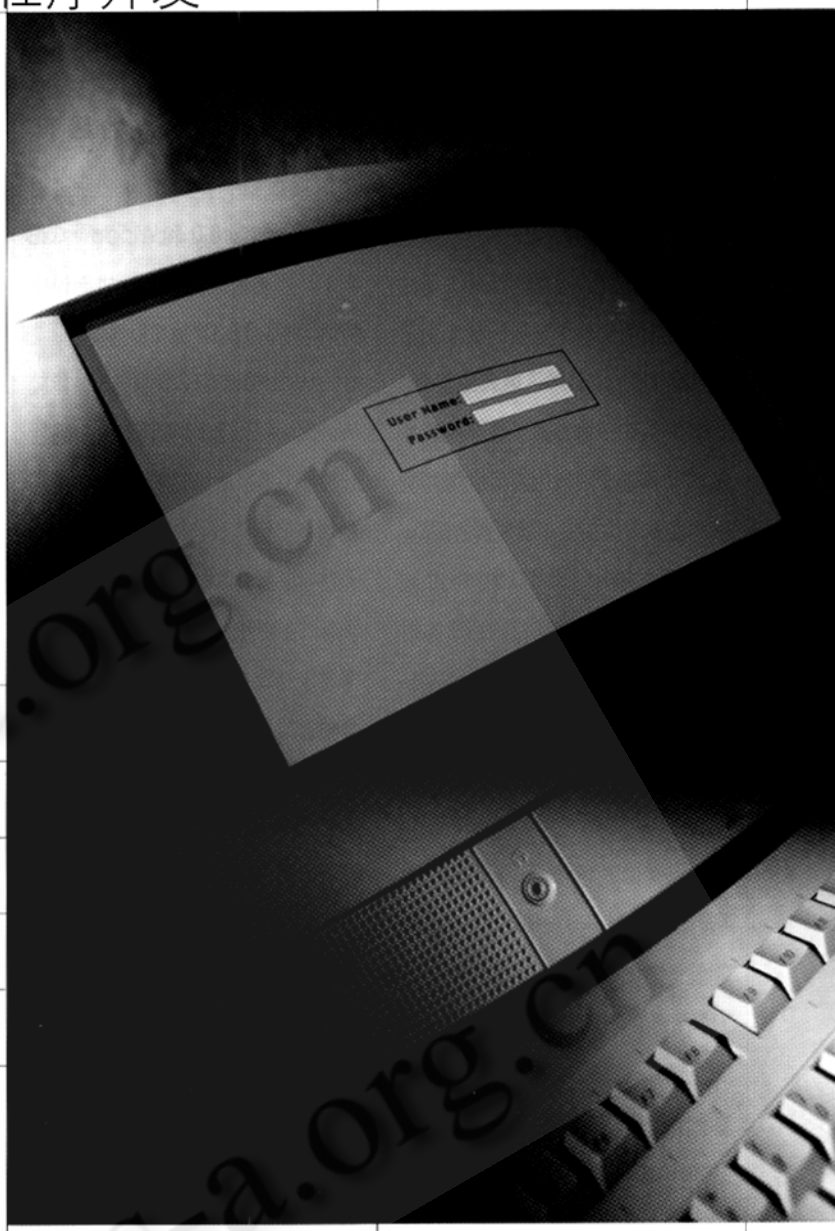
Windows CE 3.0 下的实时应用程序开发



刘少情　吴慧中（南京理工大学计算机系 210094）

摘要：本文讨论了一个实时操作系统应该具备的功能，给出了Windows CE 3.0 所提供的实时性能，并详细讨论了在基于 Windows CE 3.0 的实时系统中如何利用 Windows CE 3.0 提供的实时性能开发实时应用程序以及在开发过程中一些需要着重注意的问题。

关键词：实时操作系统　实时性能　实时系统　实时应用程序

1 引言

近年来，嵌入式系统的应用越来越广泛，在许多嵌入式应用，例如，制造过程控制，高速的数据采集设备，电信交换设备，医用设备，武器发射装备，空间航行和导航，机器人系统中，对系统的实时性要求很高，它们要求系统在规定的时间参数内做出正确的响应，这样的系统就是实时系统，其中有响应时间限制的应用程序就是实时应用程序，因此实时应用程序的开发成为开发一个优秀的实时系统的重要部分，下面首先给实时操作系统应该具备的功能，然后简要说明 Windows CE 3.0 中提供的实时功能，最后详细讨论如何在 Windows CE 3.0 下开发实时应用程序以及应该注意的一些问题。

