RT Embedded http://www.kontronn.com

嵌入式 Linux 实时性方法

胡志刚[1],MAHAMMED M E[1],余正军[1],吴运星[2]

(1. 中南大学 信息科学与工程学院,湖南 长沙,410083;
2. 中南大学 机电工程学院,湖南 长沙,410083)

摘要:针对 Linux 进程调度策略存在中断封锁时间过长、非抢占式的 Linux 内核,以及耗尽式的、机会均等的调度方式这 3 个不利于实现实时性的不足,提出提高嵌入式 Linux 实时性的方法,实现实时 Linux 系统(RTLinux)的调度算法及其他部分功能,并采用 LMbench 测试系统对改进的 RTLinux 和通用 Linux 的上下文切换时间进行对比测试。研究结果表明,改进的 RTLinux 有效地提高了嵌入式 Linux 的实时性。

关键词:嵌入式 Linux; 实时; 进程调度; 内核

Research on real-time method of embedded Linux

HU Zhi-gang[1], MAHAMMED M E[1], YU Zheng-jun[1], WU Yun-xin[2]

(1. College of Information Science and Engineering, Central South University, Changsha 410083, China;
2. College of Mechanical and Electrical Engineering, Central South University, Changsha 410083, China)

Abstract: Based on the characteristic and the applied situation of embedded system, the Linux schedule strategy and the flowchart of the schedule function are analyzed thoroughly. Aimed at the three disadvantages in Linux task scheduling method, i.e. the time of the interruption is too long, the kernel is not preemptive and the schedule strategy is exhausted and has equal opportunity, a method is proposed to improve the real-time performance of the embedded Linux system. A real-time Linux (RTLinux) scheduling algorithm and part of RTLinux functions are designed and realized. By using the LMbench testing system, the context switching time of improved RTLinux and general Linux are tested and compared. The result shows that the offered method improves the real-time performance of the embedded Linux.

Key words: embedded Linux; real-time; process schedule; kernel

嵌入式 Linux 系统是电脑软件和硬件的综合体,它以应用为中心,以计算机技术为基础,软、硬件可裁剪,因而适用于实际应用中对功能、可靠性、成本、体积、功耗等有严格要求的专用计算机系统[1]。嵌入式 Linux 系统通常面向用户,面向产品,面向特定范围。当系统工作在内存容量小,无硬盘环境时,它能根据实际应用确定其功能;系统软件一般都固化在存储器芯片或单片机中,通常无法对其进行更新和改进[2]。在实际生产时,要求嵌入式 Linux 实时工作,但 Linux 实时性不强。基于 Linux 开放源代码,通过裁剪和改进来构造嵌入式 Linux 系统,有效提高 Linux 的实时性是顺利完成实时任务的关键[3,4]。为此,作者通过改进 Linux 进程管理来提高嵌入式 Linux 系统的实时性。

Created in Master PDF Editor

RT Embedded http://www.kontronn.com

1 Linux 进程管理模块分析

内核是操作系统的内部核心程序,它向外部提供对计算机设备的核心管理调用。现代操作系统主要采用 2 种体系结构:单一内核结构和微内核结构。单一内核结构吞吐量大,效率高,因此,适用于服务器领域。UNIX 和 Linux 都是采用这种结构,但它的实时性不强。微内核结构灵活,操作系统简单、清晰,在同样的硬件条件下,实时性远强于单一内核的实时性,因此,目前大部分嵌入式系统采用这种结构[5,6]。

Linux 有 2 种类型的进程:普通进程和实时进程。进程有 5 种状态:TASK-RUNNING(运行态),TASK-INTERRUPTIBLE(可中断睡眠态),TASK-UNINTERRUPTIBLE(不可中断睡眠态),TASK-ZOMBIE(僵死状态)和 TASK-STOOPED(停止状态)。为了确定进程状态,Linux 还加入了 13 种标志。进程间状态转换的关系是:对于处于运行或就绪的进程,当等待某资源有效时,就进入等待队列。若等待的资源为硬件资源,则转为不可中断状态;否则,转为可中断状态。对于处于不可中断状态的进程,当硬件资源有效时,就将其唤醒为就绪状态;当资源有效,或者定时信号、其他进程发送的信号到达时,转换为就绪态;当进程运行结束时,进入僵死态;当调试进程或跟踪进程时,进程进入停止状态。

