信息技术和电气工程学科国际知名教材 中译本 系列

实时系统
Real-time Systems

C. M. Krishna
Kang G. Shin 著

戴琼海 译

清华大学出版社

信息技术和电气工程学科国际知名教材 中译本 系列

实时系统
Real-time Systems

C. M. Krishna
Kang G. Shin 著

戴琼海 译

清华大学出版社
北京

C. M. Krishna　Kang G. Shin
Real-time Systems
EISBN: 0-07-057043-4

北京市版权局著作权合同登记号　图字: 01-2003-7832

图书在版编目(CIP)数据

实时系统/克里希纳(Krishna, C. M.), 星(Shin, Kang G.)著; 戴琼海译. —北京: 清华大学出版社, 2004.9
书名原文: Real-time Systems
(信息技术和电气工程学科国际知名教材中译本系列)
ISBN 7-302-08808-X

Ⅰ.实…　Ⅱ.①克…　②星…　③戴…　Ⅲ.实时操作系统—教材　Ⅳ.TP316.2

中国版本图书馆 CIP 数据核字(2004)第 055499 号

出 版 者: 清华大学出版社　　地　　址: 北京清华大学学研大厦
http://www.tup.com.cn　　邮　　编: 100084
社 总 机: (010)62770175　　客户服务: (010)62776969
组稿编辑: 王一玲
文稿编辑: 赵从棉
印 刷 者: 清华大学印刷厂
装 订 者: 三河市新茂装订有限公司
发 行 者: 新华书店总店北京发行所
开　　本: 175×245　印张: 25.75　字数: 523 千字
版　　次: 2004 年 9 月第 1 版　2004 年 9 月第1 次印刷
书　　号: ISBN 7-302-08808-X/TP·6247
印　　数: 1~3000
定　　价: 39.00 元

关于作者

C. M. Krishna 自 1984 年至今任教于马萨诸塞大学(University of Massachusetts)。他发表了关于分布式处理、实时系统和容错领域的论著,为 IEEE Computer Society 出版社编辑了两卷读物,是 IEEE 计算机与 IEEE 实时系统会议的特邀嘉宾和编辑。Krishna 教授现在的研究领域为实时系统的可靠性与性能建模、容错同步、分布式实时操作系统及实时网络。

K. G. Shin 是密歇根大学(University of Michigan, Ann Arbor)电子工程与计算机科学系的实时计算实验室的教授和主任。他在分布式实时计算与控制、容错计算、计算机体系、机器人技术与自动化及智能制造领域以作者和合著作者身份撰写了 360 多篇技术论文(其中约 150 多篇发表在期刊上)和诸多书的章节。Shin 教授是 IEEE 会员,曾经是 1986 年 IEEE 实时系统研讨会(RTSS)执行主席,1987 年 RTSS 会议的主席,IEEE Transactions on Computers 实时系统杂志 1987 年八月专刊的客座主编,IEEE Transactions on Parallel and Distributed System 1991～1995 期间的编辑。

前言

实时系统在过去的几年里迅速成长,今天,计算机技术几乎被运用于任何领域,从烤面包器到汽车到线控(fly-by-wire)飞行器。在这样的嵌入式系统中计算载荷变化很广,从每秒钟做一些数学计算的机器到高速复杂运算的计算机。计算机故障带来的后果也很广,从烤焦面包片到飞行器失事或者化工厂爆炸导致死亡。

本书的目标是向读者介绍实时系统中的设计和评价的问题。我们涉及了广泛领域的论题,在一些地方对硬件和软件问题做了详细阐述。这本书读者对象为应用工程师、研究生和高年级本科生。

本书的一部分使用数学讨论，只要可能我们尽量将数学分析从文字中分离出来，这样使得本书能适应不同阅读层次的读者。较高深的章节用 * 标记，这些部分需要更多的毅力和能力来理解，可以跳过它们不看。然而我们希望读者尽量不要跳过数学分析部分，大多数时候我们是因为害怕和消极对待这些数学符号而跳过它们的，但是，对这些问题的真正理解不能不基于对这些数学分析的理解。