2 实时操作系统及其功能

实时系统是那些对任何外部事件都能够做出正确及时反应的系统，它不仅要求系统能够给出正确的计算结果，而且在的限定的时间参数内应该给出计算结果，要开发一个良好的实时系统，就必须有实时操作系统的支持，实时系统代表所有系统组成设备—硬件，操作系统和应用程序-它需要达到系统的要求，实时操作系统(Real-Times Operating System，RTOS)只是整个实时系统的一个组成部分，它必须提供足够的功能以确保整个系统达到实时性要求。

RTOS是嵌入式实时应用软件的基础和开发平台，它是一段嵌入在目标代码中的软件，用户的其他应用程序都建立在RTOS之上，RTOS中最关键的部分是实时多任务内核，它的基本功能包括任务管理、定时器管理、存储器管理、资源管理、事件管理、系统管理、消息管理、队列管理等，这些管理功能是通过内核服务函数形式交给用户调用的，也就是RTOS的API，RTOS的引入，解决了嵌入式软件开发标准化的难题，引入RTOS相当于引入了一种新的管理模式，对于开发单位和开发人员都是一个提高，嵌入式软件的函数化、产品化能够促进行业交流以及社会分工专业化，减少重复劳动，提高知识创新的效率。

对于一个实时操作系统，必须具有下面的全部或大部分功能：

(1) 操作系统必须是多线程；

(2) 操作系统必须支持线程优先级；

(3) 一个优先级继承的系统必须存在；

(4) 操作系统必须支持可预测的线程同步机制，另外，操作系统的行为必须是可预测的，这意味实时系统的开发者必须对系统中断级，系统调用和分时了如指掌；

(5) 必须知道操作系统和设备驱动器中数据匹配最大时间；

(6) 设备驱动程序用来处理一个中断最大时间和关于这些驱动程序的中断申请信息必须清楚；

(7) 中断响应延时必须可预测并满足

功能要求；

(8)每次系统调用时间必须可以预测，并且独立于系统的对象数目。

Windows CE 3.0是Microsoft在CE前几版的基础上针对市场的需求推出的又一个增强的版本，它在实时性能方面作了很大的优化，内核中的许多地方为了增强实时性能作了修改，同时也为开发者开发实时应用程序提供了很多新的接口，因此Windows CE 3.0作为一个嵌入式实时操作系统非常适合于实时嵌入式系统的应用。

3 Windows CE 3.0操作系统的实时性能

什么叫实时性能？对于Microsoft Windows CE 3.0操作系统而言，实时性能定义如下：

(1) 保证最高优先级的线程的时间上限的调度机制。

(2) 保证调度高优先级中断服务例程(Interrupt Service Routine, ISR)的延时上限，在必要的时候，可以在一个有限的时间段内将内核的抢先机制关掉。

(3) 对调度进程以及它调度的方式具有良好的控制能力。

为了适应实时系统的需要，Windows CE 3.0对内核作了很多优化，以增强性能并减少延时，包括：

①将所有的内核数据结构移到物理内存中，从而大大避免了在运行内核中的不可抢占代码时出现的状态转移查找缓冲区(Translation Look-aside Buffer，简称TLB)时不命中的次数。

②内核中所有不可抢占但可以中断的部分(也被称作Kcalls)，被分成了更小的不可抢占部分。由于不可抢占的部分增加了，这可能导致了一些复杂性，但却可以让抢占机制在一个更小的时间段中关掉。下面是通过对Windows CE 3.0和Windows CE 2.12在实时性方面的比较而说明Windows CE 3.0的实时性能。

Windows CE3.0的实时性能较Windows CE 2.12有了很大的增强，优先级从8个增加到256个，定时器的精确度提高到1毫秒，可以独立于系统定时器为每个单独的线程定义时间片，支持嵌套式中断处理，提供信号量支持等等，这些都使实时应用程序的开发更加方便。

4 Windows CE 3.0下的实时应用程序开发

要使基于Windows CE 3.0的嵌入式系统满足特定实时性要求，除了在硬件上应该支持实时性之外，在系统的软件开发过程中，必须充分利用Windows CE 3.0为用户提供的实时性能，开发出优秀的实时应用程序。下面就讨论在Windows CE 3.0平台中开发实时应用程序的过程如何充分利用操作系统提供的实时性能和应该特别关注的问题。

