嵌入式实时操作系统核心的设计与实现

屈振新　曾庆伟　韩　波
(中南财经政法大学信息学院,武汉 430064)
E-mail:quzx@263.net

摘　要　针对广泛使用的基于 8051 核的 MCU,给出一个嵌入式实时操作系统核心的设计方案。具体阐述了时间片、任务调度、信号量和中断处理的实现。

关键词　嵌入式　实时操作系统　任务　调度　信号量

文章编号 1002-8331-(2002)09-0123-02　**文献标识码** A　**中图分类号** TP316.2

Design and Implementation of an Embedded Real-time OS Kernel

Qu Zhenxin　Zeng Qingwei　Han Bo
(Department of Information,ZhongNan University of Economics&Law,Wuhan 430064)

Abstract: Based on 8051 MCU,this paper describes the design of an embedded real-time operating system kernel.It describes the realization of the clock tick,dispatcher,signal,interrupt processing.

Keywords: Embedded,Real-time OS,task,dispatcher,signal

1　简介

在传统的嵌入式程序设计中，如果有一个以上的任务,一般都是采用一个无限主循环，依次对每一个条件进行判断,一旦某件成立,则立即执行相对应的模块完成某任务。各模块按照编写的顺序依次顺序执行（当然每一个模块都有执行的前提条件)。在这期间,一旦发生中断,由中断服务程序 ISR 负责处理,整个过程如图 1 所示：

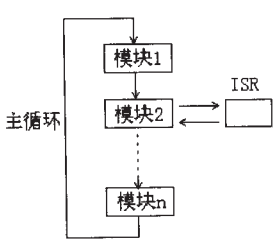

图 1

对于简单的应用,这种方式是可以满足需求的,随着应用的复杂化，系统可能同时要控制/监视很多外设，要求实时响应,有很多任务要处理。如果仍然采用传统的程序设计方法,就会出现两个比较突出的问题。一是中断响应。一般为了保证某项重要任务的实时性，就必须在 ISR 中进行处理，这样一来，ISR 的执行时间就会变长。而一般 ISR 应该非常短小。如果 ISR 只是简单地从中断设备中读取数据和状态信息,而将具体的处理过程交给任务模块去执行,因为在主循环中有若干任务模块顺序执行，所以从中断响应完成到相应任务模块的开始执行,这里有一段延迟,称其为任务响应时间。最大任务响应时间由整个循环体的执行时间来决定,但是循环体的执行时间并不固定,所以任务响应时间也不固定。二是任务多,要考虑的各种可能也多,各种资源调度不当就会发生死锁,降低系统的可靠性,同时程序量也加大。

解决这些问题的最好方法是使用嵌入式实时操作系统 RTOS。使用 RTOS 可以有效地对任务进行调度,对各任务赋予不同的优先级可以保证紧急任务及时响应。RTOS 也体现了一种新的应用程序设计思想和开放的框架，降低了程序的复杂度。目前的嵌入式实时操作系统价格一般比较昂贵,而且很多 RTOS 厂商要求用户在批量生产时要按产品数量交纳版税,对中小系统的成本影响很大。而事实上,中小应用系统往往并不需要一个完整的 RTOS，只需要 RTOS 的基本功能来对任务进行有效的管理,基于这些考虑,设计一个 RTOS 内核,这里给出基本方案。

由于嵌入式 RTOS 对 MCU（CPU)的依赖性比较大,而现有的 MCU 又千差万别,所以首先要选择所依赖的 MCU。8051 内核的单片机是国内目前在中小型应用中广泛使用的一种 MCU，国内有大量用户在使用它，所以就选定 8051 内核的 MCU 作为基本的硬件环境,采用 C 语言和汇编来实现,且系统调用以 C 函数库的形式提供,便于使用。

