文章编号: 1671 - 7147(2005)04 - 0375 - 03

一个嵌入式实时操作系统的设计

汪建新, 潘雪增*
(浙江大学 计算机科学与技术学院, 浙江 杭州 310027)

摘 要: 为了开发自主产权的嵌入式实时操作系统, 结合 uCLinux 内存管理的特性和 RTAI 的实时特性, 在总体架构、内存管理、进程管理、异常中断处理和定时器等主要模块上做了精心的设计, 开发出了具有硬实时特性的嵌入式实时操作系统——ZDRTOS.

关键词: 浙江大学实时操作系统; 实时; 嵌入式; 实时应用接口; 微控制 Linux 操作系统
中图分类号: TP 39 文献标识码: A

Design of an Embedded Real Time Operating System

WANG Jian xin, PAN Xue zeng*
(College of Computer Science and Technology, Zhejiang University, Hangzhou 310027)

Abstract: To develop an embedded real time operating system with own property rights, we combined memory management characteristic of uCLinux with real time characteristic of RTAI. With elaborate design in main modules such as architecture, memory management, process management, exception and interrupt handling and timer, we eventually made a hard real time embedded operating system——ZDRTOS.

Key words: ZDRTOS; real time; embedded; RTAI; uCLinux

嵌入式系统大多工作在对实时性要求很高的环境中, 需要一个支持实时多任务的操作系统(RTOS)内核. 目前, 中国大多数嵌入式软件开发还处在基于处理器直接编写的阶段, 尚无采用商品化的 RTOS. 国外有代表性的嵌入式操作系统有 Vxwork, QNX, Lynx, Windows CE 等, 国内嵌入式操作系统的研究相对滞后, 比较有名的仅有凯思集团自主研制的 Hopen OS[1]. 因此, 开发自主产权的 RTOS 则极具挑战性. 典型的 RTOS 结构见图 1.

为了开发自主产权的 RTOS, 作者开发出了嵌入式实时操作系统 ZDRTOS.

1 总体结构

ZDRTOS 是针对 ARM946ES 芯片设计的. ARM946ES 芯片不带 MMU, 只有一个简单的内存保护单元(MPU), 所以作者把具有硬实时特性的 RTAI 和针对 NOMMU 设计的嵌入式操作系统 uCLinux 相结合, 使 ZDRTOS 既具有 RTAI 的硬实时特性, 又不失 uCLinux 针对 NOMMU 内存管理的特性, 而且 uCLinux 基本是承自标准的 linux 的. 所以, ZDRTOS 为用户提供熟悉的 API, 有利于应用程序的快速开发. ZDRTOS 的体系结构见图 2.

收稿日期: 2004 - 09 - 30; 修订日期: 2004 - 10 - 20.
作者简介: 汪建新(1979 -), 男, 浙江杭州人, 计算机系统结构专业硕士研究生.
*通讯联系人: 潘雪增(1942 -), 男, 浙江杭州人, 教授, 博士生导师. 主要从事网络信息安全、嵌入式系统研究.
Email: xzpan@cs.zju.edu.cn

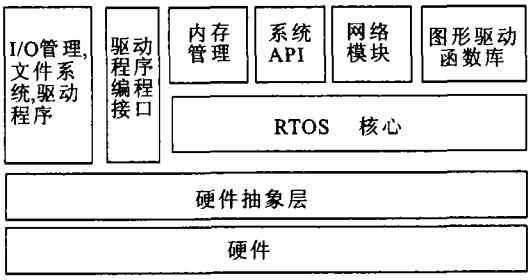

图1 标准RTOS的体系结构示意

Fig. 1 Architecture of standard ROTS

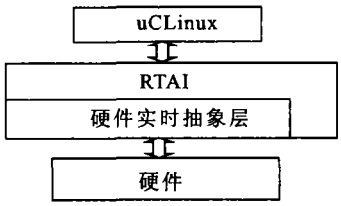

图2 ZDRTOS的体系结构示意

Fig. 2 Architecture of ZDRTS

图2中至关重要的就是精致的硬件实时抽象层(RTHAL),它截获所有的硬件中断并根据RTAI调度器的需要把中断重定向到uCLinux或者实时任务.被调度的实时任务的中断直接送向这个任务,不需任何实时任务调度的中断直接送向uCLinux.在那里,根据需要它们得到相应处理[2].在这种方式下RTHAL提供了一个RTAI完全有能力先占用uCLinux的框架,确保了系统的硬实时性.

