一种基于 μC/OS-II 的高可靠实时系统内核设计

朱怡安[1] 林 鹤[2]

(西北工业大学计算机学院 西安 710072)[1] (西北工业大学软件与微电子学院 西安 710072)[2]

摘 要 基于 μC/OS-II,设计并实现了一个高可靠的分区嵌入式操作系统内核。为了保证高关键级别分区的执行时间,提高系统可靠性,提出了一种新的周期执行时间可变的分区调度算法,其能在保障高安全关键级别分区优先执行的同时提供较高的资源利用率和任务可调度性。采用了一种基于位运算的访问控制算法,其可保证系统的信息安全,提高系统的可靠性。最后,通过算法分析和实验结果验证了所提算法的有效性和实时性以及系统的可靠性。

关键词 高可靠,时空分区,实时系统,分区调度,访问控制

中图法分类号 TP316 **文献标识码** A

μC/OS-II Based Highly Reliable RTOS Kernel

ZHU Yi-an[1] LIN He[2]

(School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China)[1]

(School of Software and Microelectronics, Northwestern Polytechnical University, Xi'an 710072, China)[2]

Abstract A highly reliable embedded partition operating system kernel based on μC/OS-II was designed and implemented. In order to ensure the execution time of the high safety-critical level partition and improve the reliability of the system, a new partition scheduling algorithm which has variable execution time in every cycle was proposed. It can not only guarantee the partition with higher safety-critical level execution first, but it also has good resource utilization and task schedulability. The kernel introduced access control to ensure information security of the system, and a bit computing based access control algorithm was also proposed. Finally, algorithm analysis and experiments demonstrate the effectiveness and timeliness of our algorithm.

Keywords Highly reliable, Time and space partitioning, RTOS, Partition scheduling, Access control

1 引言

安全关键系统(特别是应用于航空、汽车等领域的安全关键系统)对嵌入式操作系统的可靠性有着很高的要求。ARINC653 标准提出了将应用程序分区的概念,以保证 IMA (Integrated Modular Avionics)系统[1]中一个处理机模块上运行的多个应用程序执行时互不影响,从而提高系统的可靠性和可验证性。而且分区系统可以很方便地将各种不同安全关键级别的应用集成在一个处理机模块上,从而降低整个系统的体积、重量和能耗[2]。ARINC653 的分区架构[3]在需要高可靠性的领域尤其是航空电子的嵌入式操作系统中应用广泛。

由于通常一个分区内会有多个任务,因此分区系统需要提供分区调度和分区内任务的调度,也称为局部调度和全局调度。有很多关于这种分层调度的研究[4-7],但目前仍缺乏对于分区安全关键的保证与分区的资源预留的综合研究。

在分区系统中,每个运行在目标系统上的应用都有一个安全关键级别。例如在一个航空电子系统中会有飞行控制分区和一些管理诸如广播、空调等设备的任务分区,前者的关键级别显然应比后者高。只要分区没有超过运行时限,分区间的调度顺序对分区的执行就没有什么影响。然而,一个任务可能因为错误的最坏执行时间预测而超时运行,此时任务的释放点是任务能否在死限之前完成的一个非常重要的因素。因为不同的应用可能由不同的独立组织开发,但是开发者并没有整个系统的全局资料,也就无法在充分考虑每个任务的死限时间和分区的关键级别的前提下来决定分区的周期和执行时间,所以一个可以自动为每一个分区分配周期和执行时间的机制是非常有用的。

随着网络的发展和通信的需求,传统的嵌入式系统领域越来越需要信息安全的保障。访问控制是一种划分主客体并限制主体对客体的访问权限的方法,它可以保证系统资源的使用是可控且合法的[8]。强制访问控制模型是一种严格控制信息流向的高安全访问模型,最早的研究有 BLP 模型和 Biba 模型[9]。一些文献[10,11]对两种模型的优点进行了整合,并在 μC/OS-II 中进行了实现,但是其算法判断复杂,时间效率低。

