实时多任务执行模型到 Windows NT 的映射

刘晓燕,张云生,字天文,李俊昌
LIU Xiao-yan,ZHANG Yun-sheng,ZI Tian-wen,LI Jun-chang

昆明理工大学 信息自动化学院,昆明 650093
School of Information Engineering and Automation,Kunming University of Science and Technology,Kunming 650093,China
E-mail:lxykmust@yahoo.com.cn

LIU Xiao-yan,ZHANG Yun-sheng,ZI Tian-wen,et al.Mapping real-time multitasking execution model to Windows NT. Computer Engineering and Applications,2008,44(8):110-112.

Abstract: Presents mapping guidelines of a real-time multitasking execution model to system calls and C++ program skeletons under Windows NT platform,based on research and analysis for an abstraction executive of real-time software and Windows NT. The principles of communication between real-time objects and mapping methods are also given.
Key words: real-time multitasking;execution model;Windows NT;mapping guidelines

摘 要: 基于对实时多任务软件的抽象执行体的研究分析以及对 Windows NT 操作系统的分析,提出了实时多任务执行模型到 Windows NT 平台下的系统调用及 C++语言程序框架的映射规则。给出了实时对象之间的通信原理及映射方法。
关键词:实时多任务;执行模型;Windows NT;映射规则
文章编号:1002-8331(2008)08-0110-03 **文献标识码:**A **中图分类号:**TP312

1 引言

支持软件开发的工具有很多,但是支持针对实时多任务操作系统的实时应用的基于构件的软件开发工具却不多见。目前,大家已认识到无论是在软件的需求分析阶段还是在设计阶段,图形化工具可以简化实时应用的开发并且提高效率。

笔者研制的基于构件的分布式实时软件开发环境 DRSCDE(Distributed Real-time Software Component Development Environment)是针对实时多任务应用系统的图形化软件设计工具,以图形方式支持软件生存周期的基于构件总体设计阶段、详细设计阶段以及目标代码的自动生成。DRSCDE 的设计语言由图形符号表示,包括程序设计语言三种语句控制结构及实时多任务执行体支持的对象及系统调用的图形符号表示。在用图形符号语言进行设计的过程中,自动生成设计图的文本描述语言,可看成是实时应用程序的文本描述中间语言或伪码。最后通过代码生成器自动生成目标机的目标实时执行体的 C 或 C++语言程序。其目标代码的生成依赖于特定的目标实时执行体,该工具目前的实现版本是生成了实时多任务操作系统 iRMX 下的 C 语言程序框架。所产生的 C 代码程序包含必要的声明及所需要的系统调用,数据声明还需人工参与[1]。

Windows NT 操作系统(以下简称 NT)以其标准的图形用户界面,简单灵活的操作、多任务以及设备无关性等优越特性得到了广泛的应用。在工业控制领域内,用户也开始要求其控制系统具有 Windows 风格。因此,在 NT 这种开放、标准、通用的操作系统上进行实时系统的开发成为需要。尽管 NT 不是专为实时系统设计的操作系统,但是 NT 是具有一定实时特征的通用操作系统,对实时系统的支持较好,可以在一定程度上实现实时控制系统。针对我国的实际情况,由于 NT 操作系统的易得性,广大的实时多任务应用开发者大多选择 NT 作为目标实时多任务操作系统极其所支持的 C 或 C++语言作为开发语言。由此,选择 NT 作为目标实时多任务操作系统,研究将 DRSCDE 的中间代码自动生成 NT 的 API 及 C++实时代码程序框架技术及代码生成器工具原型。而 DRSCDE 实时多任务执行模型到 NT 平台下系统调用的 C++语言程序框架的映射规则是实现目标代码自动生成的关键。

本文首先分析 DRSCDE 可执行模型的建模及实时执行体操纵的对象,分析 NT 的实时特征;再找出对应各 DRSCDE 实时执行体对象的文本语言中间代码到 NT 实时执行体对象映射关系及必要的数据声明和系统调用 API。

2 实时多任务执行模型

DRSCDE 是基于实时多任务操作系统 RTMOS(Real-time Multitasking Operating System)的图形化的分布式实时软构件设计开发环境。本节仅介绍该工具对实时多任务过程设计层可执行模型相关概念。DRSCDE 对流行的 RTMOS 如 VRTX、iRMX、VxWorks、RTC 等抽象了实时执行体或执行核的常用对象,它们分别是任务(Task)、中断(Interrupt)、信号量(Semaphore)、

