入对系统软件及其框架的影响作初步的分析.

§1.永久性对象存储系统

和变量一样，对象的存储特性由作用域(作用空间)和生存期(存在时间)来刻划，其中对不同生存期的对象，语言中进行不同的显式说明：

1.生存期局限于函数计算内部，称为局部/自动类型.

2.生存期局限于进程活动(整个程序运行)内部，称为静态/外部类型.

3.生存期跨越进程，即进程终止后依然存在，并可供其他进程访问，称为永久性类型.

4.在系统瘫痪后，通过恢复/后援手段依然能存在，称为可恢复(resilient)类型.

其中,功能 1、2 目前常在语言中实现;能实现功能 3,即能处理永久性对象标识(PID)的语言称为永久性语言;而功能 4 则常在 DB 中实现. PID 具有很强的标识能力,表现在:

1. 永久性:程序语言借助 PID,能在任何时刻(和相应 PID 对象的处理进程是否终止无关),象访问临时性对象一样直接透明地访问永久性对象.

2. 透明性:凭借 PID,程序语言能在任何地点(无论相应 PID 对象驻留内存与否)都能找到并访问它.

3. 唯一性:PID 在对象生存期内,在整个系统范围内,是唯一的、不变的.

我们认为,PID 功能是实现永久性对象存储、永久性 OOPL、以及对象管理系统(OMS)、面向对象数据库(OODB)的关键和基础. 目前,处理 PID 的永久性对象存储,作为面向对象系统的基础已开始受到人们的重视,并试图用它来取代文件存储系统.

永久性对象存储系统[2],作为一个独立的实体,目前似乎还不明朗,鉴于目前的面向对象系统都建立在传统 OS 之上,因此,人们只好将永久性对象存储系统的功能分布零散地体现在面向对象系统的各个层次/部分中,如在 OOPL、OODB 中.

永久性对象存储的一个典型实现方式是采用双缓冲技术,[5]从存储空间抽象的角度看,即利用页缓冲将字空间转换成页空间,再利用对象缓冲将页空间转换成按 PID 访问的对象空间,这就是所谓双缓冲的作用.

§ 2. 面向对象系统软件的分层模型

提出面向对象系统软件分层构造模型的背景是:

1. 由于目前面向对象系统的技术还处于实验阶段,实现方式缺乏一致性,分层模型的提出有助于各系统之间的通讯、通用和移植,直至标准化.

2. 面向对象系统的核心部分——永久性对象存储在某些系统中多次重复,如将这一公共部分提取出来加以"沉淀",形成构造面向对象系统新的公共平台,取代传统的 OS 文件平台,那么不但能避免重复劳动,而且将简化面向对象系统内部的复杂性.

3. 功能影响结构,结构反映功能,传统模型难以从本质上支持面向对象系统,将对象语义引入到"基层"中,将形成一个新型的面向对象系统软件框架.

分层是人们认识世界、解决事物复杂性和多样性的一种方法,类似于网络的 ISO—OSI 模型,对于面向对象的系统软件,也可以构造出一个 7 层模型,如图 2 所示,各层之间的功能是单向依赖的,下层功能的实现对上层透明.

1. 第－2 层:物理存储层,是永久性对象存储系统中的物理存储器(内存和磁盘)及其寻址机构.

2. 第－1 层:虚拟存储层,是对第－2 层物理存储的虚拟抽象.

3. 第 0 层:永久性对象存储层,提供无类型的永久性存储平台,在其上按 PID 访问对象.

4. 第 1 层:基本类型层,在永久性对象存

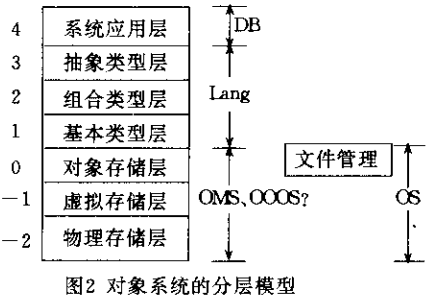

图2 对象系统的分层模型

储之上，提供一组对象的基本类型，如字符、整数等.

5. 第2层：组合类型层，提供一组类型构造规则，将基本类型构造成组合类型，使对象存储具有描述复杂结构类型的能力.

6. 第3层：抽象类型层，根据用户给出的语义定义和类型接口，解释抽象类型，使对象存储具有自行构造类型的能力.

7. 第4层：系统应用层，这里主要涉及面向对象数据库.[5]

我们以一个统一的观点来看系统软件，可以看出：第－1、－2层通常是传统OS的一部分，第1至3层的功能通常由程序设计语言(反映了逐步类型化的趋势)完成，而第4层则是我们通常所述的DBMS. 第0层反映了传统系统和面向对象系统的本质区别，因此，我们将重点放在第0层，即永久性对象存储层上.

第0层给上层提供的映象是：一个线性平坦的、按PID访问的永久性对象存储空间. 该层提供了对象位置的抽象，允许进程按PID直接、灵活、透明地访问对象，它将第－1层按地址访问的虚存空间改造成按PID访问的永久性对象存储空间，如果这层对上层的接口功能是标准通用的，则建立在此上的面向对象系统是可移植的.