Linux 中断频率为 100 Hz,由于其中断封锁时间比较长,因而中断频率不高。因此,可通过提高时钟中断频率来提高 Linux 系统实时性[7]。Linux 的调度程序同时负责处理实时进程和普通进程,这种机会均等的调度策略减弱了系统的实时性。此外,Linux 的内核是非抢占的,永远处于运行状态,这也是其实时性较弱的原因之一。

Linux 调度程序的主要功能如下:

a. 调度程序运行 bottom half 并处理系统的调度任务队列。

b. 在选择另一个进程之前必须处理当前进程。

c. 若当前进程的调度策略是轮转策略,则将其放到运行队列的最后。

d. 若任务状态是可中断状态而且它在上次调度时收到一个信号,则它的状态转为就绪状态;若当前进程超时时,则它的状态也转为就绪态;若不是就绪态或可中断状态的进程时,则从运行队列中删除。

e. 进程选择。调度程序查看运行队列中的进程,查找最满足运行条件的进程。

f. 实施进程交换。若最满足运行条件的进程不是当前进程,则挂起当前进程,运行新进程。

g. 在调度的最后,完成进程的上、下文交换[11]。

Linux 存在 3 个不利于实现实时性的不足:过长的中断封锁时间,非抢占式的 Linux 内核,耗尽式的、机会均等的进程调度策略。为此,针对这些不足,通过提高中断频率,设计实时性内核来提高嵌入式 Linux 的实时性。

2 RTLinux 的设计与实现

2.1 RTLinux 的内核原理

对于实时内核,它始终不关闭硬件中断,可以接受所有的中断信号。当中断信号需要实时进程来处理时,实时进程将抢占 Linux 内核(在 RTLinux 中把原来的 Linux 内核作为普通进程对待,并将其优先级设置为最低)。若中断信号需要原来的 Linux 内核来处理,则由实时内核将信号传递给 Linux 内核。同时,在实时内核中提供标志字,来模拟原来 Linux 内核的关中断情况。这个标志字在 Linux 打开中断时置“1”,关中断时置“0”。实时内核在中断到来时检查标志字,若置“1”,则立刻将中断传给 Linux 内核;否则,将所有待处理的中断放入一个队列中,直到 Linux 打开中断时才将它们传给 Linux 内核[8]。

2.2 RTLinux 的系统结构

RTLinux 的系统结构如图 1 所示,针对 RTLinux 系统结构,提出 RTLinux 的分层模块结构,如图 2 所示。

其中各个模块的作用分别是:

a. rtl.o 是 RTLinux 最基本的 1 个模块,负责系统的中断处理,以及低层与硬件相关的代码。RTLinux 的其他所有模块都依赖这个模块。

b. rtl-time.o 模块用于控制处理器时钟,并生成一个用于操纵时钟的抽象接口。

c. rtl-sched.o 是基于优先级的实时调度器。

d. rtl-posixio.o 是支持 POSIX 标准的设备读写操作接口。

e. rtl-fifo.o 通过/dev/rtf * 设备实现实时任务和非实时任务间的数据通信。

2.3 RTLinux 的实现

2.3.1 时钟控制

RTLinux 对时钟的控制通过下面操作来完成。

a. clockid-t rtl-getbestclock(unsigned int cpu),找到与系统 CPU 相应的最适合实时任务调度的系统时钟。若是 SMP 系统,则使用本地 APIC 时钟;否则,使用 i8254 时钟。当 rtl-sched 模块初始化时,调用这个函数为相应的 CPU 获取系统时钟,该函数返回用于描述时钟的结构 rtl-clock。

