

一种多核混合分区调度算法设计与实现

郝继锋, 虞保忠, 周 霆, 徐晓光

(中航工业西安航空计算技术研究所, 陕西 西安 710065)

摘 要: 提出了一种基于嵌入式多核虚拟机管理器的混合分区调度算法, 对多核混合分区调度算法的配置、设计和实现过程进行阐述, 而且, 在飞思卡尔 PowerPC MPC8641D 硬件平台上对多核混合分区调度算法进行充分的功能测试和验证; 最后, 对多核混合分区调度算法的特点以及将来还需要完善的工作进行总结。

关键词: 综合化航空电子系统; ARINC653 分区调度算法; 自适应分区调度算法; 嵌入式多核虚拟机管理器; 混合分区调度算法

中图分类号: TP316.2

文献标识码: A

文章编号: 1000-7180(2016)07-0140-05

Design and Implementation of the Multicore Hybrid Partition Scheduling Algorithm

HAO Ji-feng, YU Bao-zhong, ZHOU Ting, XU Xiao-guang

(AVIC Aeronautics Computing Technique Research Institute, Xi'an 710065, China)

Abstract: We interpret the principle of the configuration, design and implementation of the algorithm, moreover, we test and certify the correctness of the new partition scheduler on the Freescale PowerPC MPC8641D hardware platform. Lastly, we summarize the features of the hybrid partition scheduling algorithm based on the multicore hypervisor and the work needs to be done in the future.

Key words: IMA; ARINC653 partition scheduling algorithm; adaptive partitioning; embedded multicore hypervisor; hybrid partition scheduling algorithm

1 引言

嵌入式多核处理器的应用需嵌入式多核操作系统的支持。2015 年 8 月 21 日由 AECC 机构发布的 ARINC653P1-4 标准^[1]增加了对多核处理器的支持, 可见嵌入式多核操作系统已是大势所趋。国外 Wind River 公司已推出最新多核分区操作系统 VxWorks 653 3.0^[2]、多核虚拟机管理器 Wind River Hypervisor^[3], Green Hills 公司推出多核虚拟机管理器 INTEGRITY Multivisor^[4]。

目前, 航电系统逐步从当前的物理综合向功能综合方向发展。在一些新型飞行器中, 出现了机电系统、飞控系统和航电系统大综合的情况, 一个物理平台承载了不同类型的任务, 这些任务对平台的处理

特性有着不同的需求。飞控、机电应用要求强实时响应, 与航电应用综合在一起, 提出了强实时任务与一般任务共存的需求, 这就要求航电系统中的软件产品应提供对强实时任务的处理能力。

上述商业软件产品遵循 ARINC653 标准^[5-8], 严格按照时间表来调度分区, 难以支持强实时应用, 迫切需要提出新调度算法来扩展传统 ARINC653 分区调度。QNX Hypervisor^[9]设计出“自适应分区”调度算法, 把优先级驱动和时间驱动两种调度算法相结合, 但没有给每个分区定义时间槽, 所有分区在任何时间都能根据线程的优先级来竞争 CPU, 该算法的目标是使 CPU 利用率最大化, 复杂度为 $O(n)$, 只适用于软实时系统, 不适用于强实时系统。

所以目前迫切需要设计并实现一种基于嵌入式多核基础软件产品,且对强实时响应提供支持的分区调度算法。基于此种情况,本文针对应用于航空电子系统的嵌入式多核虚拟机管理器设计并实现了一种创新性的多核混合分区调度算法。

2 多核混合分区调度配置

2.1 单核配置项

单核混合分区调度配置数据如表 1 所示。

表 1 单核配置项	
配置项	备注
调度表 ID	[0, 255]
调度表数量	表示用户已配调度表数量
调度表名称	最多 30 个字符
帧数量	
分区 ID	[0, 255]
分区名	配置窗口中运行的分区名字
分区数量	表示用户已配置的分区数
分区持续时间	