本书包含的内容超出了一个学期的教学量，授课教师可以选择一些重点章节，比如任务分配和调度，或者容错，并且可以根据实时系统工程师的兴趣有选择地展示不同主题的广泛的素材。为了达到这两个目标我们尽可能地使每个章节相互独立，并且这也允许本书能作为一本应用工程师的参考手册。印刷或者其他错误请通知作者：

rtbook@tikva. ecs. umass. edu

我们计划在万维网上发布和维护本书的勘误表：

http://www. ecs. umass. edu/ece/gradfac/krishna. html

致谢

很多人为本书的出版作出了贡献。我们要感谢 McGraw-Hill 的 Eric Munson 的委任和对本书推迟一年出版的宽容。

一些同事和学生阅读了此书的一些章节或者全部，提供了宝贵意见或指正。他们的姓名如下（无先后次序）：

Y. -H. Lee	C. Ravishankar	N. Soparkar
A. Ansari	J. Rexford	S. H. Son
N. Suri	J. Strosnider	F. Zhou
A. Mehra	W. Feng	A. Shaikh
T. Abdelzaher	A. Indiresan	E. Atkins
P. Ramanathan	S. Daniel	T. Koprowski
S. Wilson	K. Ramamritham	W. Preska

密歇根大学的 Beverly Monaghan 和马萨诸塞大学的 June Daehler 为本书提供了有价值的秘书助理。ETP Harrison 的 Julie F. Nemer 是本书的技术编辑，她的注释使本书更具可读性。还要感谢 Michael J. Kolibaba 在我们和技术编辑之间的协调工作，同时还感谢如下评论者：Wei Zhao，In-Sup Lee，William Marcy 和 Borko Furht。

C. M. Krishna
K. G. Shin

目 录

第1章

绪论

在写了一本关于实时系统的书后，你可能会认为我们能够给出关于实时系统的准确恰当的描述。不幸的是，我们不能。如果作为一个定义来提出，我们可以像下面这样说：

任何一个对于外界的刺激都能由计算机给出及时响应的系统就是一个实时系统。

上面的叙述是正确的，但那仅仅是因为它几乎没有任何内容。如果稍微思考一下，你会发现此定义引出的问题和它所回答的一样多。如果你决定对上面的定义进行深入的剖析，你可能会和我们有下面的对话：

你："及时"是什么意思？

我们：它意味着一个实时系统运行具有时间限的任务。

你："时间限"的意思是任务到那时必须被完成吗？

我们：不完全是。有时是的：如果你正在利用计算机来控制飞行器，并且你在飞行器着陆时错过了一系列的时间限，那么你将有机毁人亡的危险。有时则不是：如果你正在玩一个视频游戏，并且响应比预计的时间稍长一些，那么不会发生什么糟糕的事情。

你：任务"完成"的具体含义是什么？一个任务"完成"了和没有"完成"有明显的区别吗？

我们：这是不确定的。当你去银行支取一百万美元时，在支款之前需要计算出总金额，这时任务完成与否是有明显区别的。有时则没有明显区别：如果你的应用需要计算 π 值，它可以决定早点结束，接受一个低精度的值，也可以继续计算，以得到越来越精确的估计。

你：你们将如何处理错过时间限的实时任务？是丢掉呢，还是无论如何都完成它？

我们：这要依情况而定。如果你坐在因错过一系列时间限而坠毁的飞机上，你和计算机都不必考虑了。在另一种情况下，如果你有一个视频会议的应用，它在处理声音包的时候发生了一个微小的延迟，你可以决定不丢掉这个包。不管属于

哪一种情况，在时间限错过之后，任务值都会下降到一个确定的水平。在某些情况下，会直接下降到零；在另外一些情况下，会渐进地下降。图 1.1 给出了一些例子。

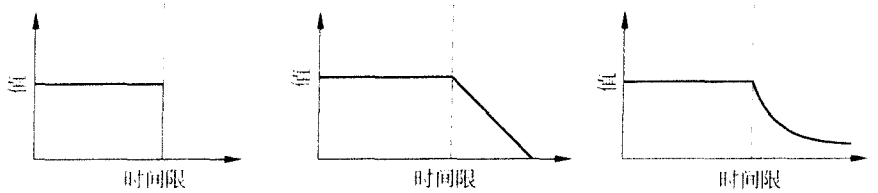

图 1.1 任务值函数的例子

你：根据你们的定义，每个计算机都是一个实时的计算机吗？

我们：不幸的是，是的。如果你按步就班地阅读我们的定义，就会发现一般意义上的工作站或者个人计算机也是一个实时系统。因为如果你敲击键盘，计算机要一个小时后才响应，把字母显示到屏幕上，你可能会很不高兴。我们要在一个有限的时间内得到结果，在这个意义上，一切都是实时的。所以，我们的定义包含了一切计算机，因此这一定义是毫无价值的。