RT Embedded http://www.kontronn.com

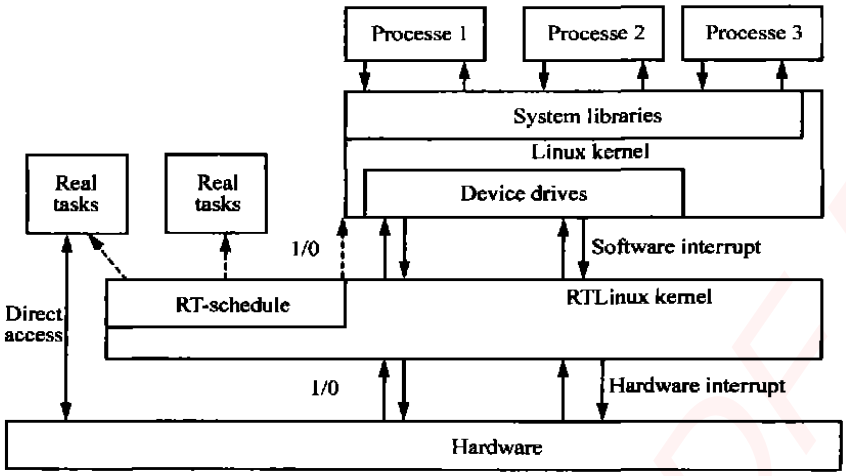


图 1 RTLinux 的系统结构

Fig. 1 System structure of RTLinux

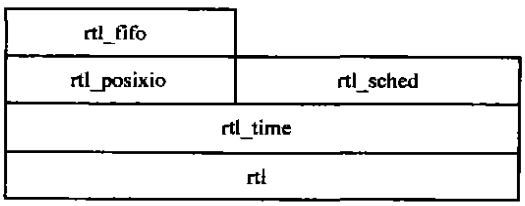


图 2 RTLinux 的模块结构

Fig. 2 Module structure of RTLinux

b. 2 个处理句柄用于设置与解除时钟。

int rtl-setclockhandler (clock-t h, clock-irq-handler-t fn);

int rtl-unsetclockhaldler(clockid-t h)。

从这组函数可看到,所有与时钟相关的操作都需要通过如下的数据结构 rtl-clock 实现:

```
struct rtl-clock
{ int ( *init) (struct rtl-clock *);
void ( *uninit) (struct rtl-clock *);
hrtime-t ( *gethrtime) (struct rtl-clock *);
int ( *sethrtime) (struct rtl-clock *,hrtime-t t);
int ( *settimer) (struct rtl-clock *,hrtime-t interval);
int ( *settimermode) (struct rtl-clock *, int mode);
clock-irq-handler-t handler;
int mode;
hrtime-t resolution;
hrtime-t value;
hrtime-t delta;
pthread-spinlock-t lock;
struct rtl-clock-arch arch;
};
```

rtl-clock 结构中包含时钟时间以及对定时器操作的函数入口。如 gethrtime 用来获取当前时间, settimer 用于设定定时器,settimermode 用于设定时钟模式。时钟的模式有 2 种:一次性模式和周期性模式[9,10]。当前的 RTLinux 调度程序中既提供周期性模式,也提供一次性模式。时钟模块的初始化在 schedulers/i386/rtl-time.c 中的 init-module() 函数中。模块的初始化完成下列工作:

a. 初始化 2 个与时钟相关的自旋锁。

b. init-hrtime()函数用于初始化与高精度时钟相关的信息。普通的 Linux 的时钟每隔 10 ms 产生 1 次中断,若要提高时钟精度就要提高频率,但这样会加重系统负担。RTLinux 采用自己独立的时钟模块而不是依靠 Linux 的时钟机制。在 RTLinux 中定义了数据类型 hrtime-t 来描述时间,以 ns 为单位。

c. 通过 rtl-create-clock-8254 () 函数对-i8254-clock 初始化,使-i8254-clock 中的各个函数指针指向实际的函数。在调度模块初始化时,执行 clock。用 init () 设置定时器。通常单 CPU 的处理系统大多使用-i8254-clock 时钟结构。

d. APIC (advanced programmable interrupt controller) 是 Pentium CPU 用来管理多处理器中断的一个控制器。APIC 的时钟精度比普通的标准时钟高,可以达到几微秒。SMP 主板都支持 APIC 时钟,而对于单处理器的主板则要根据具体情况而定。因此, rtl-create-clock-apic (cpu) 对变量-apic-clock [cpu]的初始化要根据具体的系统配置而定。

e. rtl-init-standard-clocks()。

2.3.2 实时调度

a. 实时调度模块的建立。实时调度模块的建立方法与普通 Linux 内核模块的建立方法一样,通过 init-module 函数来实现[12]。主要步骤如下:

RT Embedded http://www.kontronn.com

第 1 步　调用函数 ret = rtl-get-soft-irq (sched-irq-handler , "RTLinux Scheduler")。申请一个中断号,申请成功的返回值就是所申请的中断号,设备名为 RTLinux Scheduler。这样就将 RTLinux 的进程调度当作标准 Linux 的中断设备来处理,而在标准 Linux 的进程调度时首先要处理中断,通过这种方法来保证实时进程的优先运行。