2　系统实现

在操作系统中对任务进行调度有两种基本的调度方式:非剥夺方式和剥夺方式（可抢先的方式)。前者无法处理各任务间可能存在的优先级差别,后者可以保证高优先级任务的实时响应。但是后者也同时带来了新的问题:当系统中存在几个相同优先级的任务时,就有可能使某一任务独占处理器,直至完成。为此在该方案中采用了基于优先级的时间片轮转调度方式,即在不同的优先级的任务之间以抢先方式进行调度,在相同优先级的任务之间采用时间片进行轮转调度。

2.1　时间片的实现

在 8051 中有两个 16 位可编程的定时器 Timer0 和

基金项目:中南财经政法大学科研项目奖助金资助
作者简介:屈振新,讲师,硕士,主要从事单片机、网络等的研究工作。曾庆伟,教授,硕导。韩波,讲师,硕士。
(C)1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

Timer1，在这里，系统占用了 Timer0 作为时间片（tick）发生器，通过设置，使其每 1ms 产生一次中断，在中断处理中，对运行状态和某些等待状态的任务的时间片计数减一，只要有某个任务的时间片计数变为 0，则马上转入调度程序进行相应的处理。8051 中有两个中断优先级，在此将 Timer0 设为高优先级。Timer1 留给用户在程序中自主支配。

为了保证原语不可中断性，当用户调用原语进入核态时，停止 Timer0 的计数，且关中断。当从核态返回到用户态时恢复 Timer0 的计数，恢复原有的中断状态。

2.2 任务操作原语

对于某个应用系统，可以按照一定的原则（如：操作硬件设备的不同）把整项工作划分成若干个任务，各任务间相对独立，又通过一定的方式来进行协调，如键盘扫描、数据显示、数据采集、日历时钟的读取、数据传输等都可以作为独立的任务来处理。

该方案中实现了如下几条用于操作任务的基本原语：

createtask（prio，id）	创建任务
deletetask（id）	删除任务
suspend（）	挂起任务
wakeup（id）	唤醒任务
delay（tickcount）	延时

各原语以 C 语言库函数的方式提供，当用户调用这些函数时，系统从用户态转为核态，在 suspend 和 delay 原语中，当进行相应的操作后，转入调度程序。其它各原语执行结束后回到用户态。

2.3 任务调度

由于系统采用 C 语言实现主要的程序，所以主核心库中已经包含了 main（）函数，在 main（）函数中做完初始化操作后，自动调用 inittask（）函数，最后转入调度程序。所以在用户程序中不能包含 main（）函数，要把 inittask（）函数作为应用程序的入口，并在其中依次创建各任务。

当各任务创建后，置为就绪状态，并按照一定的原则加入到就绪队列中。在系统内，各任务在三种状态之间变化，运行状态、就绪状态和睡眠状态，关系如图 2 所示。

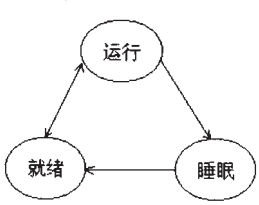

图 2

任务从运行状态转为睡眠状态主要有三种情况：一是某任务自身调用 suspend（）原语，把自己挂起，等待其它进程来唤醒；二是某任务调用 delay（）原语，转为睡眠状态，当指定的时间片过去以后，由调度程序将其唤醒，转为就绪状态；三是某任务执行 P 操作时，若计数信号量变为负数，则由调度程序将其转为睡眠状态，并由其它任务通过信号量的 V 操作将其唤醒，转为就绪状态。

为了对各任务进行有效的管理，在系统内有一任务控制块 TCB 数组，用于记录各任务的信息。各 TCB 大致包括图 3 所示的各项。

所有任务的 TCB 形成两大类链表：总链表和分类链表。所有的 TCB 构成总链表。按照各任务当前状态的不同，又可以得到不同的分类链表（队列）。因为有三种可能导致任务从运行状态转换为睡眠状态，睡眠状态的任务按照所等待发生的事件的不同，又可以细分为三类：等待延时完成的任务、等待其它任务唤醒的任务、等待 V 操作唤醒的任务。最终的分类 TCB 队列如图 4 所示。