在此，我们认为值得注意的是：

1. 第0层只提供了对象存储的内部模型，关于对象的进一步语义(如类、实例等)则由上层提供. 因此在第0层上提供的对象空间是一维线性的，随着其上各层对对象语义的"丰富"，相应对象和对象之间的关系也将"丰富"为由其语义数据模型所描述的网状关系. 可将上述分层模型看成是：以第0层为根(类似于永久性OOPL中的根类Object)，逐步向上生长，以构造/形成子类的倒置的类型树.

2. 各层之间要能相互移植通用，除了接口功能一致以外，数据的通讯格式也必须是一致的，相应地还应当有一组数据格式描述，在此从略.

3. 第0层对象数据格式的描述应当是自描述的，以便上层能进行更高层的语义解释.

§3. 若干问题的思考与认识

通过上述关于PID、永久性对象存储、面向对象系统分层模型的讨论，我们得出了一些初步的认识：

1. 关于文件存储和对象存储

如图3所示，面向对象系统和传统系统的分界线在第0层，即对象存储提供的是一维线性的永久性对象存储空间，支持一体化的面向对象系统构造；而文件存储提供的是二维树形的、按名存取的文件空间，其本质上支持传统软件的开发方式.

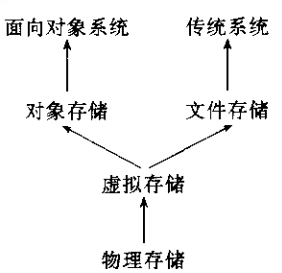

图3 对象存储系统和文件存储系统

目前，大多数面向对象系统是构造在文件存储之上的，这两者之间语义上的不协调性，通过系统中的对象存储部分解决，这使得对象→文件→虚存的转换形式效率不高，并且概念上也不够清晰.

在此，我们需要强调的是对象存储和文件存储并非截然对立的：一方面考虑到文件概念本身的特点及其软件可兼容性，将文件概念纳入到对象存储的统一框架之中是将两者相互

结合的较好途径，此时将文件也看成是对象，如 PCTE 中将文件系统看成是其 OMS 中的一部分特例；另一方面，永久性对象存储的实现和文件系统在某些部分上是相似的，因此，对文件存储和对象存储进行比较研究将有助于对文件系统以较少的改动，将其改造成一个永久性对象存储系统.

2. 永久性 OOPL

目前，有些永久性对象存储功能在永久性 OOPL 中实现，如 E[4]、PCLOS 等系统，将第 0 层纳入语言的范畴. 根据分层模型，永久性语言的编译代码只需调用标准的第 0 层接口功能，就可以使各种永久性语言在通用的第 0 层上移植，并有助于在各种语言中灵活方便地扩展永久性处理特性.

3. 面向对象 OS(OOOS)

对面向对象 OS 的理解有两层含义，表层含义是：采用 OOPL 设计/改写 OS，但其向用户提供的接口依然是传统(类 UNIX)的如 Mach、Choices 等系统，这样可以使 OS 内部概念设计更加清晰，易于理解，并降低了开发成本；深层含义是：向用户提供对象概念的接口，即永久性对象存储空间，以期从本质上，从对象开发方式上支持上层面向对象软件系统的构造，我们认为：真正意义上的面向对象 OS 应当是采用第 0 层替代文件存储后所形成的软件平台，它同时也是 OMS 的核心. 可以看出，OODB 的核心在某种程度上已渗入了面向对象 OS 中，从这个角度来说，面向对象 OS 反映了新一代 OS 的主要特征：面向对象、以 DB 为核心.

4. 面向对象 DB(OODB)

目前主要从两个途径进行 OODB 的研究开发：其一是扩展关系数据库；其二是扩展 OOPL，在其中增加永久性、复杂对象、集合操作等能力，使其成为数据库程序设计语言. 后者正日益受到人们的重视，因为这样可以以永久性对象存储为媒介，将 OODB、OOPL、OOOS 相互渗透融合，自然地体现了当代软件环境的集成思想.[1]

5. 虚拟存储是计算技术发展史上的一大里程碑，它从逻辑存储上统一了内存和外存，但在程序设计概念上两者之间的相互隔裂依然存在. 永久性对象存储是在虚拟存储的基础上，进一步扩展了永久性对象概念，这一扩展从操作能力上统一了进程对永久性和临时性信息的处理，并以对象为媒介，在更深一层的意义上进一步统一了内存和外存. 我们认为，从面向对象系统发展的长远眼光看，这一意义不亚于虚拟存储.

6. 我们认为：对象概念已对系统软件的组成框架产生了深刻的影响，以永久性对象存储为核心，系统软件各个部分达到了概念上的集成. 对于系统软件的功能分层，“分久必合”，随着技术的发展，这种“合”必将由功能级走向概念级.

参考文献

1　杨芙清等，软件工程支撑环境集成化问题研究，计算机科学，1987，4.

2　P. Balich et al.，Layered Implementations of Persistent Object Stores，Software Engineering Journal，Mar. 1989.

3　Balzer，R.，Living in the Next－Generation Operating System，IEEE Software，Nov. 1987，77－85.

4　Joel E. Richardson，Michael J. Carey，Persistence in the E Language：Issue and Implementation，Software Practice & Experience，Vol. 19，No. 12，Dec. 1989.

5　Won Kim et al.，Integrating an Object－Oriented Programming System with a Database System，in OOPSLA'88 Proceedings，142－151.