2 内存管理和进程管理

2.1 内存管理及其模型

内存管理是嵌入式操作系统设计的重点,ZDRTOS的内存管理模型见图3.

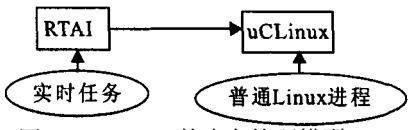

图3 ZDRTOS的内存管理模型

Fig. 3 Model of memory managent of ZDRTOS

由图3可见,普通的linux进程对内存的请求通过uCLinux实现,实时任务对于内存的请求则是由RTAI的动态内存管理模块实现,而RTAI中分配给实时任务的内存也是由uCLinux的底层函数申请得到的.

2.1.1 *动态内存管理机制* ZDRTOS对实时系统动态内存管理的方法是:在RTAI的动态内存管理模块被加载时,uCLinux内核就预分配了几个内存块(chunk)给它,分配给实时任务的内存(block)便是在这几个chunk中分配的.同时,动态内存管理系统监控着该系统中的空闲内存,如果空闲内存低于"低水位"(空闲内存数低于该值,则当下一次分配内存时,就会面临内存不足的危险),就向系统递交分配一个新chunk的请求;如果空闲内存高于"高水位"(空闲内存数高于该值,则表明RTAI的动态内存管理系统占用了过多却不必要的物理内存),则向系统递交释放一个chunk的请求.这两个请求都是当实时系统空闲,并把控制权交给uCLinux内核前处理的.这种做法使得实时任务动态申请内存时不会导致系统阻塞,从而确保了ZDRTOS系统的实时性和准确性.

2.1.2 *动态内存管理机制的改进* 现在发布的RTAI中已经纳入了动态内存管理模块,但是现有的管理机制由于分配内存时产生的"空洞",会导致一种"假死"现象:一方面它不会向内核递交新分配chunk的系统请求,而另一方面随后分配内存块的请求都会失败.究其原因,是由于在NOMMU处理器的系统中不能使用非连续内存造成的[3].因为这些内存"空洞"之间是不连续的,所以不能把多个内存"空洞"分配给某一个实时进程.当然上述情景在实时任务申请小块内存时不会发生,但对于大块内存的申请(即使这种情况在现实中较少)会出现这种现象,所以ZDRTOS对此做了两个小改进.

1) 改进一:

改进前:if(系统中总的空闲内存<=低水位)
　　向内核递交分配新chunk的请求;

改成:if(系统中总的空闲内存<=低水位‖每一个chunk的空闲内存<请求内存)
　　向内核递交分配新chunk的请求;

改进后,如果连续多次分配大块内存,当某一次失败后,紧接着下一次分配请求就会成功.

2) 改进二:

改进前:if(系统中总的空闲内存>=高水位)
　　向内核递交释放chunk的请求;

改成:if(系统中总的空闲内存>= 高水位 && 某个chunk完全空闲)
　　向内核递交释放chunk的请求;

改进后,系统就不会盲目地递交系统请求了,为控制权的交换节省了时间.

2.2 进程管理

2.2.1 *实时任务实现* ZDRTOS中的实时任务以线程的方式实现,运行在单一的核心地址空间,减少了上下文切换的开销,避免了存储保护级别的切换.RTAI使用可装载核心模块实现实时任务的动态创建和卸载.ZDRTOS中实时任务的上下文切换只需要做两项工作——整数寄存器压栈和改变栈

寄存器值, 与 CPU 的硬上下文切换相比大大节省了开销.

2.2.2 进程调度　ZDRTOS 对实时任务的调度采用先来先服务(FIFO)和时间片轮转(RR)的策略, 默认是 FIFO. 只要使宏 ALLOW_RR 生效, 就可以使用 RR. 另外还提供了单一比率调度算法(RMS)来判断任务集的可调度性. 比如, 可以让最短周期的任务使用最高的优先级, 以此类推, 只要对每个任务设置合适的优先级并配合使用 FIFO 调度算法即可. 此外还提供了最早时限优先(EDF)的调度算法, 使必须最早结束的任务获得率先执行的权力. 在 ZDRTOS 中, 所有的算法都是可以混合使用的[4].