4.1 进程、线程和优先级控制

作为一个抢先式的多任务操作系统，Windows CE 3.0允许多达32个进程同时在系统中运行。一个进程包括一个或多个线程，每个线程代表进程的一个独立部分，一个线程被指定为进程的主线程，在进程中也能创造一个定数目的辅助线程，可创建的辅助线程的数目仅仅受限于系统资的可用资源。

内核的调度程序先运行具有最高优先级的线程，然后以轮转方式运行具有相同优先级的线程，因此给线程分配优先级是控制运行速度的一种有效手段。

Windows CE 3.0中将线程的优先级从8个增加到256个，0是最高优先级，而255则是最低优先级。前一版本(2.12)的0到7优先级是与Windows CE 3.0中的248到255级相对应的，而其他的248个优先级是实时优先级。更多的优先级使开发者在控制嵌入式系统的调度时具有更大的灵活性并且可以避免任何应用程序由于限制优先级的数量而降低了系统的性能。

对于一个实时应用，例如实时数据采集，可以将它作为一个进程来运行，要创建一个新的进程，可以调用CreateProcess()函数，它将一个新的应用程序装入内存中并启动至少有一个线程的新进程来运行这个应用程序，如果还要创建其他辅助线程，可以调用CreateThread()来创建新的进程。

在Windows CE中，进程是没有优先级的，所有的进程都是平等的，而每个线程都有自己的优先级，在线程创建时，缺省的优先级是THREAD_PRIORITY_NORMAL，除非正式改变线程的优先级，否则线程总是与系统中其他大多数线程拥有相同的优先级。线程创建后，可以取得线程的优先级或设置线程的优先级，函数CeGetThreadPriority()能够返回指定线程的优先级，函数CeSetThreadPriority()能够重新设置线程的优先级，所以，在运行过程中可以根据系统的需要来提高或降低某些线程的优先级，以适应系统的实时性要求。下面的代码显示了如何在程序运行过程改变一个线程的优先级：

```
HANDLE hCurrentThread;
DWORD dwOldPriority, dwNewPriority;
...
hCurrentThread=GetCurrentThread();
dwOldPriority
=CeGetThreadPrioriy(hCurrentThread);
if(dwOldPriority >某个优先级)
 {
  dwNewPriority = dwOldPriority-2; //提升2个优先级
      CeSetThreadPriority(hCurrentThread, dwNewPriority);
}
//运行实时代码
...
//恢复原来的优先级
CeSetThreadPriority(hCurrentThread, dwOldPriority);
//继续运行其他部分
...
```

线程的运行顺序的取决于线程优先级，拥有高级优先级的线程安排优先运行，同一优先级的线程，以轮转方式运行，即每个线程运行指定时间片。较低优先权的线程，要到较高级放弃CPU控制权(放弃时间片或运行完成)之后才能运行。但是在某些情况下，线程的优先级可以被内核自动改变，这就是所谓的"优先级倒置"(Priority Inversion)，优先级倒置在一个低优先级的线程拥有一个高优先级线程所需的内核对象的时候才发

生。为了缩短响应时间，Windows CE 3.0 用优先级继承来处理优先级倒置，在这里一个拥有高优先级线程所需的内核对象的低优先线程继承了高优先级线程的优先级。优先级的倒置使低优先级的线程得以运行，从而释放了高优先级线程所需的资源。在先前的版本（2.12）中，内核处理整个倒置链，而在 Windows CE 3.0 中，内核保证对每一个优先级只处理深度为 1 的优先级倒置。

下面的例子说明 Windows CE 3.0 是如何通过改变优先级倒置的方法来提高系统的实时响应能力。假设在系统中有三个线程 A，B，C，线程 A 以比线程 B 和 C 高的优先级运行，线程 B 拥有一个线程 C 所需的内核对象，并且线程 B 由于等待线程 C 释放它所需的内核对象而阻塞，C 是处在可运行状态。在 Windows CE 2.12 中，当 A 运行并且由于线程 B 而阻塞，那么 B 和 C 的优先级都提升为 A 的优先级，以使它们能够运行。在 Windows CE 3.0 中，当线程 A 由于线程 B 而阻塞，只有线程 B 的优先级提升成 A 的优先级，B 又因为 C 而阻塞，因此 C 的优先级就提升为 B 的优先级。通过减小复杂性和改变算法，Windows CE 中最大的 Kcall（内核中不可剥夺但可中断的代码）大大地减小了和受限了，从而缩短了响应时间。

