实时 操作系统 面向对象

计算机科学1993 Vol.20 No.5

62-66

基于对象的实时操作系统

TP316

朱晓明 吴国仕

（北京航空航天大学706教研室 北京100083）

摘 要

本文在进行了实时系统的需求分析之后，提出了实时系统的生成思想，即以一个基于对象的实时操作系统（OBRTOS）内核作为支持，以"对象"作为生成实时系统构件的基础，在生成一个完整的实时系统开发环境的同时，生成一系列实时系统。本文亦简要介绍了在PC AT微机系统上对OBRTOS的实现及其特点。

面向对象技术是一种面向数据流的，并集模块化、数据抽象、信息隐蔽和消息传递等诸多优点于一身，既能适合于系统分析又能适合于程序设计的工程技术。它利用"对象"这个概念，将数据与在其上的操作封装在一起，直接作为可独立存在的建模实体，通过"消息传递"来完成对象之间的联系，若干个对象通过消息传递可以有机地组成一个系统。当问题分解为一些对象以及各对象间进行组合和联系时，易于为人们所理解和接受，易于实现。对象对外表现出一种或若干种功能，但采用"信息隐蔽"技术隐藏功能实现的细节。对象之间具有低耦合度，对象内部具有高耦合度。这种技术设计出来的系统可靠性高，可维护性好，系统部件再用性强．系统结构灵活，裁剪方便，从而为系统系列化生产，提高软件生产率等提供了便利。特别值得一提的是面向对象技术使得对象具有并行性，尤其适合于在多处理机体系结构上运行。目前面向对象技术已得到广泛应用，支持面向对象的程序设计语言如Smalltalk、Borland C++等已有很多，且大部分是应用于办公事务或用户界面，而支持实时的面向对象操作系统尚不多见。

面向对象技术所具有的诸多特点，大部分均适合于面向数据流的实时系统的要求。特别是在实时问题分解时，对生成的实时系统提出的并行性、再用性、结构性等要求，利用面向对象技术可以得到很好的解决。对象本身，包括利用数据抽象和数据与操作封装，使得实时系统的系统分析与系统设计趋于简便。目前的面向对象程序设计语言都是非实时的，即在对象中没有符合实时要求的约束条件，且支持面向对象程序设计语言的操作系统也是非实时的。有两种方式可用来为应用实时系统服务：一是支持实时面向对象语言的操作系统方式，一是借助其它语言来支持对象的实时操作系统方式。本文是从后一种方式展开论述的。

一、实时操作系统的需求分析

随着计算机应用领域的扩大，操作系统的发展也从非实时的通用型操作系统扩展到实时的专用型操作系统。但其特点已与通用机的操作系统有所区别，已不再是以扩大系统用户数量、提高系统资源利用率作为重点，而是根据不同的实时要求，完成应用软

and Distributed Computing 1991

〔5〕张可军、杨桃栏，向量块中的指令调度，电子学报，1990，6

〔6〕张可军等，并行编译对控制语句的几种优化技术，待发表。

〔7〕Philip J, Hatcher等，"A Production-Quality C* Compiler for Hypercube Multicomputers, 1991 ACM

〔8〕M. Annaratone等，"The K2 Distributed Memory Parallel Processor: Architecture, Compiler, and OS", 1991 Supercomputing

件的实时调度和系统资源的实时分配，以解决过去实时系统不够灵活、CPU利用率低的状况，在一定范围内，提供系统的通用性。更多的情况是牺牲系统资源利用率，牺牲系统通用性，以空间换取时间，取得实时的效果。由于从实时和应用角度的考虑，要求实时操作系统应能提供一个短小精悍的实时内核和系统程序库，便于配合用户的应用软件构成实时系统；要求从结构上保证实时系统的高可靠性和可扩展性；同时也要求实时操作系统应为系统软件和应用软件提供软件标准，提高软件特别是应用软件的可再用性，为实时系统系列化生产或提高实时系统软件生产率奠定基础。

作为实时操作系统，其首先应具有如下几方面的处理能力：

①异步事件响应：实时操作系统应能对异步产生的各种外部事件实时提供异步处理和中断服务，并应保证不丢失异步产生的每一个外部事件。

②实时调度策略：实时操作系统应能按照一定的调度策略，使得每一事件都能按照"实时"的轻重缓急程度得到处理，并能时刻保证当前最急迫的事件通过抢占CPU得到处理，而使当前正被处理的事件暂缓处理。当抢占事件处理完毕后，被抢占事件从中断点自动恢复执行。

