

嵌入式操作系统的硬实时微内核设计

郭 杨¹, 王新社²

(1. 中国科学院研究生院, 北京 100080; 2. 中国科学院软件中心, 北京 100080)

摘 要 :目前的嵌入式实时操作系统存在着内核结构较为复杂、臃肿、稳定性不高、对硬实时应用支持不足等问题。针对这些问题,结合现有的操作系统内核理论及嵌入式实时系统的特殊需求,通过组件化的设计,将内核划分为核心态基本模块及用户态扩展模块,提供多种进程间通讯(IPC)方式,并引入独特的硬实时抢占式调度机制,设计出一种精炼、稳定的硬实时微内核。通过这种设计思路成功开发出了昊鹏(Hopen)操作系统新版内核,应用于最新的 3G 手机产品上,取得了非常好的效果。

关键词 :嵌入式实时操作系统; 硬实时; 组件化; 抢占式调度; 微内核; 昊鹏

中图分类号 :TP316.2 **文献标识码** :A **文章编号** :1000-7024(2008)01-0115-04

Design of hard real-time micro-kernel in embedded operating system

GUO Yang¹, WANG Xin-she²

(1. Graduate University, Chinese Academy of Sciences, Beijing 100080, China;

2. Software Center, Chinese Academy of Sciences, Beijing 100080, China)

Abstract : Many embedded real-time operating systems have some problems such as the struction of kernel is somewhat complex, over-staffed, poorly stable, not properly fit for hard real-time applications and so on. According to the theories of operating system kernel and special requirements of embedded real-time system, a hard real-time micro-kernel is proposed, which through modular design includes basic modules in kernel mode and extended modules in user mode, provides various interprocess communication mechanisms and introduces a special hard real-time preemptive scheduling strategy. The new Hopen operating system upgraded with this kernel is implemented in new 3G mobile phones and excellent results are obtained.

Key words : embedded real-time operating system; hard real-time; modularization; preemptive scheduling; micro-kernel; Hopen

0 引 言

嵌入式系统是以应用为中心,软硬件可裁减的,适用于对功能、可靠性、成本、体积、功耗等综合性严格要求的专用计算机系统。它具有软件代码小、高度自动化、响应速度快等特点,特别适合于要求实时和多任务的体系。嵌入式实时操作系统是一种实时的、支持嵌入式系统应用的操作系统软件,它是嵌入式系统重要的组成部分。与通用操作系统相比较,嵌入式操作系统在系统实时高效性、硬件的相关依赖性、软件固态化以及应用的专用性等方面具有较为突出的特点^[1-2]。

嵌入式系统的发展经历了早期的无操作系统的嵌入式算法阶段、简单监控式的实时操作系统阶段直至今天的通用嵌入式实时操作系统阶段。这一阶段系统的特点是能运行在各种不同类型强大的微处理器上;具有强大的通用型操作系统的功能,如具备了文件和目录管理、多任务、设备支持、网络支持、图形窗口以及用户界面等功能;具有大量的丰富的应用程序接口(API)和嵌入式应用软件。系统的复杂化以及应用的多

样化带来了诸多问题,如内核结构臃肿、效率低、功耗大,对实时、非实时混合任务的支持不足,特别是无法很好的保证硬实时任务的应用。

本文主要介绍一种应用于嵌入式实时操作系统中的可抢占式微内核的设计。它以传统操作系统理论为基础,结合嵌入式实时的特性,对现有同类产品的不足进行了改进和提高。通过探索和实践,设计开发出了 Hopen4.0(我国自主研发的嵌入式实时操作系统)^[1]可强占式微内核,使得新版系统在稳定性、功效、硬实时性等多个方面有了很大的提高。

1 各种嵌入式实时操作系统的比较

目前比较具有代表性的嵌入式实时操作系统主要有: Windows CE^[3]、嵌入式 Linux^[4]、VxWorks^[5]以及 μ C/OS^[6]。

Windows CE 是一种针对小容量、移动式、智能化和模块化要求的嵌入式软实时操作系统。它是从整体上为有限资源的平台设计出的完整优先级、多任务、多线程的操作系统。但是它在效率、功耗方面的表现并不出色,并且无法很好的支持

收稿日期: 2007-01-11 E-mail: poplargo@hotmail.com

基金项目: 2006 年信息产业部电子发展基金项目(信部运[2006]634)。

作者简介: 郭杨(1982-),男,河北秦皇岛人,硕士研究生,研究方向为嵌入式操作系统、实时操作系统;王新社(1954-),男,江西南昌人,研究员,研究方向为操作系统、软件工程和计算机网络。

