嵌入式操作系统中的抢占式调度策略

■ Jean Labrosse Michael Barr

摘 要 绝大多数嵌入式操作系统采用抢占式的调度方式。本文主要讲述采用抢占式方式进行任务调度的嵌入式操作系统的调度策略和原理。

关键词 抢占式 多任务 中断

通过逐行阅读顺序程序的源代码，不难说出程序会向处理器提出什么样的特定操作，并且也能说出这些操作的顺序。实际上，如果能够知道对一个顺序程序的所有输入，那就不仅能精确预测到处理器所执行的一系列机器码，还能计算出系统最终的输出值或系统行为。这样，无论这个程序运行的速度快慢，都能够得到一个唯一的结果。

然而，在现实中顺序操作程序是很少见的。譬如，在嵌入式系统C程序中的main（）函数，尽管它看上去是顺序的，但是这种看似顺序执行的程序最终会被系统的硬件中断，在某个时刻所打断。当外围设备中断产生后，相应的中断服务程序会运行，从而取代当前main（）函数的执行。这个过程就可以称为抢占。

抢占意味着main（）函数将以比预期慢的速度执行。这是因为它执行速度的快慢与系统中断的数量、中断程序的执行时间以及用于保存和恢复处理器状态的操作时间有着直接的关系。实质上，大部分的处理器周期都被中断服务程序给占用了。除非对中断服务程序有时间的限制，否则，这些中断自身并不会改变系统其它部分的输出，它们只是减慢了程序的执行。

由于绝大多数中断服务程序处理来自于系统设备的中断，这样它们的执行必将会带来系统状态的改变。这种状态的改变最终会给后续指令主序列中的系统行为带来变化，指令主序列则必须做出适当的反应来避免状态改变带来的影响。此时，不但难以预测处理器将执行的操作，同时也难以知道何时，并且以何种顺序来执行这些操作。

绝大多数处理器支持中断的嵌套。一个打断程序顺序执行的中断服务程序可以被另一个更高优先级的中断服务程序所打断。当该高优先级的中断服务程序执行完成后，最初的中断可以在指令主序列之前得以继续执行。

当每一个抢占过程发生时，处理器标志位、当前PC指针以及关键寄存器的内容都应该被保存起来（通常在RAM中），这称为被抢占程序的上下文（context）。这些信息会在该程序进入运行态之前恢复到处理器中。在中断事件产生时，大多数处理器会自动保存这些值，剩下的就是必须执行中断服务程序的入口和出口代码了。

1 伪并行

一个近似的技术就是使处理器像对待处理硬件事件一样处理软件事件。为了达到这个目的，需要将系统划分为一系列独立的事件来处理，即任务。抢占式调度方式使这个想法成为可能。该调度方式管理着系统软件对处理器的使用，并且使系统能够保证对时间要求严格的事件得以有效的执行。

每个任务就是一个顺序执行的函数，并常常以一个无限循环作为结束。这样，就好像任务独占了处理器；同时，每个任务都被赋予特定的工作，如读传感器、扫描键盘、记录一些数据或者刷新显示等。每个任务都拥有一个相应的优先级，并在RAM中有属于自己的堆栈空间。总的来说，这一系列任务一起完成了整个系统所要完成的功能。

当一个高优先级任务抢占一个低优先级任务时，调度器所做的操作与处理器处理中断的操作一致。首先，将当前运行任务的上下文保存到内存中

Jean Labrosse是Micrium的总裁和创始人，他是MicroC/OS-II:The Real-Time Kernel（中译本：《嵌入式实时操作系统μC/OS-II》北京航空航天大学出版社）一书的作者。Michael Barr是《Embedded Systems Programming》杂志的主编。

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

某个地方，然后开始执行新的任务。如果这个新任务在先前已经运行了，那它必有一个保存的上下文，那么，则需要恢复这些内容使其继续运行。当高优先级任务执行完后，调度器将保存其最终的上下文，并且恢复被抢占任务的运行，就好像该低优先级任务从没被打断过。

经过这样划分，每个任务函数可以被写成独占处理器的形式。在实际应用中，往往大多数系统中只有一个处理器。所以在某个特定时刻应该只有一个任务或中断在执行。当没有中断发生时，调度器会根据就绪任务的优先级来决定任务的运行次序。

图1显示了两个不同优先级任务和一个中断服务程序的执行情况。首先，中断服务程序抢占低优先级的任务进入运行态，但是该中断服务程序使一个更高优先级的任务进入就绪态。所以在中断服务程序执行完成后，调度器选择该高优先级任务进入运行，这样就推迟了恢复执行被抢占任务的操作。需要注意的是，处理器总是认为系统中低优先级的中断比高优先级的任务更重要。

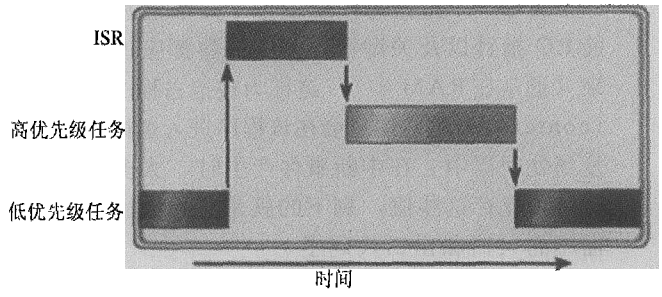

图 1

2 任务控制

有关每个任务的信息，如任务起始地址（在C中就是函数名的地址）、任务优先级和任务执行需要的堆栈空间，都必须提供给调度器。系统调用正是利用这些信息来建立一个新的任务。尽管在不同的操作系统中，这些信息会有所不同，但它们的作用是一样的。