第 2 步　调用函数 rtl-no-interrupts (interrupt-state)。先将中断信息保存在变量 interrupt-state 中,再关中断。

第 3 步　对系统中每个 CPU 所对应的结构 rtl-sched-cpu-struct 进行初始化,主要包括:

1) s- > rtl-current = &s- > rtl-linux-task。将当前进程指向 rtl-linux-task。

2) s- > rtl-tasks = &s- > rtl-linux-task。将普通 Linux 也当作一个实时进程,链到实时进程的任务队列中。

3) s- > rtl-linux-task. sched-param. sched-priority = - 1。将普通 Linux 的优先级赋为最小的优先级值(- 1)。

4) s- > clock = rtl-getbestclock (cpu-id)。通过 rtl-getbestclock 函数来找到与系统 CPU 相应的最适合实时任务调度的系统时钟。

第 4 步　调用函数 rtl-restore-interrupts (interrupt-state),恢复中断信息字,并开中断。

b. 实时进程调度。在所有与进程相关的函数中,最主要的是实时调度函数 rtl-schedule() ,在函数中采用的调度算法决定了进程调度的实时性。RTLinux 中采用的调度算法是 RMA (rate monotonic algorithm) ,有周期性 (periodic) 和一次性 (one shot) 2 种调度模式。实现调度的主要思路是:搜索实时任务队列,比较当前精确时刻与任务的恢复执行时间。若当前时刻超过了恢复执行时间,则恢复执行时间加上任务时间间隔,最后根据进程的优先级来决定下一个运行的进程。同时系统依靠时钟模块的定时器功能,完成进程抢占。函数实现的流程图如图 3 所示。