2.2 多核配置项

多核混合分区调度配置方案:给每个处理器核配置一个调度表,每个处理器核独立进行调度。

为了每个处理器核上的调度可同时执行,用户可配置是否进行处理器核调度同步,所有配置同步属性的处理器核在主帧结束时刻都必须进行调度同步操作,如图 1 所示。

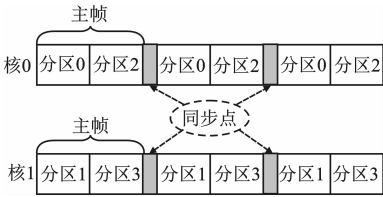


图 1 调度同步时机

为有效使用帧分区调度器,必须把分区绑定到相应的处理器核上。因此,并不是所有的调度表都可以由所有的处理器核来执行,例如,如果只把分区 0 分配给核 0,那么,核 1 就不能执行包含分区 0 的调度表。

在多核环境下,需要增加两个额外配置项:同步配置项(在同步使能前提下)和 CPU 列表配置项。如表 2 所示。

表 2 多核配置项

配置项	子配置项
同步配置项	初始同步持续时间
	初始同步超时时间
	主帧持续时间
	最大抖动时间
CPU 列表配置项	CPU ID
	同步属性

同步配置项包括四个配置子项,如下所示。

初始同步持续时间:最后一个参与调度的处理器核完成初始同步的时间。

初始同步超时时间:在初始同步期间,每个参与调度的处理器核等待另一个核的最大时间。

主帧持续时间:任意一个参与调度的处理器核上任意一个调度表的最大持续时间。

最大抖动时间:在调度同步期间最大的超时。

每一个处理器核使用主帧开始时间和主帧持续时间相加得出新的主帧开始时间。处理器核必须等待到这个时刻才能开始进行下一次调度。如图 2 所示。

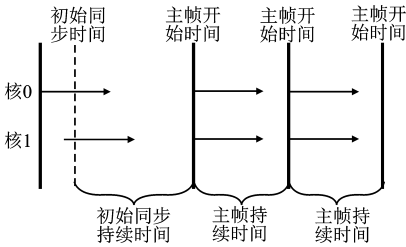


图 2 调度同步时间关系

如表 2 所示,CPU 列表配置项包括以下两个配置子项。

CPU ID:处理器核 ID。

同步属性:配置一个处理器核是否参与调度同步,缺省值为“禁止”。如果配置为同步使能,则进行处理器核的同步操作。

3 多核混合分区调度设计与实现

3.1 多核混合分区调度架构设计

在进行分区混合调度时,把分区分为两类:在时间调度表中的普通轮转分区和不在时间调度表中的紧急响应分区(例如设备服务分区),如图 3 所示。

参与时间表调度普通分区中的线程加入普通分区位图和相应就绪队列;而紧急响应分区中的线程

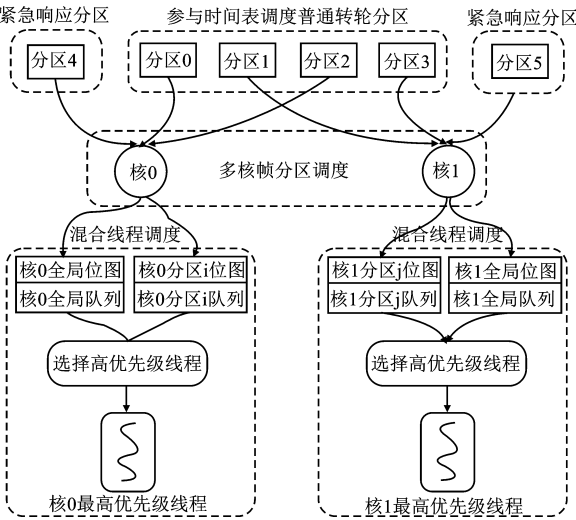


图 3 多核混合分区调度设计

加入紧急响应分区位图和相应就绪队列,紧急响应分区将一直和当前分区参与线程调度,在选择线程运行时,分别从紧急响应分区就绪队列和当前帧对应的普通分区就绪队列选择最高优先级线程,优先级最高者获得处理器资源,这样的设计既能达到分区轮转的效果,又能保证紧急线程的响应。

3.2 关键数据结构设计

基于嵌入式多核虚拟机管理器的混合分区调度算法设计的关键数据结构包括多核帧分区调度数据结构、混合线程调度数据结构。

3.2.1 多核帧分区调度数据结构设计

在实现多核帧分区调度策略时,需要设计两个主要数据结构:分区调度控制块和帧节点控制块。

(1)分区调度控制块

分区调度控制块主要记录的数据信息包括主帧时间、最大帧索引、当前帧的窗口时间、当前主帧已经执行的时间、已经执行的帧数、用户配置的帧数、时间表的定时器(需要声明为每 CPU 变量)、当前执行的帧节点(需要声明为每 CPU 变量)、时间表的节点表和帧分区管理位图。