4.2 对定时器和调度机制的控制

内核要求用户指定一个周期中断时间，称为定时器（也称为系统 Tick），系统 Tick 源每隔一个 Tick 的时间向 CPU 发一次中断。在先前的版本中，OEM 在 OA（OEM Adaptation Layer）中将定时器和线程的时间片设置成一个常量，通常是 25 微秒。在 Windows CE 3.0 中定时器被设置成 1 微秒，但是可以为每个线程单独设置时间片，因此定时器和 Sleep()函数以及 Wait()函数的精确度为 1 微秒，而定时器不再与线程的时间片直接相关的。

另外，在 Windows CE 3.0 中，系统开发者可以通过实现自己的时钟中断处理函数来控制调度进程的调度方式。在 Windows CE 3.0 的内核中提供了几个开发者可以用来决定在系统 Tick 到来时是否需要重新调度的变量。系统时钟中断服务例程在适当的时候以通过返回 SYSINR_NOP 而不是返回 YSINTR_RESCHED 来避免内核进行重新调度。

4.3 控制线程的时间片

当有多个相同优先级的线程准备好运行时，调度程序会以轮转的方式让每个线程运行一个时间片，在 Windows CE 3.0 中，缺省的线程时间片是 100 微秒，这个值是由 OEM 在 OAL 中设置的，该值在整个操作系统中是作为一个常量。Windows CE 3.0 允许应用程序基于线程来设置每个线程的时间片，这意味这开发者可以调整调度程序以满足自己应用程序的当前要求。为了调整线程的时间片，Windows CE 3.0 提供了两个新函数：CeGetThreadQuantum()和 CeSetThreadQuantum()，CeGetThreadQuantum()可以取得一个指定线程的时间片大小，而 CeSetThreadQuantum()可以为线程设置特定的时间片大小。下面的代码显示了如何在程序中改变一个线程的时间片大小：

```
HANDLE hCurrentThread;
DWORD dwOldQuantum, dwNewQuantum;
...
hCurrentThread=GetCurrentThread();
dwOldQuantum=CeGetThreadQuantum(hCurrentThread);
if(dwOldQuantum<某个数值)
{
 dwNewQuantum=dwOldQuantum+100; //时间片增加 100 微秒
 CeSetThreadQuantum(hCurrentThread, dwNewQuantum);
}
//运行实时代码
...
// 恢复成原来的时间片大小
CeSetThreadQuantum(hCurrentThread, dwOldQuantum);
//继续运行其他代码
...
```

4.4 中断处理与嵌套中断

4.4.1 Windows CE 中的中断处理方式

实时应用利用中断作为一种保证外部事件能够快速地为操作系统所知的方法。在 Windows CE 中，内核和 OAL 都为中断的传送和将事件分派到系统中的其他部分作了优化。Windows CE 将中断处理分割成两个步骤：中断服务例程（Interrupt Service Routines，ISR）和中断服务线程（Interrupt Service Thread，IST）而使实现起来更容易。

每一个硬件中断请求线都与一个 ISR 相联系。当中断没有被禁止并且有中断发生的时候，内核就调用这个中断的注册的 ISR，ISR 是中断处理的内核模式的部分，它的代码一般都很短，ISR 的主要任务是指导内核来启动相应的 IST。

ISR 完成它的最小处理并将一个中断标识符返回给内核，内核检查返回来的中断标识符并设置将 ISR 与 IST 相联的相关事件。IST 一直在等待相应的事件到来。当内核设置了事件后，IST 就停止等待并如果它是具有最高优先级准备运行的线程就开始完成他其他的中断处理。大多数中断处理实际上是在 IST 中完成的。