你：你们想改变对实时系统的定义吗？

我们：是的。新的定义更加模糊，更少的限定，不十分明确，但是它的可贵之处是我们可以结束现在的讨论。

一个实时系统就是所有我们（本书的作者）认为是实时系统的系统，这包括一些嵌入式系统，控制像飞行器、核反应堆、化学电力工厂、喷气式发动机和一些其他的对象。在这些应用中，如果计算机不能及时给出它的输出，会发生一些很糟糕的事情，这些称作强实时系统。另一类称为软实时系统（比如多媒体系统），即使错过了某些时间限也不会有什么灾难发生，但是性能会下降到一般可以接受的水平以下。一般来讲，一个实时系统要尽最大努力确保满足任务的时间限。

1.1 汽车-司机的例子

为了理解实时计算中的主要问题，现在，我们考虑一个熟悉的问题：人驾驶汽车。在这个活动中，司机是实时控制者，汽车是被控制的进程，其他的汽车和路况构成了操作环境。通过了解司机所需要的东西，我们能够理解实时计算中的一些主要问题。

汽车的主要执行部件是车轮、发动机和刹车。控制部件是加速器、方向盘、刹车板和一些其他部件（例如车灯、收音机、擦净器等的开关）。

司机必须满足的限定条件是什么呢？他必须安全抵达目的地，途中不能和其他车辆或固定的物体相撞，还要把速度保持在限定范围之内。现在让我们把这些变换成实时计算中的术语。

司机可以说成将要完成一项任务：在满足上述条件的前提下到达目的地。我们应该如何评价司机的表现呢？我们评价司机的行为成果，还要考虑相应的操作环境。一个明显的成果是到达目的地，第二个指标就是所用的时间。然而，我们对于花费时间的评价必须被路况条件加权：一个司机如果在暴风雪的天气里保持平均每小时15英里的速度，他表现得很好（安全抵达目的地）；如果天气晴朗，路况良好，同样是每小时15英里，却意味着失败。

假定司机未能抵达目的地，毫发无损地停在了路边的壕沟里。同样，我们对司机表现的评价不能仅仅局限在结果上——他停在了壕沟里，还要考虑致使他停在那里的原因。如果他是为了避免由于某个人开错了车道而造成碰撞，被迫停在了那里，我们认为他的表现是成功的；如果他是由于急速转弯、车轮打滑，而掉到了壕沟里，我们认为这是失败的。

关键在于性能不是绝对的东西，但是性能必须在条件允许的情况下评价。换句话说，性能评价是成果与该条件下最大可能的成果的比较。这是很重要的一点，我们会在第2章继续讨论。

现在来考虑司机必须完成的任务。它们中的一些是任务成功的关键，另一些则不是。操作方向盘和刹车是关键任务，收音机调频则不是。转弯与刹车是依赖操作环境而具有不同时间限的实时任务。星期日早晨六点在安静的街道上开车的时间限显然与交通高峰期的高速公路上不同。实时系统中的任务时间限不是常数，它们是随着操作环境的不同有所变化的。

根据司机身体条件的什么信息来预测其表现呢？即使知道他在旅途中99.99%的时间里都是清醒的也是不够的。如果他在一次行程中睡着了一两秒钟，可能会导致灾难性的后果。另一方面来讲，如果他一路上有许多微小的瞌睡，每次都不超过半秒，他不失败的机会还是很高的。简单地采用均值来预测实时控制器的性能是没有用的。

即使司机安全抵达目的地，他的微小瞌睡很可能会导致突然的刹车和加速，这会增加燃料的消耗。因此即使成功地完成了任务，他的微小瞌睡也不是没有代价的，他的成功很可能在别的因素上与完全成功的任务有所区别，如燃料消耗或者时间花费。

假如我们试图准确地描述司机的责任。任务的综合目标是清晰的：在没有违反任何交通规则的情况下安全抵达目的地。但是细节怎么样呢？假定我们想把每种可能的情况下司机必须做的工作都准确地描述出来，我们怎样才能写下这样的描述以确保该描述是完备的呢？作为一个练习，读者可以试着用平实的语言写出一个描述的集合。几分钟的书写工作就足以说明：要想写出一个完备确切的描述，即使是驾驶汽车这样易于理解的任务，也是极度困难的。写出正式的描述并且使它们有效可能是实时系统中最困难的任务，它们也是研究者不太了解的东西。