硬实时应用。

嵌入式 Linux 是开源项目,有着优秀的网络功能,稳定,内核精悍,运行所需资源少,支持的硬件数量庞大。但是 Linux 体系提供实时性能需要添加实时软件模块,而这些实时模块是在内核空间运行的,因此代码错误可能会破坏操作系统从而影响整个系统的稳定性和可靠性。

VxWorks 具有可裁剪的微内核结构,效率高、移植性好,具备完整的 TCP/IP 网络协议,是一个嵌入式硬实时操作系统。总的来说,VxWorks 在很多方面做得很好,但是某些方面仍有改进的余地,比如线程调度,现有的调度策略比较简单,可以增加稍微复杂一些的却更适合硬实时特性的调度策略。

μC/OS 是专为嵌入式应用设计的开源项目,它同样具有可移植,可固化,可裁剪,实用可靠等特点。但是由于它只是一个实时内核,只提供了内核的基本功能,诸如文件系统、网络管理等额外功能则需要用户自己去添加实现。

从以上的比较可以看出,现有的嵌入式实时操作系统都或多或少的存在着一些缺陷,仍有很多值得改进的地方。能否在充分吸收各自优点的基础上,尽可能的改善现有的不足,使得系统更加完善,设计是关键。

2 可抢占式硬实时微内核的设计

通过组件化的设计,内核不仅实现了一个硬实时系统所需要的一切基本要素:多任务、优先级驱动的抢占式多种调度策略和快速现场切换、完善的中断机制^[2,7],而且提供了诸如文件系统、网络服务等可选功能组件。在提供强大完善功能的同时,内核还具有开销小、效率高、稳定健壮、对目标机的系统资源要求较低、支持多种处理器等特点。

2.1 微内核的系统结构

微内核是一个小型的操作系统核心,它为模块化扩展提供了基础。微内核的基本原理是:只将最基本的操作系统功能放在核心态的内核中,其它的服务和应用程序在微内核之上构造,运行在用户态。微内核结构用一个水平分层结构代替了传统的纵向分层结构,在微内核外部的操作系统部件被当作服务器进程实现,它们之间可以借助于微内核的通讯机制来实现交互^[7]。本文介绍的微内核是一个硬实时多任务抢占式微内核,其系统结构如图 1 所示。采用组件化的设计思路,将内核划分为非组件部分和组件部分。这样设计一方面增强了内核的可扩展性和灵活性,另一方面也给内核各个模块在开发和测试的过程中带来了方便,从而在整体和局部上更好的保证了系统的可靠和稳定。

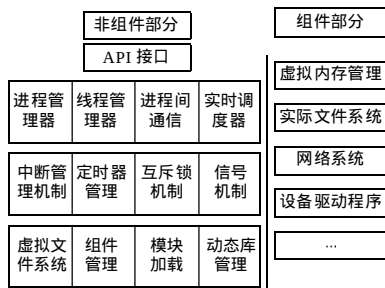


图 1 微内核的系统结构

2.2 内核非组件设计

非组件部分是构成微内核的最小集合。这部分模块始终运行在核心态,它们之中有的功能上依赖于硬件,有的提供基本服务和机制,有的支持组件化和模块化管理。为了使文章结构更加精炼和突出重点,以下只针对一些较为重要或是具有特色的功能模块进行介绍,而对于其它部分或是穿插在其它部分中进行说明或是略过说明,不做说明的部分设计上可参考其它的嵌入式实时操作系统的通用实现。

2.2.1 进程间通讯

内核提供了 4 种进程间通讯(IPC)的机制:管道(Pipe)、信号量(Semaphore)、消息队列(Message Queue)及信箱(Mailbox)机制。其中前 3 种机制是可移植操作系统接口标准(POXIS)中定义的进程间通讯机制^[7],由于这 3 种 IPC 机制的局限性,及对于通讯手段实时高效的要求,这里特别设计了一种信箱机制。

信箱机制是一种高效实用的机制,是内核用于进程间通讯的主要手段,它综合了 POSIX 的管道以及消息队列等特点,是一种适合在“消息驱动形式”的程序中进行进程间通讯的便利方法。它具有双向信息传输功能,而管道和消息队列只能单向传输;它可以进行一点对多点的信息传输,而管道和消息队列只能进行点对点的传输。

Mailbox 的设计与真实世界中的信箱非常相似,每个进程都可以拥有一个自己的信箱用来与其它进程通讯,接收和发送的信息是以 mail 为单位的数据包,发送时需指明接收方信箱 ID,接收方在信封上可以得到发送方地址和发信时间等等。当然除此之外还有一些不同点,如发送操作直接将信件投递到接收方的信箱中,并且需要等到接收方接收后才返回;如果收件人地址不存在,系统立刻拒绝投递信件;如果系统不太繁忙,信件从发送到接收之间的延迟小到几乎可以忽略。