基金项目:云南省教育厅资助科研课题(the Research Project of Department of Education of Yunnan Province,China under Grant No.07C10799)。
作者简介:刘晓燕(1964-),女,博士生,副教授,主要研究领域为软件开发环境及开发技术;张云生(1948-),男,博士生导师,主要研究领域为计算机控制及实时软件开发环境。
收稿日期:2007-07-25 **修回日期:**2007-10-17

消息(Message)和邮箱(Mailbox)等。笔者把这些实体定义为原子对象,其图形表示详见文[1]。这样,图形设计工具中过程设计层的原子对象不必在每个实时执行器中去找,目标代码的生成依赖于目标实时执行体的具体对象。设计者在实时构件服务的过程设计阶段,要用到原子对象、状态原语及动作原语。原子对象分为两类:客户对象(又称为可编程对象)和服务器对象(又称为可配置对象)[2]。

客户对象:是应用程序可执行模型的动作元素,表示信息转换的过程。有三种客户对象:任务、中断和报警。任务或线程是由 RTMOS 调度的实体,中断是对外部事件或硬件设备的响应,报警对异常情况发出警告。

服务器对象:动作完全由对象创建时(实例化)定义的参数决定的对象。有信号量、消息、消息池和邮箱,用于客户对象之间的通信和同步。

动作原语:表示对象的行为。原子对象的行为由编程人员以动作原语(主要为通信和同步原语)来定义,如创建原语、删除原语、发送消息及读/写邮箱等。

状态原语:设置对象的状态,对所有的实时对象都是公用的,如挂起原语、恢复原语、连接原语和断开原语等。

3 NT 的实时特征

NT 不是专为实时系统设计的却是具有一定实时特征的通用操作系统,对实时系统的支持较好,可以在一定程度上实现实时控制系统,但却存在一些问题。

首要的问题是优先级级别的管理,它可用的优先级级别只有 32 级,用于描述一个复杂的实时多任务应用是不够的。

第二个问题是中断管理,其中断分两个阶段操纵。第一阶段在 ISR(Interrupt Service Routine)中处理中断;第二阶段在 ISR 中判断是否需要进一步处理中断, 是则请求 DPC(Deferred Procedure Call)处理。DPC 使用 FIFO 方式进行管理,没有指定完成一次运行所需要的时间。因此,预测系统的时间性是不可能的,并且要恢复 NT 下的实时应用也是不可能的。

第三个问题是基于计时器的快速显示及系统原语的执行时间期间是不可预测的。

NT 下的时间代码在原型工具中不能决定, 但是笔者认为更有效且值得做的事情是分析及讨论所选择的目标执行模型代码生成的可能性。

4 实时多任务模型到 NT 下代码的映射方法

DRSCDE 中的实时多任务可执行模型采用了面向对象的建模技术, 而 NT 的设计和实现也遵循了实时操作系统的标准,采用了面向对象的建模技术。NT 执行体对象有进程、线程、事件、文件、文件映射、信号量、互斥量、邮件槽及管道等等[4],这些对象可以通过调用不同的 win32 API 函数产生。因而,两个系统之间的建模对象具有相似性, 从而使得 DRSCDE 的一些建模对象到 NT 内核执行体对象的映射成为可能。

显而易见,DRSCDE 的三种控制结构形式(顺序、选择和循环结构)和 NT 下 C++语言语句控制结构是一一映射的。

4.1 任务的映射规则

DRSCDE 的任务是可编程对象之一,是一个具有独立功能的无限循环的程序段的一次运行活动, 是实时内核调度的单位。任务一旦创建,就处于以下五种状态之一,运行态、活动就绪态、活动等待态、挂起就绪态及挂起等待态等。DRSCDE 中的任务对应 NT 下的线程(thread),在 NT 中,进程是应用程序的一个运行实例,线程是描述了在进程中的一条执行路径。初始化一个进程时,系统会创建一个主线程并启动该线程。下面给出任务对象的创建及撤销动作原语及挂起及恢复状态原语的文本描述语言到 NT 的映射关系。

4.1.1 任务创建及撤销

DRSCDE 任务创建的原语是 CREATE_TASK(task_tk,priority),其中,task_tk:任务标识符;priority:任务优先级。在 NT 中, 创建线程的 API 函数为 CreateThread。由此, 创建任务 CREATE_TASK 在 NT 下的实现是 CreateThread 函数, 在目标代码中的各参数位置插入 DRSCDE 动作原语的参数。