在这个背景下，我们现在列出实时计算中的一些主要问题。

1.2 实时计算中的问题

实时计算机一定要比单独的硬件和软件组件更加可靠。它应该可以在恶劣的环境下工作，如充满电磁噪声和基本粒子辐射和面临快速变换的计算负载。

实时计算领域有很多值得探索的问题，因为所有计算机体系结构、容错计算及操作系统中的问题同样是实时计算中的问题，而且此领域还有一个附加的复杂性，即必须满足实时的约束。

例如，制作任务调度计划。在一般意义的系统中，制作任务调度计划的目的在于公平，即所有的用户合理地共享计算机资源。通常使用循环调度来达到这个目的。每一个进程都与一个时间片有关。计算机处理一个进程直到下列情况之一发生时停止：进程完成；当一些存取工作或中断发生时被迫停止；或者该进程分配的时间片过期了。这时计算机切换内容处理其他进程。在这个基本思想下，可能会有一些变化，例如，时间片可以根据该项任务已经花费的执行时间而有所变化。

循环调度确保一个用户不会得到不均衡的计算机共享资源。在实时应用中，这个方法并不能很好地实施。为什么呢？我们来看示例 1.1。

例 1.1 考虑由计算机控制飞机的情况。在它的任务中有保持稳定和控制机舱温度在可接受的限制内等。假定飞机发生了扰动致使它有瞬间的不稳定，计算机这时应该调整控制面使其重新稳定。如果我们对这个应用使用循环调度，计算机可能中途通过控制调整更换内容，目的是花费时间确保机舱温度正常。结果可能会发生碰撞，这样，当飞机从空中坠落时，使机舱保持最佳温度对乘客来说已不重要。我们想做的是给稳定性维持的任务一个很高的优先级，这样就能确保当稳定性受到威胁时，所有其他的任务为此让路，允许这种至关重要的任务有足够的计算机时钟周期。

我们来看第二个例子，在这个例子中，可以证明由于实时的时间限的不同，一个设计问题的解也有所不同。缓存是计算机中的常用设备，它能减少有效读取内存时间和增加平均吞吐量。有两种方法分配缓存空间。一种是允许当前正在处理的进程使用整个缓存空间。这样就保证丢失率低，且因此有效内存访问时间很低，这是一般系统所采用的一种方法。然而，从一个实时系统工程师的观点来看，这个方法的缺点是任务在运行时缺少一定的预测。

例 1.2 假定进程 A 在不被其他进程打断时需要 t_1 秒的时间处理。考虑如果 A 被另一个进程 B 所抢占，在 B 结束后，A 从被打断的地方继续运行的情况。如图 1.2 所示。$t_{2.1}+t_{2.2}=t_1$ 吗？不一定！这是因为 B 在执行时可能为了扩大空间存放自己的工作集而从缓存里置换了 A 的一些代码行。当 A 要继续执行和存取这些被取消的代码行之一时，从缓存区内得不到结果，要重新到内存中去读取，这就使存取时间大于没有 B 出现独自运行时的情况。结果是 $t_{2.1}+t_{2.2}$ 很可能会大于

t_1。A 的执行时间将与(许多其他事情之一)A 被抢占的次数和每次被抢占时缓冲区发生的情况有关。

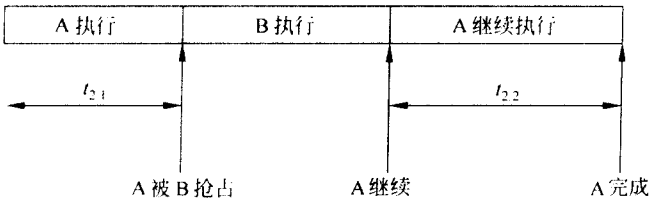

图 1.2 进程 A 被进程 B 抢占

任务的执行时间能够被预测是很重要的,它使设计者能够计算出是否所有关键的任务都能满足它们的时间限。如果缓存的影响是使任务的运行时间非常的不可预测,那么从设计者的角度来讲,它的坏处要大于它所带来的好处。

实时计算机系统在两个重要方面不同于通用的计算机系统。首先,对于它们的应用来说,它们的专用性很强;其次,它们失败的后果是极为严重的。

实时计算机是为专门的应用而设计的,例如,用来控制一架特定的飞机。这样的优点在于比起一般目的的机器,此应用的特点和其操作环境信息可以更准确地被测知。结果是,有可能准确地对实时系统进行细微调整以得到最优的性能。