本文设计的高可靠实时嵌入式操作系统内核采用了分区机制并引入了访问控制,同时针对上述的问题提出了新的分区调度算法和访问控制算法。本文的分区调度算法可防止在时间资源充裕的情况下发生低关键分区中任务对高关键分区

本文受航天支撑技术基金(2013-HT-XGD(10)),航空基金(20150753010),陕西省工业科技攻关项目(2015GY035),航空基金(20130753006)国家 xx 专项(XJZ-2015-X-X76)资助。

朱怡安(1961-),男,博士,教授,博士生导师,主要研究方向为高性能计算、云计算、普适计算;林 鹤(1992-),男,硕士生,主要研究方向为嵌入式操作系统,E-mail:linhe@mail.nwpu.edu.cn。

中任务的抢占，并且可提供较高的系统吞吐量和任务可调度性；而访问控制算法可在 O(1)时间内完成访问控制权限的判断，具有高效、稳定的特点，在提供信息安全保障的同时保证了系统的实时性。

2 基于 μC/OS-II 的分区系统内核设计

μC/OS-II 是一种适用于对安全性要求较高的系统的开源实时系统内核，2000 年 7 月，其在一个航空项目中得到了美国联邦航空管理局对用于商用飞机、符合 RTCA DO-178B 标准的认证[12]。μC/OS-II 内核的设计简洁高效，因此本文使用 μC/OS-II 作为高可靠嵌入式操作系统的开发基础，扩展和修改了一些 μC/OS-II 的原有功能，并加入了新的内核模块来实现应用分区功能，整体的设计如图 1 所示。

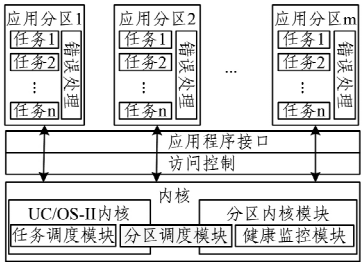

图 1 系统整体设计

μC/OS-II 原有的任务模型并不能满足 ARINC653 对任务的描述和要求，需要进行修改。在 μC/OS-II 中，任务按其优先级进行标识，任务之间不能有相同的优先级，同时任务按优先级进行调度，而 ARINC653 中规定任务可以有相同优先级，相同优先级任务之间使用先进先出(First in First out，FIFO)算法进行调度。修改之后的任务就绪队列如图 2 所示，其中 OSTCBPrioRdyTbl 是一个按优先级排列的表，可以查询每个优先级对应的就绪任务队列，任务的就绪队列是一个循环队列结构，可以快速进行队尾的插入和队首的查询。任务调度的算法中保留了 μC/OS-II 的高效查询算法，即按照优先级位图查找最高优先级的算法。

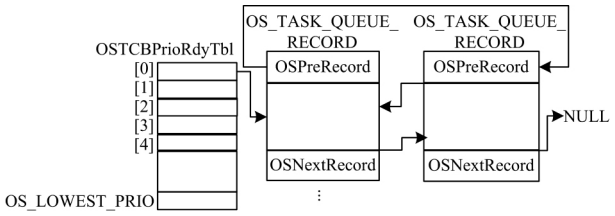

图 2 任务就绪记录结构

分区将程序的运行环境分为系统级和应用级，分区内的任务运行在应用级，内核程序运行在系统级，而 μC/OS-II 并没有系统级和应用级的概念，因此需要根据硬件提供的特权级别对系统级和应用级进行区分，并增加从应用级切换到系统级程序的系统调用功能。实现分区之间的空间隔离则需要内存管理硬件 MMU 的支持，通过内存的映射表可以给每个分区设置独立的地址空间，使不同的分区在空间上实现互相隔离。