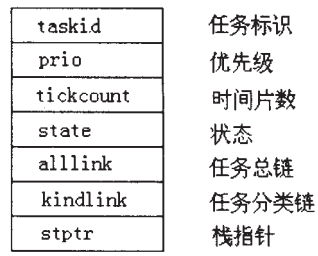

图 3

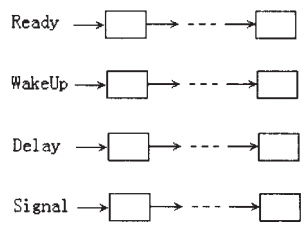

图 4

每当时间片中断发生时，就将运行状态的任务和 Delay 队列上的各任务的时间片计数减一，经减一操作后，把所有时间片计数为零的任务从 Delay 队列转换到 Ready 队列（如果该任务原来是运行状态，则直接加入 Ready 队列），并将其状态改为 Ready。

当有任务调用 wakeup（id）原语时，由调度程序在 wakeup 队列中查找是否有指定 id 的任务，如果有，则将其从原队列中删除，加入 Ready 队列，并将状态改为 Ready。

当有任务进行信号量的 V 操作时，信号量加一之后仍然不大于 0，则将 signal 队列的队首元素出队，加入 Ready 队列，并将状态改为 Ready。

每当有任务加入 Ready 链表时，首先将它们的时间片计数重新设置为预设值，然后再加入链表。每当有新任务加入时，在 Ready 链表中选择一个适当的位置，使得该 TCB 加入后，整个 Ready 链是按优先级从高到低的顺序排列，如果原链表中已经有和新 TCB 同优先级的元素，新 TCB 放在同优先级元素的最后。每当运行状态的任务释放出对 CPU 的控制权之后，总是把 Ready 链表的链首元素摘下来，置为运行状态后，运行该任务。

2.4 信号量

利用信号量，可以实现对临界资源的互斥访问、任务间的同步等。

该系统实现的是计数信号量，提供的操作有：信号量的初始化、P 操作和 V 操作。通过初始化，可以把信号量预置为某个值。当某个任务进行 P 操作时，过程如下：

```
void signal_p（）
{signal--；
 if（signal<0）
 {保存现场；
  将本任务加入 signal 队列；
  跳转到调度程序 scheduler；
```

（下转 128 页）

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

当前有效 CRL 表。

(6)系统内存,用于热缓存提高数据访问效率,主要内容包括:

根据优先算法选出的部分用户证书和系统当前有效证书;

当前有效 CRL 表。

CA 系统在各种操作和系统行为中必须保证和维护三者的一致性。

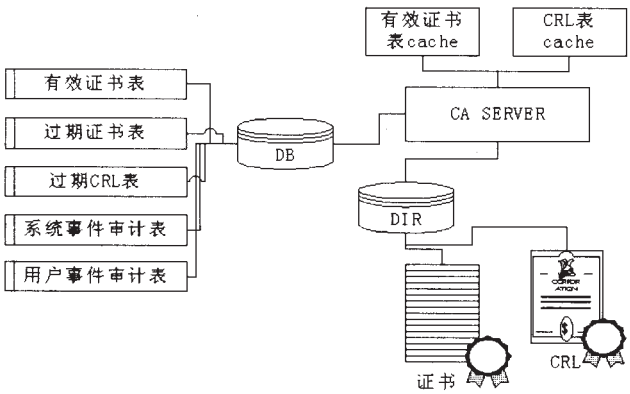

图 4 单结点系统主要数据存储分布图

4.4 认证系统的模块划分

公用模块的功能包括:

ASN。1 编码;密码算法;数据库接口;目录服务器接口。

证书基本功能模块内容包括:

证书的生成,证书删除,证书归档,证书冻结,证书解冻,证书发布,非顶级 CA 证书更新,从本地证书库中获取证书,从目录服务器中获取证书。

CRL 基本功能模块内容包括:

签发本地 CRL,在本地 CRL 中插入条目,从数据库获取本地 CRL,从目录服务器获取本地 CRL。

系统基本控制模块内容包括:

顶级 CA 系统密钥管理,CA 系统初始化,CA 系统开机恢复。

证书/CRL 综合功能模块内容包括:

证书的签发,证书的作废,证书的冻结,证书的解冻,证书的发布,用户证书更新,证书有效性检验,证书状态查询,从本地证书库中获取证书,从目录服务器中获取证书,CRL 管理。

策略管理模块的内容包括:

系统证书格式,证书的应用,算法的应用,附属层次模型内的命名空间的控制等。

在线支持模块的内容包括

在线申请证书,撤销证书,更新证书,验证证书,在线获得 CRL 等。

管理模块:负责整个系统的管理调度。

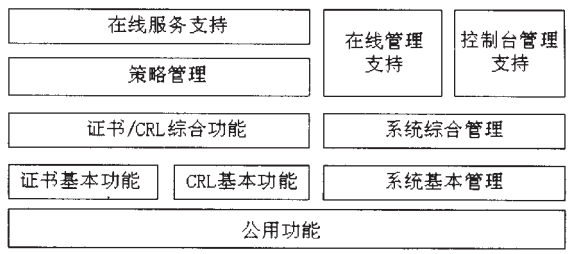

图 5 系统模块结构示意

5 总结

该文对附属层次型信任模型作了分析,并对其构建技术进行了分析,给出了一个基于附属层次型信任模型的可行有效的实现——RealCert 认证系统,该系统可以很好地解决电子商务和电子支付中的多级信任问题。 (收稿日期:2001 年 7 月)

参考文献

1.William E Burr.Public Key Infrastructure Technical Specification:Part C-Concept of Operations[S].1998

2.Understanding Public-Key Infrastructure:Concept,Standard,and Deployment Consideration[S].Carlisle Adams Steve Lloyd:Macmillan Technical Publishing,1999

3.W Ford,R Housley,W Polk et al.Internet X.509 Public Key Infrastructure Certificate and CRL Profile[S].RFC 2459,1999

(上接 124 页)

```
  }
}
```

当某个任务进行 V 操作时,过程如下:

```
void signal_v ()
{signal++;
 if (signal<=0)
 {将 signal 队列的队首元素出队;
  将其加入 Ready 队列;
 }
}
```

当使用信号量来控制共享资源的使用时,若用户程序设计不当,存在着死锁的可能。该系统不提供任何死锁的预防、检测和恢复措施,需要由用户在设计应用程序时仔细考虑。

2.5 任务间的通信

为了减少系统对资源的占用,该系统并未提供任务间的直接通信手段。但由于各任务都可以直接访问物理内存,所以如果需要在任务之间传递数据,可借助于全局变量或者指定的内存来实现。

2.6 中断

在系统中,中断不隶属于任何一个任务。每当中断发生时,由中断服务程序先作快速预处理,再唤醒某个进程以作进一步处理。这样就可以使得中断服务程序足够地短,同时又保证了数据的及时处理。

3 结束语

在笔者开发的专用收银机等产品中,使用了该内核,实际运行情况很好。既在一定程度上满足了实时性要求,又实现了任务的并发执行,而且简化了程序设计,程序模块分明,可以很容易地扩展新任务。 (收稿日期:2001 年 12 月)

参考文献

1.刘腾红.计算机操作系统[M].北京:科学出版社,2000

2.许俊,许克平.在 51 系列单片机上实现非抢先式消息驱动机制的 RTOS[J].单片机与嵌入式系统应用,2001;(6):7~11

3.庞丽萍.操作系统原理[M].武汉:华中理工大学出版社,1994

4.李华.MCS-51 系列单片机实用接口技术[M].北京:北京航空航天大学出版社,1993

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