③同步机制：实时操作系统应能提供事件之间同步执行和共享数据协调使用的协议，以解决由于事件并发处理所带来的同步、互斥、通信以及资源共享与保护等问题。

④异常管理：实时操作系统应具有对异常事件实时处理的能力，将其对其它事件的影响减少到最小，以保证系统的正常运行，同时允许用户设置自己的异常处理程序。

⑤开放性：实时操作系统应具有允许用户开发自己的应用系统的能力。提供编程、链接及调度各种程序的能力，并通过对实时操作系统的裁剪，增加用户的应用软件，配置并生成用户自己的应用系统。

二、OBRTOS

基于对象是指支持对象作为系统特征。它与面向对象不同之处在于它只支持对象的部分而不是全部特征。在本文是指支持对象的抽象性和封装性，而不支持其继承性和多态性，这主要是基于实时的原因。西北工业大学的周兴社等提出了实时对象和时间封装的概念，通过扩展传统对象机制使其具有实时计算语义，以维护对象特征的完整性，达到充分利用对象继承性的目的。但实时应用软件由于其与硬件结合的紧密性而导致千变万化，并在大部分应用中具有针对性，实时周期也不相同，所以这种由于时间封装带来的继承特点将对实时产生严重的影响。

以对象作为实时操作系统的基本运行实体，可以满足对实时系统的要求，因而如何实现支持对象及消息的机制成为OBRTOS的关键。基于对象的方式使得在实现支持对象方面得到简化，只需向应用程序设计人员提供OBRTOS对象的设计标准和OBRTOS支持的系统元语句，即可得到所需的应用对象。这些应用对象在装入OBRTOS后，通过消息传递与其它对象一起构成有机整体——实时系统。消息及消息传递机制是由OBRTOS的内核来支持的。内核以对象为基础，根据对内部机制的综合考虑和调整，实现了对消息机制的支持。

1. 对象与消息

对象是OBRTOS的基本运行实体，是OBRTOS调度的基本单元。消息的外在表现形式为对象的消息模式。用户利用OBRTOS提供的元语句，根据已知对象的消息模式，向已知对象发送消息。元语句则将消息送到对象的信箱，并激活该对象进入就绪队列。当轮到某一对象利用CPU时，OBRTOS根据信箱中的消息和该对象的消息模式，解释并调度对象中相应的方法执行。整个系统的消息流程即为实时系统的控制流程。这种控制流程的安排是由用户根据自己的应用需要而事先安排好的，因而编制对象时就确定了消

息流程。对象的基本结构如下：

```
OBJECT {
  输入数据类型；/* 接收来自外部的数据 */
  对象的属性；/* 对象的内部状态 */
  对象的约束条件；
    /* 对象中相应方法的执行前提 */
  消息模式；/*对象与外部的唯一接口*/
  方法；/* 对象的操作 */
}
```

宏观上，是外部通过消息模式通知对象执行某种操作，对象则在判断方法执行的约束条件得以满足的前提下，启动方法根据对象的属性去处理数据，最后通过消息向外部发送处理结果，在约束条件没有得到满足时该对象自动挂起。对象的方法在执行过程中，可以向外部对象发送消息，但这种发送是异步的，对象可以在不必等待返回结果的情况下继续执行。

对象在系统中有如下四种状态，并为其中的准备、挂起、睡眠三个状态建立三个系统队列。

*运行状态：占有CPU的当前对象状态；

*准备(就绪)状态：准备就绪，等待CPU资源的对象状态；

*挂起状态：对象在约束条件没有满足的情况下，在系统中的等待状态；

*睡眠状态：对象处于延迟指定时间的状态；

对象在系统中有如下两种状态：

*创建状态：对象初始化状态；每个对象必须有一个初始化方法，由系统在创建对象时调用。系统可以动态创建对象，对象在初始化完毕后，即进入就绪状态。

*删除状态：对象退出系统时的状态。除非指定，每个对象不一定需要一个删除的方法。系统可以为对象指定删除方法，系统也可以动态删除对象。

由于在实时系统中，特别是过程控制的系统中，绝大部分功能模块是局部确定的，且为经常使用的，故在存贮空间保证的情况下，对象在执行完毕后，除非指定删除，其总是直接进入挂起状态，等待下一次激活。系统在一般情况下，总是在系统初始化时，静态创建所有对象，以免在系统动态执行时，增加额外的系统开销。

对象中操作都是对数据或状态进行顺序处理，在处理中间除被高优先级任务抢占CPU外，不应在代码中间位置停止，总是直到该操作执行到某一出口。每个操作总是只有一个入口，一个出口。