其中,帧分区管理位图用于对已挂载的分区进行管理,大小定义为 8×32 位来管理 256 个分区。设计的管理方案:当系统检测到哪一个分区配置成功时,就需要设置帧分区管理位图的相应位,例如,当系统在编译链接时配置分区 0、分区 1 和分区 2 时,则需要设置帧分区管理位图的第 0 位、第 1 位和第 2 位。

(2)帧节点控制块

为了表示每一个帧节点,需要定义节点控制块,

它主要记录的数据信息包括:节点对应的分区 ID、节点对应的分区指针、节点起始时间和节点持续时间。

3.2.2 混合线程调度数据结构设计

在实现混合线程调度算法时,需要设计两种类型的位图和就绪队列:普通分区位图/就绪队列、紧急响应分区位图/就绪队列。其中,普通分区位图/就绪队列在分区创建时进行相应初始化;而紧急响应分区位图/就绪队列在创建调度器控制块时进行相应初始化。

3.3 多核混合分区调度过程

在实现混合分区调度算法时,主要包括三个过程:调度表创建过程、多核帧分区调度过程和混合线程调度过程。

3.3.1 调度表创建过程

调度表的创建过程如图 4 所示。

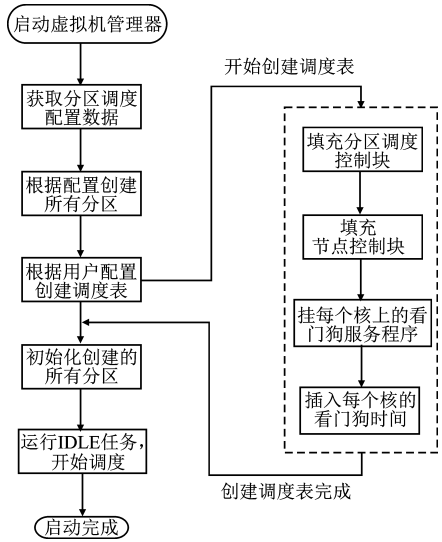


图 4 调度表创建过程

在创建时需要根据用户的配置数据填充分区调度控制块和节点表控制块,然后为每个处理器核挂看门狗服务程序(即时间表调度程序),而且,插入每个处理器核看门狗时间(即当前窗口的持续时间)。

3.3.2 多核帧分区调度过程

虚拟机管理器触发时钟中断时,主核和从核都会响应时钟中断,主核和从核在处理时钟中断时,会递减等待时间,并更新指定看门狗链的状态,如果计时值为 0,则触发相应看门狗服务程序(时间表调度程序)。多核帧分区调度过程如图 5 所示。

3.3.3 混合线程调度过程

混合线程调度过程如图 6 所示。虚拟机管理器在调度时,分别从普通分区队列以及紧急响应分区

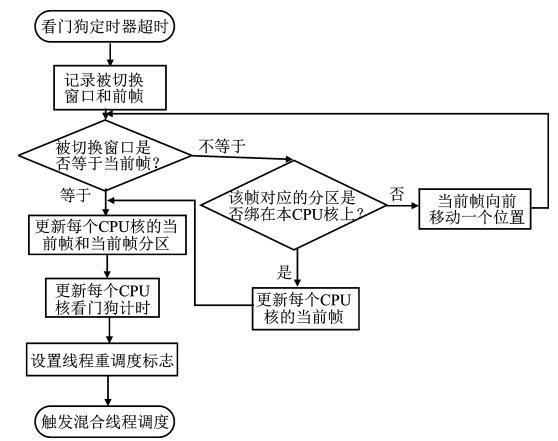


图 5 多核帧分区调度过程

队列中获取最高优先级就绪线程,比较两个线程,运行优先级高的线程;如果优先级相同,优先运行紧急响应分区的线程。

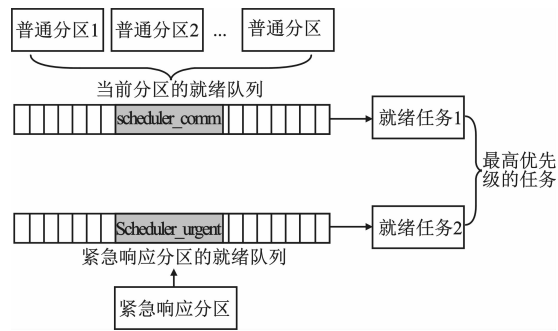


图 6 混合线程调度过程

4 多核混合分区调度测试与验证

4.1 测试硬件环境

硬件测试环境如图 7 所示,1 台 PC 机和 1 台 MPC8641D 目标机通过 1 台以太网交换机连接,用于加载待测试程序及调试控制;1 台 MPC8641D 目标机的串口通过 1 台 N-port,经以太网交换机连接到 PC 机,用于查看目标机程序的执行输出。