任务撤销原语是 DELETE_TASK (task_tk)。在 NT 下终止线程有以下三种方法:(1)线程通过调用 ExitThread 函数来终结,这是最常用的方法。(2)调用 TerminateThread 函数终止在同一进程或另一进程中的某个线程,一般尽量避免使用这种方法。(3)包含有该线程的进程终结了。这里采用第一种方法来实现线程的终止,因此,任务撤销 DELETE_TASK 在 NT 下的映射是 CreateThread 函数。

4.1.2 任务挂起及恢复

DRSCDE 任务挂起的原语为 SUSPEND(task_tk)。映射为 NT 下挂起线程函数 SuspendThread。

任务恢复原语为 RESUME(task_tk)。映射为 NT 下恢复线程函数 ResumeThread。

4.1.3 任务优先级的管理

NT 可用的优先级级别只有 32 级,用于描述一个复杂的实时多任务应用是不够的。DRSCDE 设计的任务优先级的区间数是 0~255。问题是如何将 DRSCDE 较大的优先级级别映射到 NT 所提供的较小的优先级级别。该问题可分为两种情况解决:

(1)由 DRSCDE 的任务执行模型的优先级级别推断目标代码的优先级,目标代码的优先级的顺序应保持原执行模型的任务优先级顺序。如一个应用系统由三个任务构成,其优先级分别是 20、30 和 40,可将其映射为 NT 下优先级的级别为 2、3 和 4,保持原任务优先级顺序不变。

(2)DRSCDE 的任务优先级级别数多于目标系统的优先级数,其区间数是 0~255。解决方案是将 DRSCDE 任务优先级区间数 0~255 分为 32 个小区间,如表 1 所示,优先级在 255~239 之间的定义为第 1 个区间,则某一个区间的优先级映射为目标系统 NT 的一个优先级。NT 的同一个优先级级别将被赋值给多个任务的优先级, 同一优先级级别的任务再分时进行管理。虽然一个区间内的若干优先级映射为一个优先级,会导致区间内优先级的缺失,但通过对同一优先级的分时管理,可保障系统原有的总的执行及调度顺序,是一个可行的适用的方案。

表 1 DRSCDE 的任务优先级到 NT 优先级的映射

优先级区间数	DRSCDE 任务执行模型的优先级级别	相应的 NT 优先级级别
1	255~239	31
2	238~222	30
3	221~205	29
4	204~188	28
5	187~171	27
	…	…
	…	…
32	15~0	0

DRSCDE 获得任务优先级原语是 INQUIRE(task_tk)。映射为 NT 的系统调用 GetThreadPriority,该函数用于查询线程的相对优先级。

改变任务优先级原语为 CHANGE_PRIORITY(task_tk,priority)。在 NT 下,创建线程时,并不需要用数值为它们分配优先级,由系统确定,系统提供了改变线程相对优先级的函数 SetThreadPriority。因此,改变任务优先级 CHANGE_PRIORITY 原语,映射为 NT 的函数 SetThreadPriority。

4.2 中断例程对象

NT 对中断的管理作了限制, 通过中断向量表进入中断是不可能的,中断被限制为系统中断,在使用级别上,授权给键盘中断(如 CTRL+C),该中断可通过应用 C 函数 Signal 得到。函数 Signal 的参数是信号名及信号到达时执行的过程名。

因此提出 DRSCDE 的中断对象 INTERRUPT 映射为如下 NT 代码:

```
Signal (SIGINT,flSR_nom);
Void_cdecl flSR_nom (int arg)
{
PostThreadMessage(h_Thre_PSnom,WM_MSG,NULL,(LPARAM));
/* 传送一个消息给偶发性任务 */
}
```

4.3 报警对象

报警 ALARM(Alarm_id,source)是一种预先定义的中断形式,是一个并发、同步处理的事件,如激活源可以是中央处理器的时钟。DRSCDE 执行模型的报警源 source 有两种类型:type 和 period,分别表示报警类型和报警时间区间。type 又有两种类型:S 和 C。S:代表简单事件,C:代表周期性事件,是由日期激活的报警。若 type 为 S,则 period 代表一个具体的激活日期;若 type 为 C,则 period 代表一个具体的相邻两个报警被激活的时间周期,总是通过计时器来进行管理。因此,将其映射为 NT 的计时器,通过函数 TimeSetEvent 进行参数设置。参数 S 映射为函数参数 TIME_ONESHOT, 表示只激活一次; 参数 C 映射为函数参数 TIME_PERODIC,表示每隔一段时间重复激活,即周期性的激活。如参数 C 用如下 NT 代码表示:

```
UINT idTimer ();
{
… …
idTimer()=TimeSetEvent (Period,1,id1,NULL,TIME_PERODIC);
/* 周期性的初始化报警 */
}
```