信箱分为有名信箱和无名信箱两种。有名信箱是指在系统的 IPC 机制中注册了信箱名字的信箱。这种信箱一般用于系统中的各种服务器程序。非服务器程序的信箱一般是无名信箱,这种信箱的 ID 号不对外公开。其它程序只有通过接收到的信件的信封上才能得到该信箱的 ID 号。

2.2.2 实时调度器

一个实时系统性能的好坏很大程度上取决于实时调度算法的选择。为了满足更广泛的各种实时和非实时任务的需求,以及更有效的协调各种不同任务以最大限度的提高系统性能,内核提供了丰富的线程调度机制。线程调度器是一个可以替换的核心组件,在最简单的固定优先级调度机制基础上,实现 FIFO 调度(类似 POSIX 中的 SCHED_FIFO)、时间片轮转调度(类似 POSIX 中的 SCHED_RR)以及混合时限(Deadline)调度等多种调度策略^[7]。

这里特别说明一下混合时限调度,这是为了更好的支持硬实时需要专门设计的一种调度策略。目前绝大多数的嵌入式实时操作系统的调度策略实现相对比较简单,简单的优先级调度结合相同优先级上的时间片轮转调度。使用混合时限调度策略后,系统通过增加少量的开销,充分的区分考虑了包含硬实时任务在内的各种任务的特点,针对性的统一调度,提升了多种任务并存时的系统整体性能,扩展了系统的适用范围。

混合时限调度可以将运行在系统中的硬实时任务、软实

时任务和非实时任务(交互任务、批处理任务等)统一调度,调度策略既满足了所有硬实时任务的时限要求,最大限度的按时完成软实时任务,又最大程度上缩短了非实时任务的响应时间和周转时间。

通过在 EDF(early deadline first)调度算法^[2]的基础上加以改进,对于每一个非实时任务,系统创建一个周期性服务器来周期性的执行,它们连同所有的实时任务统一的按照 EDF 调度。为了更进一步的提升系统,调度策略中还引入空闲回收以及周期性服务器参数自适应动态调整等特别机制^[6]。

2.2.3 中断管理机制

实时内核的中断处理程序可以分为低级中断处理和高级中断处理。低级中断处理即中断入口程序,中断发生后,经过核心的中断分配器直接进入的程序。由于进入中断入口程序后,处理器处于关中断状态,因此这段程序的执行时间应控制在 10us 内,只用来处理最紧急的 IO 操作,而将不太紧急的操作放在高级中断处理程序中。高级中断处理即中断处理程序,处理中断入口程序没有处理完的 IO 操作。中断入口程序执行最紧急的 IO 操作后激活中断处理程序。所有的高级中断处理程序在系统中一个高优先级线程中执行,按照 FIFO 方式调用。高级中断处理程序不允许进入阻塞状态,可以等待互斥锁,其前提是该互斥锁是该设备驱动程序独立使用的,并且所有程序在拥有该互斥锁的期间不会进入阻塞状态。

中断服务线程是在必要时替代高级中断处理程序的核心线程。例如当一个设备驱动程序的中断处理比较复杂(如网卡),可以使用中断服务线程来代替高级中断处理程序,在驱动程序初始化时由驱动程序的初始化过程创建,其优先级可以根据需要选择一个适当的数值。中断服务线程处理完中断后,进入阻塞状态,等待下一个中断,并由中断入口程序在适当的时候将其唤醒。

2.2.4 定时器

实时内核提供普通精度和高精度两种定时器。

普通精度定时器的精度为 1~10 ms。定时器靠硬件的定时器中断运转,系统中实现的软件定时器的个数没有限制。定时器的种类包括一次性定时器和周期性定时器。为了达到较高的效率,定时器管理器采用将定时器定时时间散列到 3 个(或多个)用于存放不同定时器的定时器环中的方法,将定时器操作的时间控制在常数的时间内。

高精度定时器的精度要视硬件的精度而定,如果硬件的定时精度小于 1 us,则为 1 us,否则为硬件的定时精度。

高精度定时器和普通精度定时器的区别除了精度不同之外,还在于调用定时器程序时机的不同。高精度定时器的定时器程序在时钟中断处理程序中被调用,普通精度定时器的处理程序则在时钟处理线程中被调用。中断处理程序的优先级具有高于所有线程的优先级,时钟处理线程的优先级则是在系统初始化时被指定的,它一般应具有中等的优先级。