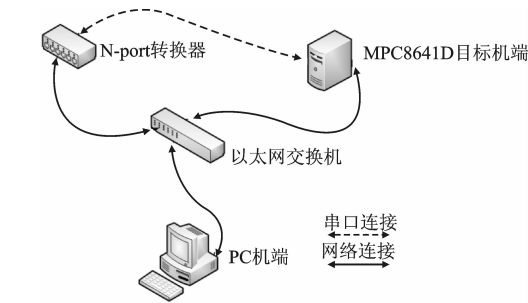


图 7 硬件测试环境

4.2 调度测试方案

功能测试分两步来实施:单核混合分区调度测试和多核混合分区调度测试。

4.2.1 单核混合分区调度功能测试

(1)标准 ARINC653 分区调度测试

测试方案:为虚拟机管理器配置 3 个分区(分区 0,分区 1 和分区 2),每个分区对应窗口的持续时间相等,都配置为 5 tick;

在配置的 3 个分区内为每个分区创建一个优先级为 240 的用户线程;

(2)分区调度的故障隔离测试

测试方案:为虚拟机管理器配置 3 个分区(分区 0,分区 1 和分区 2),每个分区对应窗口的持续时间相等,都配置为 5 Tick;

在配置的 3 个分区内为每个分区创建一个优先级为 240 的用户线程;分区 0 模拟故障分区,它在参与 50 次正常的分区窗口转轮调度后,处理故障,而分区 1 和分区 2 正常进行分区转轮调度。

(3)紧急事件响应测试

测试方案:在 Eside 集成开发环境中为虚拟机管理器配置 4 个裸应用分区:分区 0、分区 1、分区 2 和分区 3;在每个分区内创建一个优先级为 240 的线程;在分区 3 中创建一个优先级为 245 的周期线程(紧急线程),每隔 10 tick 激活一次该线程;配置 3 个时间窗口,每个时间窗口的持续时间配置 20 tick。

4.2.2 多核混合分区调度功能测试

多核混合分区调度测试方案如图 8 所示。

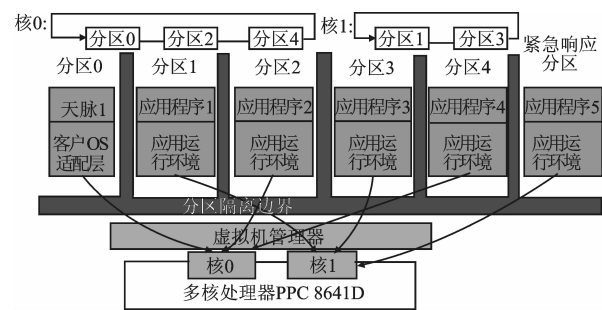


图 8 多核混合分区调度测试方案

测试方案说明:

(1)在 PowerPC8641D 双核处理器平台上为虚拟机管理器项目配置 6 个分区,5 个处于调度表中的分区(分区 0、分区 1、分区 2、分区 3 和分区 4),每个分区的窗口持续时间为 600 tick;1 个紧急分区(分区 5),紧急线程 800 tick 运行一次。

分区 0 是天脉 1 分区,在分区 0 创建了两个相同优先级客户操作系统打印线程;

分区 1 是裸应用分区,在分区 1 创建两个相同优先级的打印线程(优先级为 245);

分区 2 内创建两个线程相同优先级(优先级为 245),线程 0 运行一段时间非法访问内核空间会触发异常,之后被挂起,线程 1 普通打印线程;

分区 3 内创建两个线程相同优先级(优先级为 245),线程 0 运行一段时间访问其他分区空间会触发异常,之后被挂起,线程 1 普通打印线程;