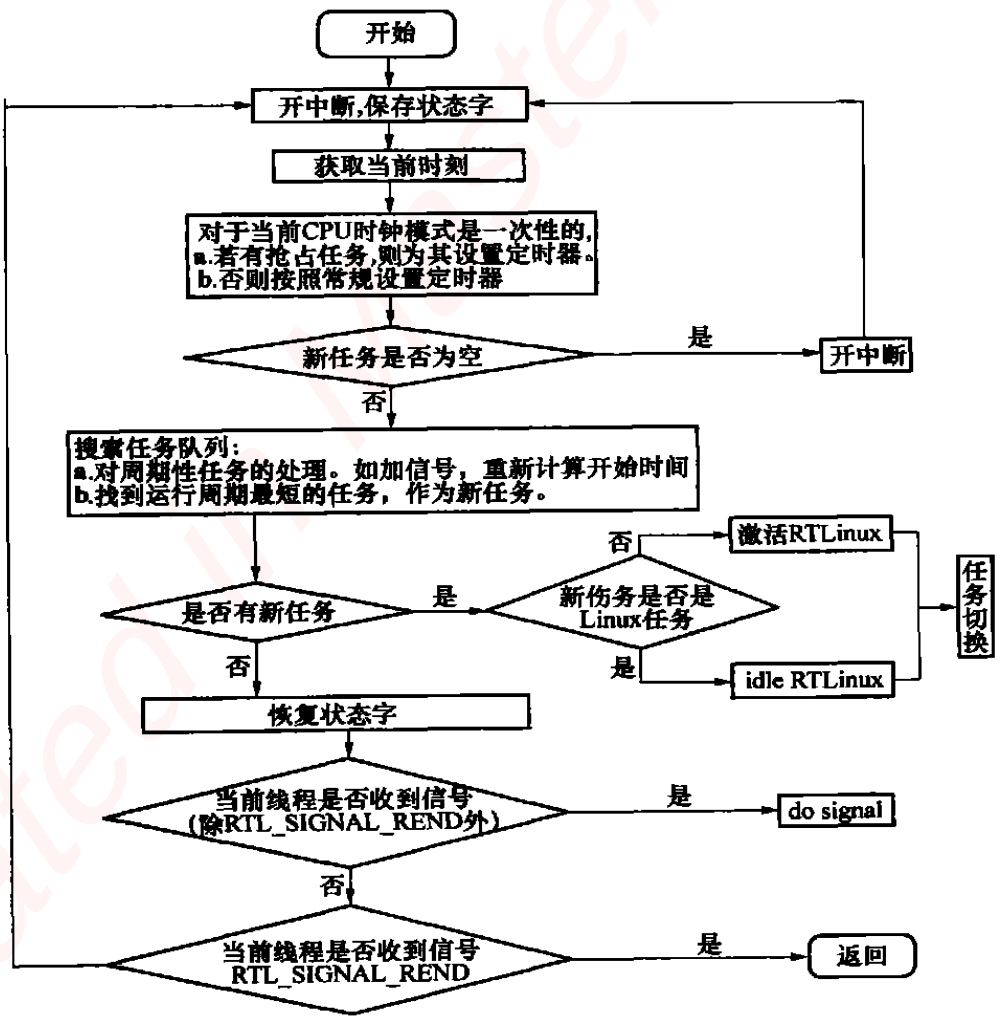


图 3　rtl-schedule() 函数流程图

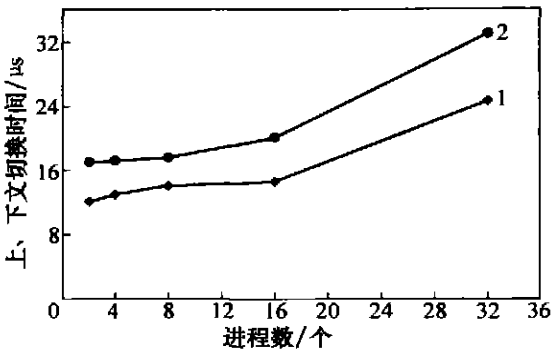
Fig. 3　Flowchart of rtl-schedule() function

RT Embedded http://www.kontronn.com

3 RTLinux 性能测试

对 Linux 进行了实时性改造后,需要对其进行测试来确定系统是否符合实际应用的需要。实时应用程序的重点是:系统中断延时,时钟的精度,上、下文切换时间,系统调用的开销以及内核的抢占能力。

采用 LMbench 测试系统的上、下文切换时间。测试机器配置为:CPU, Intel 赛扬 633;内存 128 MB;操作系统,Red Hat Linux 8.0。RTLinux 与普通 Linux 上、下文切换时间对比如图 4 所示。



1 —RTLinux; 2 —普通 Linux

图 4 RTLinux 与普通 Linux 上、下文切换时间对比

Fig. 4 Comparison of context switching time for RTLinux and general Linux

可见,对于上、下文切换时间,改造后的 RTLinux 比普通 Linux 短。测试结果表明,RTLinux 的系统调用开销降低,内核的抢占能力增强。

4 结 论

a. 针对 Linux 进程调度策略存在中断封锁时间过长、非抢占式的 Linux 内核以及耗尽式的、机会均等的调度方式这 3 个不利于实现实时性的不足,提出了提高嵌入式 Linux 实时性的方法,实现了实时 Linux 系统(RTLinux)的调度算法以及其他部分功能。

b. 采用 LMbench 测试系统对改进的 RTLinux 和通用 Linux 的上、下文切换时间进行对比测试,结果表明,改进的 RTLinux 有效提高了嵌入式 Linux 的实时性。

参考文献:

[1] 李善平. Linux 与嵌入式系统[M]. 北京:清华大学出版社, 2003.
LI Shan-ping. Linux and embedded system [M]. Beijing: Tsinghua University Press, 2003.

[2] 邓 明. PC104 嵌入式计算机在海底大地电磁信号采集中的应用[J]. 中南工业大学学报(自然科学版), 2002,33(6):555 558.
DENG Ming. The application of PC104 in sea-floor magnetotelluric signal acquisiton [J]. Journal of Central South University of Technology(Natural Science), 2002,33(6):555 558.

[3] 陈闳中. Linux 在嵌入式操作系统中的应用[J]. 同济大学学报(自然科学版),2001,29(5):565 567.
CHEN Hong-zhong. Application of Linux in embedded operating system [J]. Journal of Tongji University (Natural Science), 2001,29(5):565 567.

[4] 周德新,张向利. Linux 与嵌入式操作系统[J]. 桂林电子工业学院学报,2000,20(4):22 24.
ZHOU De-xin, ZHANG Xiang-li. Embedded operating system and embedded Linux [J]. Journal of Guilin Institute of Electronic Technology, 2000,20(4):22 24.