4.4.2 中断的嵌套

在 Windows CE 的前一版本中，当一个 ISR 在运行的时候，所有其他的中断都被关掉。这就使内核在一个 ISR 完成之前不能处理任何其他的中断。因此，如果一个高优先级的中断已经准备好，内核在当前的中断服务例程没有完成并返回到内核之前是不能处理这个新的中断。为了防止高优先级的中断的丢失和延迟，Windows CE 3.0 支持基于优先级的中断嵌套，如果 CPU 和（或者）其他相关的硬件支持嵌套式中断处理。当一个 ISR 在 Windows CE 3.0 中运行的时候，内核和以前一样运行特定的 ISR，但是只禁止同级和低级的 ISR。如果有一个高优先级的 ISR 准备好运行，内核就保存正在运行的 ISR 的状态，并让高级的 ISR 运行。内核可嵌套的中断的数量是与 CPU 支持的嵌套的数量相同，ISR 嵌套的顺序是它们的硬件优先级。

4.4.3 中断延时的控制

内核实时性能的一个重要特性就

是在特定的时间那为一个中断提供服务。中断延时主要是指软件中断处理延时，即从外部中断到达处理器到中断处理开始所经历的时间。Windows CE 3.0 的内核对于锁定在内存中的线程作了优化，从而使得中断延时缩短。在进行系统开发时，开发者可以通过控制中断嵌套的层次来控制ISR和IST的延时，根据系统的实际情况，适当降低嵌套的层次，既可以保证高优先级的中断能够得到及时的响应，又可以限制所有中断服务例程的延时，不至于使得某些中断的延时过大而出现"饿死"的情况。

4.5 内存与实时性能

Windows CE 3.0 内核支持几种类型的内核对象，包括进程，线程、临界区、互斥体、事件和信号量。因为操作系统使用虚拟内存，所有的内核对象都位于虚拟内存中，而这些对象的内存是在需要的时候才分配的。因为在需要的时候分配内存会影响性能，所以在实时应用程序的过程中应该在一个线程开始的任何时候分配内核对象。有三种类型的内存可能影响实时性能，它们分别是虚拟内存，堆内存和堆栈内存。为了保证应用程序的实时性能，在程序中分配内存时要注意以下几点：

(1) 在虚拟内存分配请求期间，内核从物理内存缓冲池中查找空闲的物理内存，由于依赖于正在被使用的内存和内存分片的情况，因此查找内存的时间是变化的。为了减少分配虚拟内存而对系统实时性产生影响，一个进程应该在进行正常的处理之前分配和提交所有的虚拟内存。

(2) 一个应用程序可以利用进程的堆来用作内存分配，堆的大小会根据需要相应地增长或缩减。然而，如果由于内存的分配而导致堆中形成了许多碎片，就会造成对实时性能的损害。如果一个堆变成了许多碎片，内核就会花更多的时间来寻找为一个新的内存分配的空间，这就可能影响性能。为了防止在一个进程中出现过多的堆碎片，OEM就应该为相似的对象创建独立的堆来控制内存的分配过程。如果一个堆中只包含大小相同的对象，堆管理器就会更容易地找到一个合适新的内存分配需求的空闲块。为了减少堆内存分配而对系统实时性产生影响，一个进程应该在进行任何正常的处理之前分配堆内存。

(3) 当一个新的线程在系统中创建时，内核为堆栈保留内存。为堆栈保留的内存的大小是由模块编译时传给连接器的/STACK参数来决定的。当一个线程在第一次被调度的时候，线程的堆栈内存就提交了。为了防止初始提交的堆栈内存影响系统的性能，开发者应该保证线程在进行实时处理之前至少调度一次。如果一个线程不需要比初始分配的堆栈更大的堆栈空间，那么进一步的堆栈提交可以避免。

5 总结

上面讨论了Windows CE 3.0 的实时性能，并且详细讨论了在Windows CE3.0下如何利用它提供的实时性能开发实时应用程序以及在开发过程中需要注意的各方面的问题。Window CE 3.0作为一个优秀的嵌入式实时操作系统，在以后的实时系统中会得到更广泛的应用。

参 考 文 献

1 Designing and Optimizing Microsoft Windows CE 3.0 for Real-Time Performance Microsoft Corporation. September 2000。

2 Introducing Microsoft Windows CE 3.0 Microsoft Corporation. January 2001。

3 Douglas Boling Programming Microsoft Windows CE Microsoft Press 1998, 10。

4 John Murray Inside Microsoft Windows CE Microsoft Press 1998, 9。

5 洪英、陈曦，嵌入式实时多任务程序设计，计算机应用，2000年7月。