分区 4 内创建两个线程相同优先级(优先级为 245),线程 0 运行一段时间访问特权指令会触发异常,之后被挂起,线程 1 普通打印线程;

分区 5 是紧急响应分区,分区 5 创建一个紧急线程(优先级为 250),用于定时抢占普通分区轮转线程。

(2)分区 0、分区 2 和分区 4 绑定在主核(核 0)上运行;分区 1 和分区 3 绑定在从核(核 1)上运行。

4.3 测试结果分析

4.3.1 单核测试结果分析

(1)标准 ARINC653 分区调度测试分析

串口循环输出“00..0011..1122..22”结果,其中,0 表示 0 分区,1 表示 1 分区,2 表示 2 分区。如果打印 0 的个数、1 的个数和 2 的个数基本相等,表示系统的分区窗口轮转调度算法可以正常进行调度。

测试结果:可按照标准 ARINC653 进行分区调度。

(2)分区调度的故障隔离测试分析

首先串口输出 3 分区进行 50 次正常的轮转调度结果,类似“00..0011..1122..22”;分区 0 在参与 50 次分区轮转调度后模拟分区跑飞的情况,打印类似“P0 BUG_ON,handle error...”,表示分区 0 出现故障,不再参与分区窗口轮转调度;分区 0 发生故障后,分区 1 和分区 2 仍旧可以正常进行分区窗口的轮转调度,串口打印类似“11..1122..22”。

测试结果:分区调度可进行故障隔离。

(3)紧急事件响应测试分析

串口循环输出打印信息为:“0..U..0 U 1..U..1 U 2..U..2”表示 ARINC653 线程可隔固定时间被高优先级的线程抢占。

测试结果:分区调度允许紧急事件响应。

4.3.2 多核测试结果分析

串口循环输出打印信息为:核 0 上依次循环调

度分区 0、2、4;每个分区中的两个线程都能够正常打印输出;分区 2 中的线程 0 输出打印 8 次后触发异常被挂起,该分区只有线程 1 正常运行;分区 4 中的线程 0 输出打印 8 次后触发异常被挂起,该分区只有线程 1 正常运行。

核 1 上依次循环调度分区 1、3;每个分区中的两个线程都能够正常打印输出;分区 3 中的线程 0 输出打印 5 次后触发异常被挂起,只有线程 1 正常运行。

测试结果:具备分区时空隔离和紧急事件响应能力。

5 结束语

本文对基于嵌入式多核虚拟机管理器的混合分区调度算法进行了设计、实现、测试和验证。该方法具有一定的创新性,在嵌入式多核软件平台上把时间驱动调度和优先级驱动调度相结合,是航空领域标准 ARINC653 分区调度算法的一个超集。

创新点是:区别于标准 ARINC653 调度算法,除了处于调度表中的普通时间分区之外(处于调度表中的普通分区在一个时刻只能有一个分区处于激活状态),混合分区调度方法还设计了一个紧急响应分区(诸如设备服务分区),它一直处于激活状态,即同时允许有两个分区处于激活状态。

多核混合分区调度算法从普通分区和紧急响应分区中选择最高优先级的线程执行。因为混合分区调度器每一次切换只考虑两个分区,所以它的执行时间是常数,即多核混合分区调度器实现 $O(1)$ 复杂度。

该调度算法已在中航工业计算所自主研发的天脉 MILS 操作系统中实现,且已进行了充分的功能测试。将来还需完成的工作主要是处理器核间调度同步功能。为了保证在有限次迭代中完成调度同步,需要实现经典或者改进栅栏同步算法。

参考文献:

- [1] AECC. Arinc Specification 653P1-4. Avionics Application Software Standard Interface Set[S]. 2015-08-21.
- [2] Vxworks653 product overviewmulticore. pdf[R]. USA: Wind River,2015.
- [3] wr_hypervisor_getting_started_2. 0. pdf. [R]. USA: Wind River,2013-10-28.
- [4] Multicore Virtualization Update. pdf. [R]. 绿山公司,2015.