ZDRTOS 把 uCLinux 作为实时核心的一个优先级最低的进程同实时进程一起调度, 当实时系统空闲时才会得到执行, 并且它可以被实时进程所抢断. 当调度周期设定太小或者周期任务过于繁重时, uCLinux 将得不到执行[5], 所以要选取合适的调度周期. 经过测试, 当 TICK_PERIOD 设为 10e5 时, uCLinux 得不到执行; 设为 5× 10e5 时, Linux 运行无误; 而 uCLinux 的时钟中断周期是 10e7, 所以实时核心的 TICK_PERIOD 值不得大于 10e7.

2.2.3 进程间通信　ZDRTOS 中可用进程通信机制有先进先出队列(FIFO)、邮箱(Mailbox)、消息队列(Message Queue)、共享内存(Shared Memory)等.

3　异常中断处理及定时器的设计

ARM 体系中的异常中断主要有复位异常、未定义指令异常、软件中断(SWI)异常、指令预取异常、数据访问异常、外部中断请求(IRQ)和快速中断请求异常(FIQ)等. 对于这些异常 uCLinux 已具备了一套完备的处理体系[6], 但是由于 ARM946ES 中 MPU 的存在, ZDRTOS 重新设计了指令预取异常和数据访问异常处理机制. 这两种异常大多是由于不满足 MPU 所设定的该指令或者数据所在内存的访问权限而引起的. ZDRTOS 的做法是: 当发生这两种异常时, 首先判断当前发生异常的进程是否是实时进程, 如果是, 由于其重要性, 关键资源不允许破坏, 则必须杀掉该进程并使系统停机等候处理; 如果不是, 只需杀掉该进程重新调度即可.

ZDRTOS 对于中断的处理机制是: 在中断控制器和 uCLinux 核心之间用一层软中断模拟器进行隔离, 所有的硬件中断都被中断模拟器接受. 当关中断时, 模拟器只是清软中断标识; 当中断发生时, 不管软中断标识为何值, 中断处理函数被立即调用, 因此可以保证实时中断不被阻塞并有最小的中断延迟时间. 经测试, ZDRTOS 的中断响应时间约为 15 μs, 可完全满足硬实时操作系统的标准.

在 ZDRTOS 中, 硬件定时器的周期被编程为 100 Hz, 任务调度的最小周期为 10 ms[7]. ZDRTOS 采取两种定时方式: 一种为周期性定时器(Periodic timer), 设定成的周期时间不变, 每到一周期就产生一次时钟中断; 另一种是一次性定时器(Oneshot timer), 每次时钟中断的时间是根据当前的系统时间和每个实时任务的运行周期决定下次时钟中断的间隔. 由于 ARM946ES 是 counter down 的计数器[8], 所以采用软件方式保存 TSC.

4　结　语

作者在开发过程中主要负责内存管理模块的设计, 由于发现 RTAI 原有内存管理模块的缺陷, 不能满足既定要求, 所以改进了 RTAI 的动态内存管理模块, 改善了实时系统的性能. 最终开发出 ZDRTOS 的性能可以与最好的商业实时操作系统竞争, 它提供 4 μs 典型的任务切换时间、15 μs 的中断响应时间、10×10^4 Hz 的周期任务率和 3×10^4 Hz 的一次性任务率, 而且还提供了现代操作系统方便的环境和强大的功能, 如网络服务、GUI 系统、C/C++/Java 开发工具和标准的编程接口, 有利于应用程序的快速开发. ZDRTOS 完全可以应用在对实时性要求很高的环境中.

参考文献:

[1] 邹思轶. 嵌入式 Linux 设计与应用[M]. 北京: 清华大学出版社, 2002. 5 – 305.

[2] 毛德操, 胡希明. 嵌入式系统[M]. 杭州: 浙江大学出版社, 2003. 275 – 408.

[3] Micbael Barr. Programming Embedded Systems in C and C++[M]. 北京: 中国电力出版社, 2001. 65 – 189.

[4] 杜春雷. ARM 体系结构与编程[M]. 北京: 清华大学出版社, 2003. 165 – 230.

[5] 吴明辉. 基于 ARM 的嵌入式系统开发与应用[M]. 北京: 人民邮电出版社, 2004. 89 – 145.

[6] 周立功. ARM 微控制器基础与实战[M]. 北京: 北京航空航天大学出版社, 2003. 206 – 215.

[7] 慕春棣. 嵌入式系统的构建[M]. 北京: 清华大学出版社, 2004. 23 – 54.

[8] Qing Li. Real Time Concepts for Embedded System[M]. 北京: 北京航空航天大学出版社, 2004. 32 – 87.

(责任编辑: 杨勇, 彭宇敏)