2.2.5 组件管理

组件管理是用来支持内核组件化结构的功能模块,核心内部服务不是组件,而是所有组件的支持平台。内核不需要支持组件的动态加载,所有的组件可以在配置核心时进行选择,选择完后将所有被选入核心的组件与核心内部服务一起连接

成一个可执行的映像。

核心的组件管理比较简单,只需要提供组件的注册和发现机制即可。每一个被连接到系统中的组件在初始化过程中进行注册。每一个组件可以提供若干个接口程序表,系统其它部分通过调用接口程序表中的程序获得组件的服务,组件接口程序表的格式和内容必须根据该组件提供的服务类型规定。

2.3 内核组件设计

非组件部分只是提供了嵌入式实时系统的基本模块,一个功能强大的内核应该还要提供诸如虚存管理、文件系统、网络服务等高级模块。内核组件可以采取较为灵活的设计思路,它们可以以静态库或是用户态进程等方式存在,通过合理的搭配可以同时确保系统的高效和稳定。

2.3.1 虚拟内存管理

内核非组件部分没有专门设置低级存储管理模块,原因是核心服务不使用动态内存分配策略,各种数据结构需要的内存存在初始化过程中一次性全部分配到位。这些数据结构可以包括进程线程数,最大打开文件数,IPC 对象数,IPC 通讯使用的内存数等。但是核心支持虚拟存储管理,由于硬实时的特点,虚存管理一方面要尽量控制相关操作的响应时间,另一方面可以简化或去除传统内核虚存管理的一些功能,如内外存交换,动态页面分配等,每一个应用程序只有在得到它正常运行所需的全部内存页面后才能正常工作。

在整个 4 G 的虚拟地址空间中,每一个进程各自占用自己私有的 32 M 地址空间,而不是占用除去核心之外的全部 3 G 空间,这样做的好处是进程间切换时不需要做页表和高缓存的切换,提高了进程间切换的速度,保证实时性。进程的地址空间从低到高依次为保护地址、代码段、数据段、动态内存堆、堆栈段、环境和参数、动态库数据段以及 IO 端口映射段。每一个段的内存都是在进程加载时分配好的,这样同样是为了保证进程的实时特性。如果在进程运行过程中出现缺页中断,则认为是应用程序的错误,系统不会自动扩展堆栈段。

无论在核心中还是在用户程序中都提供实时内存管理和非实时内存管理两种内存管理模式。实时内存管理的内存分配与释放具有可控的最大执行时间。实时内存管理通过管理应用程序创建的内存池进行。每一个内存池中可供动态分配的内存块的大小是一个常数,该常数在创建内存池时给定。内存池可以指定是否支持多线程访问。如果内存池支持多线程访问,则该内存池将使用互斥锁来进行数据保护。如果该内存池不允许多线程访问,则不进行数据保护,执行内存分配与释放需要的时间将大大减少。

非实时内存管理提供的操作是 ANSI C 中的内存分配与释放操作。采用可以尽量减少内存碎片发生的内存分配方法,并且提供接近于常数的内存分配与释放操作需要的时间。

2.3.2 文件系统

文件系统是由核心文件系统和用户态文件系统组成,文件系统的体系图如图 2 所示。核心文件系统由虚拟文件系统和处于核心中的一些实际文件系统组件组成。用户态文件系统是由处于用户态的文件服务器进程和文件服务守护进程组成。

用户态的文件系统程序只对文件系统的结构进行解析,不会造成文件操作速度的降低和数据复制次数的增加。文件

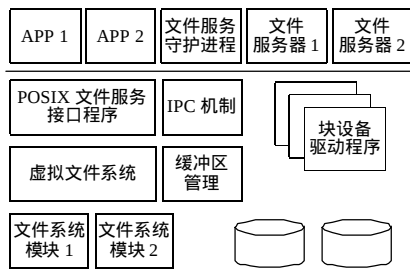


图 2 文件系统体系

数据的读写不通过 IPC 机制实现,而是由核心的文件系统代理直接调用块设备驱动程序,将文件的内容从存储介质直接传输到用户地址空间(或从用户地址空间传输到存储介质中)。

核心通过虚拟文件系统 rootfs 实现文件系统的加载,实际文件系统只能加载到 rootfs 的一级目录上。文件系统加载时,首先查找核心模块支持的文件系统表,如果是核心文件系统模块支持的文件系统,则通过组件接口调用核心的文件系统模块执行加载操作。否则,将加载请求构造成一个 IPC 请求包,向文件服务守护进程发送加载请求并等待回环。文件服务守护进程收到请求后,查找注册表文件,找到支持该文件系统的程序,创建一个新的文件服务器进程用来和核心以及文件守护进程交互。