[5] 王学龙. 嵌入式 Linux 系统设计与应用[M]. 北京:清华大学出版社,2001.
WANG Xue-long. Design and application of embedded Linux system [M]. Beijing: Tsinghua University Press, 2001.

[6] 吴圣宁,吴海平. 嵌入式操作系统规范化内核设计研究[J]. 计算机工程, 2001,27(3):154 156.
WU Sheng-ning, WU Hai-ping. Research on the standard design of embedded OS kernel [J]. Computer Engineering, 2001,27(3):154 156.

[7] 叶以民. 嵌入式系统中的实时操作系统[J]. 测控技术,2000,19(4):6 8.
YE Yi-min. Real time operating system for embedded system [J]. Measurement & Control Technology, 2000,19(4):6 8.

[8] 陈文智,王总辉. 嵌入式实时系统 RTLinux 的实现和测试[J]. 计算机工程与应用,2001,37(19): 157 159.
CHEN Wen-zhi, WANG Zong-hui. Implementation and appraisal of embedded RTLinux [J]. Computer Engineering and Applications, 2001,37(19):157 159.

[9] SPURI M, BUTTAZZO G. Scheduling a periodic tasks in dynamic priority systems [J]. Real-Time Systems Journal, 1996, 10(1):179 210.

[10] LIU C L, LAYLAND J. Scheduling algorithms for multiprogramming in a hard real-time environment [J]. Journal of the ACM, 1973,20(1):40 60.

[11] 何 军,孙玉方. 提高软非周期任务响应性能的调度算法[J]. 软件学报,1998,9(10):721 727.
HE Jun, SUN Yu-fang. An algorithm to improve the responsive performance for scheduling soft-aperiodic tasks [J]. Journal of Software, 1998,9(10):721 727.

[12] 邹 勇,李明树,王 青. 开放式实时系统的调度理论与方法分析[J]. 软件学报,2003,14(1):83 90.
ZOU Yong, LI Ming-shu, WANZG Qing. Analysis for scheduling theory and approach of open real-time system [J]. Journal of Software, 2003,14(1):83 90.

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究
23. IEEE1588 精密时钟同步关键技术研究
24. GPS 信号发生器射频模块的一种实现方案
25. 基于 CPCI 接口的视频采集卡的设计
26. 基于 VPX 的 3U 信号处理平台的设计
27. 基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计
28. AT89C52 单片机与 ARINC429 航空总线接口设计
29. 基于 CPCI 总线多 DSP 系统的高速主机接口设计
30. 总线协议中的 CRC 及其在 SATA 通信技术中的应用
31. 基于 FPGA 的 SATA 硬盘加解密控制器设计
32. Modbus 协议在串口通讯中的研究及应用
33. 高可用的磁盘阵列 Cache 的设计和实现
34. RAID 阵列中高速 Cache 管理的优化

Creator Master PDF Editor

35. 一种新的基于 RAID 的 CACHE 技术研究与实现
36. 基于 PCIE-104 总线的高速数据接口设计
37. 基于 VPX 标准的 RapidIO 交换和 Flash 存储模块设计
38. 北斗卫星系统在海洋工程中的应用
39. 北斗卫星系统在远洋船舶上应用的研究
40. 基于 CPCI 总线的红外实时信号处理系统
41. 硬件实现 RAID 与软件实现 RAID 的比较
42. 基于 PCI Express 总线系统的热插拔设计
43. 基于 RAID5 的磁盘阵列 Cache 的研究与实现
44. 基于 PCI 总线的 MPEG2 码流播放卡驱动程序开发
45. 基于磁盘异或引擎的 RAID5 小写性能优化
46. 基于 IEEE1588 的时钟同步技术研究
47. 基于 Davinci 平台的 SD 卡读写优化
48. 基于 PCI 总线的图像处理及传输系统的设计
49. 串口和以太网通信技术在油液在线监测系统中的应用
50. USB30 数据传输协议分析及实现

VxWorks：

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发

Create by Master PDF Editor

21. 基于 VxBus 的驱动程序架构分析
22. 基于 VxBus 的高速数据采集卡驱动程序开发
23. Vxworks 下的冗余 CAN 通讯模块设计
24. WindML 工业平台下开发 S1d13506 驱动及显示功能的实现
25. WindML 中 Mesa 的应用
26. VxWorks 下图形用户界面开发中双缓冲技术应用
27. VxWorks 上的一种 GUI 系统的设计与实现
28. VxWorks 环境下 socket 的实现
29. VxWorks 的 WindML 图形界面程序的框架分析
30. VxWorks 实时操作系统及其在 PC104 下以太网编程的应用
31. 实时操作系统任务调度策略的研究与设计