(下转第 149 页)

(4) 使用统计工具对表 3 得到的类与类之间的关系权重矩阵实施聚类分析,得到如图 5 所示的树状图。

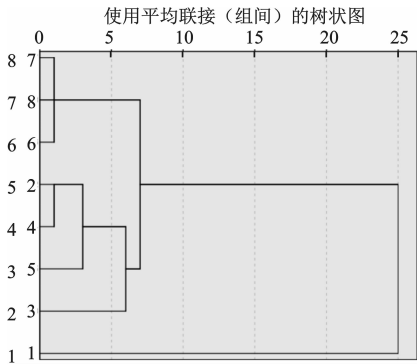


图 5 聚类后的树状图

设置重新调整距离为 15,从图 5 可以看出,该软件系统可以划分为三个组件: $S1 = \{C1\}$; $S2 = \{C2, C3, C4, C5\}$; $S3 = \{C6, C7, C8\}$. 从图 3 所示的类图来看这种划分组件的方法是合理的,因为类 C1 在整个类图中是属于关键类,它的修改变动会影响类 C2, C3, C4, C5, 而类 C6, C7, C8 的修改变动会影响类 C1, 从而间接影响类 C2, C3, C4, C5, 所以可以单独将类 C1 作为一个组件,有利于软件系统交付后的维护活动;而把类 C2, C3, C4, C5 聚合在一个组件上是因为它们都受到类 C1 的影响;类 C6, C7, C8 聚合在一个组件上,是因为它们具有共同的特征,就是它们的每一个修改变动都会影响类 C1, 从而进一步影响类 C2, C3, C4, C5.

4 结束语

UML 类图是软件设计阶段最重要的产品,为了提高软件系统的可维护性,减少软件产品交付后的维护活动、降低软件维护成本、使维护活动易于进行,模块化设计是最佳的设计方案,如何组织 UML 类图中的类,以便构造可以再用、易于维护的组件,本文提出了一种基于 UML 类图中类与类之间的关

系将 UML 类图转化为带权的有向图,利用图论的相关理论最终实现将关系相近的类组合成一个组件,从而实现软件系统模块化设计的目的。

参考文献:

[1] Wensheng Hu, Ming Zhao, Shiyi Wu, Jianfeng yang. Requires analysis based on software maintainability [C]// Reliability, Maintainability and Safety(ICRMS), 2014 International Conference Washington, DC: IEEE 2014: 354-357.

[2] GAOHui, ZHANG Li, Li Lin. Software flexibility prediction model in software architecture level[J]. Journal of Software, 2010; 21(9): 2118-2134.

[3] 杨芙清,梅宏. 构件化软件设计与实现[M]. 北京:清华大学出版社, 2008.

[4] Griswold W G, Sullivan KJ, Song Y, et al. Modular Software Design with Crosscutting Interfaces [J]. IEEE Software, 2006, 23(1): 51-60.

[5] Olaf Seng, Markus Bauer, Matthias Biehl, et al. Search-based improvement of subsystem decomposition [J]. In GECCO'05: Proceedings of the 2005 conference on Genetic and evolutionary computation, New York, NY, USA: ACM Press, 2005: 1045-1051.

[6] Zhang Yan, Mei Hong. Non-functional attributes oriented description and verification in UML class diagrams [J]. Journal of Software, 2009, 20(6): 1457-1469.

[7] WangXB, Wang H, Liu C. Automatic Hierarchical Layout Algorithm for UML Class Diagram[J]. Journal of Software, 2009, 20(6): 1487-1498.

作者简介:

胡文生 男, (1969-), 博士, 副教授. 研究方向为软件可靠性、可维护性. E-mail: 1204489713@qq. com

杨剑锋 男, (1987-), 博士, 副教授. 研究方向为软件可靠性、可维护性.

赵 明 男, (1962-), 教授, 博士生导师. 研究方向为软件可靠性、应用统计.

(上接第 144 页)

[5] AEEC. Avionics Application 65394 Software Standard Interface Part 1 - Required Services[S]. 2015-12-01.

[6] AEEC. Avionics Application 65394 Software Standard Interface Part 2 - Extended Services[S]. 2015-01-22.

[7] AEEC. Avionics Application 65394 Software Standard Interface Part 3 - Conformity Test Specification[S]. 2016-10-11.

[8] AEEC. Avionics Application 65394 Software Standard

Interface Part 4 - Subset Services[S]. 2012-06-28.

[9] QNX. Adaptive Partitioning User's Guide[R]. 2014-10-16.

作者简介:

郝继锋 男, (1982-), 硕士, 工程师. 研究方向为嵌入式实时操作系统、TCP/IP 网络协议栈、UNIX/Linux 环境编程. E-mail: haojifeng1187@163. com.