4.4 实时对象之间的通信

在 DRSCDE 中,可配置对象信号量、消息和邮箱是可编程对象之间通信的一种重要手段,用来在可编程对象之间传递信息。通过信号量、消息和邮箱对实时可编程对象(任务、中断和报警)之间的通信进行建模。实时操作不同,可能实现的方法不同,但基本原理大致相同,通常使用先进先出模式 FIFO 或优先级模式。

图 1 显示了 DRSCDE 中通过消息和邮箱两个简单且典型的通信。在无限循环中,任务 Task_A 创建消息 MSG。该任务等待某个假定的事件发生,然后通过邮箱 mbx_c,发送消息给任务 Task_B。在 Task_A 循环创建一个新的消息,然后再次等待事件。Task_B 等待邮箱 mbx_c 中的消息、处理消息、发送信号给另外的任务,最后撤消收到的消息。

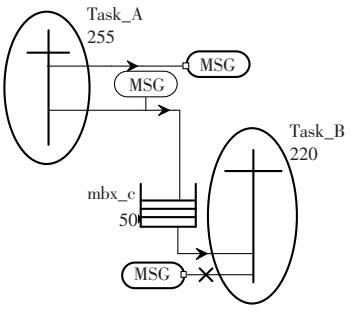

图 1 任务间的通信模型

4.4.1 信号量对象的映射规则

信号量是用于线程间同步及资源同步的常用对象。DRSCDE 中的信号量对象用于任务之间及资源使用的同步,任务队列通过信号量对象申请信号状态。有两个原语可以动态地改变信号量状态: 发送原语 SendToSem 和等待原语 WaitOnSem。队列管理方式有 FIFO 和 PRIORITY 两种,信号量以 FIFO 或者优先权方式在队列中调度。

NT 下的信号量内核对象同步采用资源计数法, 计数大于 0,信号量为有信号;计数等于 0 时,为无信号。其队列管理方式仅有 FIFO。信号量自动执行测试加设置的操作,即当从信号量请求一个资源时,操作系统察看资源是否可用,并减少可用资源的计数而不允许其它线程干涉。只有在资源计数被减少后,系统才允许另一个线程请求资源。

DRSCDE 创建信号量原语 CREATE_SEMAPHORE(name,max,init,mode),创建执行所必需的信号量并且初始化。映射为 NT 下的 API 函数 CreateSemaphore。

撤消信号量原语 DELETE_SEMAPHORE(name),作用是销毁信号量。可映射为 NT 下的函数 CloseHandle。

发送原语 SEND_TO_SEM(sem_id,[nu],[flow]),用于减少资源计数。可映射为 NT 下的函数 ReleaseSemaphore。

等待原语 WAIT_ON_SEM(sem_id,[nu],[timeout]),用于获得信号量, 在指定的时间内等待资源可用, 与 NT 下的函数 WaitForSingleObject 有相同语义。因此,可映射为 NT 下的函数 WaitForSingleObject。DRSCDE 中,如果调用任务申请的同一类型设备数 unit 大于 1, 则映射到 NT 下, 需多次调用 WaitForSingleObject(次数为 unit),但释放时,可以不必多次调用 ReleaseSemaphore。

4.4.2 消息及邮箱的映射规则

DRSCDE 消息及消息池由消息标识符(Mes_id)来识别,是目标语言的数据结构(如数组、记录、结构等)。消息可以为每个消息作标识,消息池可以为一组消息作标识。因此,对应 NT 下的数据结构声明:MSG Mes_id。

创建消息原语 CREATE_MESSAGE(Mes_id)对应 NT 下的用 new 函数为消息申请一个 MSG 类型的内存空间。删除消息原语 DELETE_MAEEAGE(Mes_id)对应 NT 下的用 delete 函数为消息释放内存空间。

邮箱是任务之间交换信息的机制,用于消息的传递,是存放消息的队列,它可以由多个任务访问。消息寄发到邮箱,根据取出访问操作使用两种模式:先进先出模式 FIFO 或优先级模式。

(下转 144 页)