Linux：

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）
17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究
22. Linux TCP IP 协议详解
23. Linux 桌面环境下内存去重技术的研究与实现
24. 掌握 Android 7.0 新增特性 Quick Settings
25. Android 应用逆向分析方法研究

Crea... Master PDF Editor

26. Android 操作系统的课程教学
27. Android 智能手机操作系统的研究
28. Android 英文朗读功能的实现
29. 基于 Yocto 订制嵌入式 Linux 发行版
30. 基于嵌入式 Linux 的网络设备驱动设计与实现
31. 如何高效学习嵌入式
32. 基于 Android 平台的 GPS 定位系统的设计与实现
33. LINUX ARM 下的 USB 驱动开发
34. Linux 下基于 I2C 协议的 RTC 驱动开发
35. 嵌入式下 Linux 系统设备驱动程序的开发
36. 基于嵌入式 Linux 的 SD 卡驱动程序的设计与实现
37. Linux 系统中进程调度策略

Windows CE：

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究
17. Windows CE 环境下无线网卡的自动安装
18. 基于 Windows CE 的可视电话的研究与实现
19. 基于 WinCE 的嵌入式图像采集系统设计
20. 基于 ARM 与 WinCE 的掌纹鉴别系统

Create by Writer PDF Editor

PowerPC：

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现
9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发
10. 基于 MPC8313E 嵌入式系统 UBoot 的移植
11. 基于 PowerPC 处理器 SMP 系统的 UBoot 移植
12. 基于 PowerPC 双核处理器嵌入式系统 UBoot 移植
13. 基于 PowerPC 的雷达通用处理机设计
14. PowerPC 平台引导加载程序的移植
15. 基于 PowerPC 嵌入式内核的多串口通信扩展设计
16. 基于 PowerPC 的多网口系统抗干扰设计
17. 基于 MPC860T 与 VxWorks 的图形界面设计
18. 基于 MPC8260 处理器的 PPMC 系统

ARM：

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现
14. 基于 S3C6410 处理器的嵌入式 Linux 系统移植

Create PDF Master PDF Editor

15. CortexA8 平台的μC-OSⅡ及 LwIP 协议栈的移植与实现
16. 基于 ARM 的嵌入式 Linux 无线网卡设备驱动设计
17. ARM S3C2440 Linux ADC 驱动
18. ARM S3C2440 Linux 触摸屏驱动
19. Linux 和 Cortex-A8 的视频处理及数字微波传输系统设计
20. Nand Flash 启动模式下的 Uboot 移植
21. 基于 ARM 处理器的 UART 设计
22. ARM CortexM3 处理器故障的分析与处理
23. ARM 微处理器启动和调试浅析
24. 基于 ARM 系统下映像文件的执行与中断运行机制的实现
25. 中断调用方式的 ARM 二次开发接口设计

Hardware：

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计
10. 基于 COM Express 的回波预处理模块设计
11. 基于 X86 平台的简单多任务内核的分析与实现
12. 基于 UEFI Shell 的 PreOS Application 的开发与研究
13. 基于 UEFI 固件的恶意代码防范技术研究
14. MIPS 架构计算机平台的支持固件研究
15. 基于 UEFI 固件的攻击验证技术研究
16. 基于 UEFI 的 Application 和 Driver 的分析与开发
17. 基于 UEFI 的可信 BIOS 研究与实现
18. 基于 UEFI 的国产计算机平台 BIOS 研究
19. 基于 UEFI 的安全模块设计分析
20. 基于 FPGA NiosⅡ的等精度频率计设计
21. 基于 FPGA 的 SOPC 设计
22. 基于 SOPC 基本信号产生器的设计与实现
23. 基于龙芯平台的 PMON 研究与开发

Creat Master PDF Editor

Programming:

1. 计算机软件基础数据结构 - 算法
2. 高级数据结构对算法的优化
3. 零基础学算法
4. Linux 环境下基于 TCP 的 Socket 编程浅析
5. Linux 环境下基于 UDP 的 socket 编程浅析
6. 基于 Socket 的网络编程技术及其实现

Created in Master PDF Editor