既然实时系统中失败的后果比通用计算部件严重得多,所以这种系统需要更为仔细的描述,并且它的性能需要更好的评估,即,我们需要能够表达实时需要的描述语言和性能评价。除了其他方法,我们还需要一些方法来预测程序的执行时间;为软、硬件的稳定性建模;为处理器分配任务,并且调度它们使时间限得以满足;开发相应机制,这样当个别的组件出现差错时,系统能很快地恢复。我们应该研究这些和其他问题的细节。

例 1.3 对比一个工程师设计控制喷气式发动机的计算机和某人设计通用的工作站的情况。发动机控制的设计者清楚地知道系统必须处理的工作负荷。例如,他大概知道计算机每秒钟必须调整多少次机器设置,同时能粗略地计算出每次设置需要多长时间。与之相对应的是,工作站的设计者知道自己的设计大概的应用领域(例如,机器将被主要用来处理图形或多媒体),但是他对这个机器将要运行什么软件没有具体的想法。即使他能够预知他的机器将要运行的每一份复制,那也不见得对他有什么好处——调整设计来满足一种类型用户的同时可能会对另一种类型的用户产生极坏的影响。这正强调了我们所说的实时设计者(特别是严格的实时设计者)比一般意义上的工作站设计者对于其产品的操作环境要掌握更多的信息。

然而,实时的工程师比其他的一般意义上的工程师有满足性能目标的更重的负担。如果一个变周期喷气式发动机不能按时得到其控制输入,可能会爆炸。对于一个给定应用,如果工作站比计划慢了几毫秒,不会有什么严重的后果。

1.3 实时系统的结构

图 1.3 显示了一个实时系统控制某个进程的原理框图。被控制的进程和操作环境的状态(例如,压力、温度、速度和高度)通过传感器来获得,作为输入提供给控制器(即实时的计算机)。每个传感器的数据速率依赖于测量参数的变化快慢,它通常会小于 1KB/s。

有一个确定的应用任务(或工作)的集合[1],即图 1.3 中的"工作菜单"。处理这些任务的软件预先载入计算机。如果这个计算机有一个共享的主存,则全部软件都将被载入。相反,如果它包含一系列属于各个处理器的私有内存,相应的问题就会出现了,即每个任务应该被载入哪个内存。这个问题与任务的分配与调度有关,在本书中我们详细讨论这个问题。

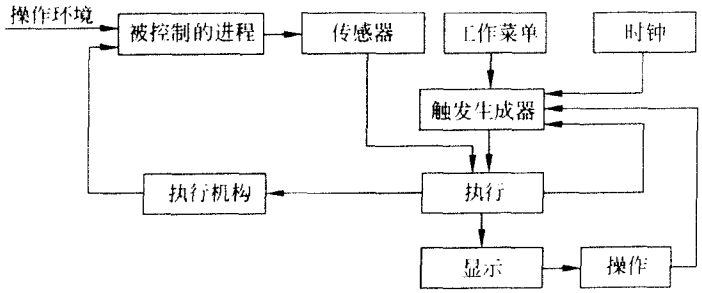

图 1.3 实时系统方框图

"触发生成器"是一种机制的表示,用来触发每个任务的执行。它并不真是一个单独的硬件单元,一般来讲它是执行软件的一部分。很多任务都是周期性的(即它们有规律地执行)。这些任务的进度表可以脱机获得,并且作为一个查找表载入供调度器使用。任务可以根据被控制进程的状态或者操作环境进行初始化。例如,当化工厂中锅炉的压强高出预置的阈值或飞机的高度低于一定的阈值时,可能需要每 x 毫秒执行一次某个任务。最后,任务可以根据操作者的输入面板的命令进行初始化。

计算机的输出被提供给执行机构和显示器。容错技术(将在第 7 章详细讨论)确保即使计算机的输出有一些小错误,执行机构也会被正确地设置。执行机构一般有一个机械或液动部件,因此它们的时间常数很高,这导致执行机构数据速率很低,平均每 25ms 一个命令是很正常的。

一个控制计算机根据数据速率会表现出二义性。传感器与执行机构以相对低

[1] 本书中我们所说的"任务"和"工作"可替换,是在不同工作方式下的说法。