进行文件操作时,首先通过根据文件描述符查找进程控制块中的文件描述符表找到相应文件的文件表,再通过文件表中的文件系统操作指针调用相应操作对文件进行操作。

2.3.3 网络系统

内核提供 TCP/IP 服务,TCP/IP 协议栈是作为一个用户态进程运行的,它通过 IPC 机制同其它进程进行通讯,通过调用底层 MAC 驱动程序提供的接口实现网络服务操作。整个 TCP/IP 服务进程的体系结构如图 3 所示。

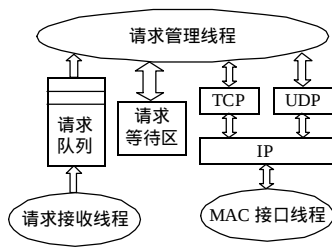


图 3 TCP/IP 服务进程的体系结构

请求接收线程通过 IPC 机制从其它进程接收网络服务请求包,放入本进程的请求队列中。请求管理线程处理请求队列中的请求,如请求包可以立即完成则立即完成,如不能立即完成则放入请求等待区中。请求包经过一系列的处理后,得到处理结果,再通过 IPC 机制发送给发送进程。

3 结束语

本文介绍了一种嵌入式实时操作系统内核的设计。采用组件化的设计方法,对内核功能进行细致划分和取舍,无论从规模上还是逻辑上都称得上是真正意义上的微内核,极大的提高了系统的灵活性、可扩展性、稳定性以及可靠性。设计上各个环节均充分考虑了实时性的要求,并且引入了具有特色的硬实时调度器,使得系统可以更好的支持硬实时应用以及多种任务的混合应用环境,扩展了系统的适用范围。

目前这种内核已经应用于 Hopen 操作系统中,新版的 Hopen 操作系统是一个嵌入式硬实时操作系统,它已经广泛的应用于多个领域之中,如 3G 手机产品等,无论从系统的功能、效率还是稳定性来说,都取得了非常好的效果。

参考文献:

- [1] 钟锡昌,张倪.嵌入式软件与 Hopen 操作系统[M].北京:北京航空航天大学出版社,2004.
- [2] Liu Jane W S. Real-time systems [M]. America: Prentice Hall, 2000.
- [3] 何宗键.Windows CE 嵌入式系统[M].北京:北京航空航天大学出版社,2006.
- [4] 邹恩轶.嵌入式 Linux 设计与应用[M].北京:清华大学出版社,2002.
- [5] 王金刚.基于 VxWorks 的嵌入式实时系统设计[M].北京:清华大学出版社,2004.
- [6] Jean J Labrosse.嵌入式实时操作系统μC/OS-II[M].邵贝贝,译.2 版.北京:北京航空航天大学出版社,2003.
- [7] William Stallings.Operating systems:Internals and design principles[M].4th ed.America:Prentice Hall,2004.
- [8] Scott Banachowski,Timothy Bisson,Scott A Brandt.Integrating best-effort scheduling into a real-time system[C].25th IEEE International Real-Time Systems Symposium.Lisbon:IEEE Computer Society Press,2004:139-150.

(上接第 11 页)

- [2] Cao J G, James E F, Younan N H. An image-adaptive watermark based on a redundant wavelet transform [J].IEEE Int Conf Image Processing,2001,2:277-280.
- [3] Kenneth R Castleman. 数字图像处理[M].北京:电子工业出版社,2002.
- [4] 孟兵,周良柱,万建伟.基于维纳滤波的数字水印算法[J].计算机工程与应用,2000,36(11):96-98.
- [5] Lu C S, Liao H Y M. Multipurpose watermarking for image authentication and protection [J]. IEEE Transactions on Image Processing,2001,10(10):1579-1592.
- [6] Hsieh Ming-Shing, Tseng Din-Chang. Hiding digital watermarks using multire solution wavelet transform [J].IEEE Trans Industrial Electronics,2001,48(5):875-882.
- [7] Patrick Loo. Digital watermarking with complex wavelet [D]. Doctor University of Cambridge,2002.
- [8] Meerwald Peter, Uhl Andreas. A survey of wavelet-domain watermarking algorithms [C]. San Jose,California: Proceedings of SPIE, 2001:181-190.
- [9] Kaew Kamnerd N, Rao K R. Wavelet based image adaptive watermarking scheme[J].Electronics letters,2000,36(2):235-238.