2．OBRTOS的系统调度策略

在实时操作系统中，根据实时程度及所选硬件环境的不同，有很多种调度策略可供选用。OBRTOS在不考虑应用操作系统所带来的开销的情况下，选择基于优先级的抢占调度策略作为系统调度策略，以便在充分享用操作系统所带来的便利的前提下，获得最大程度的实时性。这种调度策略是在OBRTOS运行之前，为OBRTOS中的每一个对象静态指定一优先级，然后启动OBRTOS执行。调度程序根据基于优先级抢占的策略，总是将准备状态队列中具有最高优先级的对象取出，并根据系统状态判别该抢占对象与当前占有CPU的当前对象，谁最终获得CPU的占有权：若当前对象的优先级高于或等于抢占对象的优先级时，当前对象继续运行，抢占对象返回到准备队列；若当前对象的优先级低于抢占对象时，抢占对象抢占到CPU，当前对象排到准备队列中同一优先级的对象队列尾部，等到下一次轮到时的CPU抢占。其它情况下，当前对象由于处理完所有消息或由于其它原因如睡眠等被挂起而自动放弃CPU的控制权时，调度程序则从准备队列中选择具有最高优先级的对象投入运行。

激活系统调度的情况有如下几种：①中断事件发生后，事件处理程序通过发送消息激活一个中断对象；②当前对象发送消息；③当前对象挂起；④当前对象处理完成所有消息。

3．内部通讯和数据交换

消息传递是OBRTOS内部通讯的唯一机

制，是实时系统中同步机制的一种实现方式。这样的方式硬性规定了在对象之间建立唯一的狭窄接口，同时保证了通常对数据进行交换时所必备的互斥手段，解决了对某一共享资源的竞争问题，满足了实时系统对同步机制的要求。每当发送一个消息时，总是激活一个接收对象；若该接收对象已在就绪队列中，则所接收的消息按发送对象的优先级排入接收对象的消息队列中。若该对象不在就绪队列中，则该对象按规则抢占CPU，其优先级高于当前对象(即发送对象)则其抢占成功，否则排入就绪队列中同一优先级对象队列的尾部。若发送对象和接收对象分布在不同处理机中，则系统以中断或查询方式激活接收对象，发送对象和接收对象的执行无论在空间还是在时间上都得以同步；而在同一处理机内，则从宏观上讲也具有时间和空间上的同步概念。

数据交换对于面向数据流的实时系统而言是必不可少的，并且它与传统的面向对象系统的数据交换不同之处在于，它的数据交换往往不是一个、两个，而是成批交换，因而常常存在于对象之间的数据交换是一个存有无序数据的存贮区在对象之间传递。正如前面对对象基本结构的描述，接收对象通常是按照它自己对输入数据类型的定义及相应的内部属性，解释和处理输入数据，同时它也可以将处理结果存在一个存贮区中，通过消息将该存贮区传递出去。这与对象的封装性并不矛盾。首先对象的封装在于对对象内部属性的封装；其次，当对象接收到一个存贮区后，它在处理该存贮区的数据时，始终拥有该数据区的控制权，直到所有数据处理完毕，它或者释放该数据区，或者通过发送消息交出该存贮区的控制权。

4．中断机制

中断机制是实时操作系统为及时响应突发性随机事件而必须具备的基本机制之一。OBRTOS为中断机制提供两类特殊对象：事件对象和中断对象。

事件对象只有一个方法（即事件处理程序），且事件对象是由对应的中断信号激活的。对事件对象中方法的约束条件是根据事件的处理时间确定的。一般而言，事件对象应尽可能快地完成简单的事件处理，并根据需要向外界发送消息。事件对象在发送消息或不发送消息而直接退出中断处理之前，将屏蔽系统中所有可能发生的中断。事件对象不受优先级控制，随时激活，随时占用CPU，并根据需要借用被中断对象的堆栈段。当然在不过份增加系统开销的情况下，事件对象也可以拥有自己的数据段。

中断对象是由事件对象所发送的消息激活的对象，它完成事件处理的复杂功能。中断对象具有高于一般对象的优先级，以便能抢占CPU，得到及时处理。中断对象在执行过程中与一般对象一样，由OBRTOS调度程序管理与调度，并同样可以被更高优先级的对象抢占CPU。中断对象的结构也等同于一般对象，只是其激活状态与一般对象不同，由事件对象激活对象的激活状态是系统属性，由OBRTOS管理。

采用两种特殊类型的对象，是为了避免在中断事件处理过程中屏蔽掉可能发生的高优先级中断事件。事件对象是为了处理短时间的中断服务，而长时间的中断服务则通过事件对象激活中断对象来处理。