在任务函数的实现体中，可能会调用与软件事件或定时事件相关的系统函数。许多任务会等待一个特定类型的事件并对之作出响应。如一些可能产生一个软件事件；另一些可能做一个100ns的等待，然后再进行重复。

软件事件和超时事件可以由其它任务或中断服务程序来产生。对于后者，可以参看图1。图1显示了一个中断服务程序产生了一个高优先级任务正在等待的事件（从而唤醒该任务）。当然，可能该中断服务程序只是执行了一个时钟中断，而该高优先级任务只是等待计数器到达某一个数值。由于新的软件事件的到来，该高优先级任务将在下一次任务调度后投入运行。

可以采用不同方式，甚至随机地为任务设置优先级。然而，单调执行率算法（RAM）为我们提供了一个理想的方法来保证关键任务的时间期限总能得到满足。

3 权 衡

在使用抢占式调度策略的系统中，内存消耗主要包含系统调用函数额外的ROM和任务堆栈使用的RAM。另一个代价就是CPU时间的损耗。例如，调度策略会消耗处理器的时钟周期；上下文切换和时钟节拍会消耗相当一部分的CPU时间，特别是它们频繁发生的情况下。

当任务共享诸如全局变量、数据结构或外设控制以及状态寄存器等系统资源时，此时一个称为互斥的系统机制被用来避免对这些共享资源竞争。互斥机制是一个避免资源竞争的有效方法；但同时

面向大学赠书《嵌入式实时操作系统μC/OS-II》

《嵌入式实时操作系统μC/OS-II》（第2版）一书的作者Jean Labrosse先生为美国Micrium公司总裁。作为该公司的中国代理，麦克泰公司受Jean Labrosse先生委托，向中国高等院校从事μC/OS-II教学的优秀教师免费赠书。所赠之书是由清华大学邵贝贝教授主持翻译的，已于今年6月份由北京航空航天大学出版社出版。

μC/OS-II在任何商业产品中使用均需首先获取正式产品授权，详情请见www.micrium.com或书的前言，用于大学教育是免费的。书中所附μC/OS-II是2.52版本，与书中内容完全一致，如需要使用μC/OS-II最新版本并获得技术支持，请购买μC/OS-II产品授权。全部大学赠书共250本，赠完为止，详情请访问麦克泰公司网站http://www.bmrtech.com/uCbook.htm。

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

蓝牙关键技术与智能家居

■ 华中师范大学 王莉 黄光明
■ 空军第一航空学院 赵丽

摘 要 蓝牙技术是一种短距离的无线连接技术，智能家居系统是整个未来智能小区系统的重要组成部分。在分析蓝牙技术的基础上，探讨蓝牙技术在智能家居系统中的应用。

关键词 蓝牙技术 微微网 智能家居系统

1 概 述

伴随着数字化和网络化的进程，智能化浪潮席卷了世界的每一个角落，成为势不可挡的历史大趋势，其中正在兴起的智能家居建设热潮，就是在这种形势下应运而生的。

但是现代家庭中，弱电线缆越来越多，如电话线、有线电视线、宽带网络线、防盗报警信号线等，带来线缆多、乱的麻烦，因此，家庭弱电系统需要进行统一、规范的管理。然而，传统家庭布线方式因为施工不规范、维护和使用不方便等因素，已不能适应当前家庭装修的需要，更无法满足未来智能家居生活的更高要求。蓝牙技术的出现，正好解决了这个问题，使智能家居中的无线控制成为可能。

2 蓝牙技术

2.1 蓝牙简介

蓝牙技术是Ericsson移动通信公司在1994年开始启动的，其目的是实现最高数据传输速率1Mb/s(有效传输速率为721kb/s)、最大传输距离为10m的无线通信。

“蓝牙”(Bluetooth)是一个开放性的、短距离无线通信技术标准，也是目前国际上最新的一种公开的无线通信技术规范。它可以在较小的范围内，通过无线连接的方式，实现固定设备以及移动设备通信环境之间灵活、安全、低成本、低功耗的网络互联，使得近距离内各种通信设备能够实现无缝资源共享，也可以实现在各种数字设备之间的语音和数据通信。由于蓝牙技术可以方便地嵌入到单一的CMOS芯片中，因此，特别适用于小型的移动通信设备，使设备去掉了连接电缆的不便，通过无线建立通信。

蓝牙技术工作在全球通用的2.4GHz ISM频段。从理论上讲，以2.4GHz ISM频段运行的技术能使相距30m以内的设备互相连接，但实际上很难达到。现阶段，蓝牙的发射范围可达10m，可以同时实现8台设备的相互联通。当检测到距离小于10m时，接

它会带来一个新的问题——优先级反转（见参考文献）。

在一些应用中，将系统分成一个个独立任务来设计，并使用抢占式的调度策略。这将简化系统设计，但这种方法的利弊是需要权衡的。我们只有充分考虑这些权衡条件，才能对该方法是否适合我们的应用做出正确的判断。

北京师范大学 周东
北京麦克泰软件技术公司 何小庆 译

参考文献

1 Barr Michael, David Stewart. Rate Monotonic Scheduling. Embedded Systems Programming, 2002(3)

2 Barr Michael, David Kalinsky. Priority Inversion. Embedded Systems Programming, 2002(4)

3 www.ucos-ii.com , www.bmrtech.com

（收修改稿日期：2003-06-17）

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net