5．异常管理

异常情况是表示在对象执行过程中出现异常而导致对象执行失败。这种失败的原因通常有两种：程序员设计错误和环境出错。程序员设计错误可以通过逐步调试而得到解决；环境出错则是由于无法知道的环境错误而引起的。OBRTOS为可能出现的异常提供尽可能详细的异常处理服务。同样OBRTOS也提供了一种特殊类型的对象——异常对象。当一般对象在执行过程中出现异常时，OBRTOS就激活对应的异常对象。异常对象具有相当高的优先级，并在执行过程中屏蔽所有中断。故在出现异常时对应的异

常对象立即抢占CPU。在异常处理服务完毕后，根据需要，或返回到当前的出错对象，或返回到OBRTOS。一般情况下，对于返回到系统的情况，异常对象应保留出错环境的相应数据，供系统调试使用。

四、系统裁剪与应用软件再用

由于OBRTOS采取了基于对象的设计方法，所以所有与OBRTOS内核结合在一起并构成成功的实时系统的应用软件，都可以以系统构件库的形式升级成系统对象。对象所具有的内耦合度高、外耦合度低的特点，使得对象的增加、删除与修改变得极为容易，且对外界影响甚少，从而使实时系统的生成极为容易。在生成实时系统时，可以由下列方式得到：

系统内核＋裁剪过的系统构件库＋用户应用软件→实时系统

软件再用问题是为了提高软件生产率、保证系统可靠性而引起计算机界关注的主要问题之一。单纯利用标准件库是比较困难的，一是标准件难于生产；二是应用针对方向不同导致处理标准不同。面向对象技术对产生可再用软件的设计和实现提供了极大的帮助。面向对象的抽象机制、封装性等特点保证了对象内部实现与外部表现的相对独立，从而使对象成为一个可再用软件的构件。这种构件使得整个系统的构成极为灵活，整个系统的可靠性可以由构件的可靠性来保证。同时用户也可以对已有构件作针对性处理，通过检验而形成新的构件。

五、OBRTOS的实现

OBRTOS的硬件运行环境是IBM PC/AT及其兼容机，CPU为Intel 80286及其以上的微处理器，并要求能在保护方式下工作。目前，已在80286微机上实现了OBRTOS的内核，并配有简单的人机接口和构件库，而文件系统则是直接借用DOS的文件系统。对象则是在SMALL或COMPACT模式下用C或汇编语言编写的可执行模块。在OBRTOS中运行的对象是通过内核提供的简单人机接口，或是逐个，或是成批装入系统的。一般而言，在开发环境下可以采用成熟的对象模块成批装入、新开发的模块逐个装入的方法，对新模块加以调试。OBRTOS的启动可在DOS的命令提示下启动(系统的开发环境)，也可写到ROM中启动(系统的应用环境)。

保护方式为OBRTOS的实现提供了两点硬件支持：一是任务切换；一是大的存贮空间。在8MHZ的80286中任务切换的时间大约为23μs(25MHZ的80486DX则为17μs)，其主要完成功能是用一条指令或通过中断实现将当前任务的寄存器状态保存到指定的存储区域中，并更换为与当前任务无关的任务的相应寄存器状态。从某种意义上讲，这里所指的80286任务即为OBRTOS中的对象概念，但有相当大的区别。这样再加上内核在对象调度上所花费的极小系统开销，足可以满足硬实时系统的强实时要求。

存贮空间通常是实时系统在通用微机上实现时较难解决的问题之一。传统DOS下的604KB RAM空间使代码空间和数据空间受到限制，且两者为争夺系统存贮资源而极易发生冲突。一般情况下，在代码空间必须保证时，总是减少数据空间，造成常用数据不必要的重复计算，增加了时间开销。使用大存贮空间的另一好处是可以避免因空间的不够而不断从外存(硬盘等)上交换代码和数据，增加额外的系统时间开销。同时由于空间问题的解决，可导致对实时问题多种算法的选择，特别是对以“空间换取时间”的算法的选择，从而加快实时问题的处理速度。在保护方式下，286微机系统可以提供16MB的存贮访问空间，386及其以上微机系统可以提供4GB的存贮访问空间。在硬件价格大幅度下降的今天，以“空间换取时间”已从希望变为现实。

可以预言，OBRTOS具有极强的生命力和广阔的发展前景，但其主要障碍是如何设计出良好的对象接口，以便程序员能很快地应用系统对象和设计出自己的应用对象，并把应用对象加到库中。(参考文献略)