分区间的时间隔离是通过分区的调度表来实现的，分区调度表有一个主时间框架，一般是每个分区周期的最小公倍数，系统以主时间框架为周期对所有的分区进行调度。在每一轮的调度中，分区会被分配一个或多个时间窗口。当某个分区的时间窗口到达时就运行该分区，时间窗口结束时进行分区切换。表 1 和表 2 是一个有 3 个分区的系统的示例，其给出了每个分区的时间窗口的偏移和执行时间，图 3 为分区调度的示意，主时间框架为 200ms。

表 1 分区的时间调度

分区	P1	P2	P3
周期(ms)	100	100	200
运行时间(ms)	40	20	40

表 2 分区的时间调度表

分区	P3	P1	P2	P1	P2	P3
窗口偏移(ms)	0	20	60	100	140	160
窗口时间(ms)	20	40	20	40	20	20

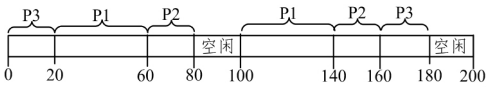

图 3 按表 2 进行分区调度

3 周期执行时间可变的分区调度算法

3.1 分区调度模型

本文的分区调度算法在决定分区执行顺序时按照固定优先级抢占的方法，系统的集成者需要根据关键级别给每个分区设置一个固定且唯一的优先级。本文只讨论分区中的任务都为周期性任务的情况。首先定义周期性任务为 $T_i^j(p_i^j, e_i^j)$ 代表分区 j 中的任务 i，p_i^j 代表它的周期，e_i^j 代表它的执行时间，然后定义分区为 $P^j(\lambda^j, \theta^j)$，表示分区 j 的周期为 λ^j，执行时间为 θ^j，j 不仅用来标识分区，同时也用来表示分区的优先级，数值越小其优先级别越高。设定分区 j 中最大的任务周期 $p_{\max}^j = \mathrm{Max}(p_1^j, p_2^j, \cdots, p_n^j)$，最小的任务周期 $p_{\min}^j = \mathrm{Min}(p_1^j, p_2^j, \cdots, p_n^j)$，$n$ 为分区 j 中的任务数目。用 $p_{\max}^{all} = \mathrm{Max}(p_{\max}^1, p_{\max}^2, \cdots, p_{\max}^m)$ 表示所有分区中的最大任务周期，$p_{\min}^{all} = \mathrm{Min}(p_{\min}^1, p_{\min}^2, \cdots, p_{\min}^m)$ 表示最小的任务周期，并设定 $R^j = (\frac{p_{\max}^j}{p_{\min}^j})$。为了简单起见，假设一个任务的周期总是可以被任何比它小的任务周期整除。本文所有的调度算法的讨论都只针对单处理器系统，且忽略上下文切换和中断处理的时间。

下面通过分析两个简单的算法示例来说明分区调度算法需要解决的问题。

(1)简单分区调度算法：简单的分区调度如式(1)、式(2)所示，分区的周期直接选择最小的任务周期，分区的执行时间则为每个任务在该周期内需要执行的时间(任务周期等于分区周期)或者是平均需要执行的时间(任务周期是分区周期的倍数)之和。

$$\lambda^j = p_{\min}^j \tag{1}$$

$$\theta = \sum_{i=1}^{n} \left\lceil \frac{e_i^j}{(\frac{p_i^j}{p_{\min}^j})} \right\rceil \tag{2}$$

现假设在一个系统中有两个分区，其参数如表 3 和表 4 所列，设任务的序号越小优先级越高，分区 $P1$ 比分区 $P2$ 优先级更高。根据式(1)和式(2)可以得到 $P1(20,6)$ 和 $P2(40,16)$，其具体的调度过程如图 4(a)所示，可以看到在 6ms 时 $P1$ 中的任务 3 被 $P2$ 中的任务 1 所抢占，因此将分区 $P1$ 的任务 5 的运行一直推迟到了第 62ms，此时如果任务运行超时，可能没有足够的缓冲空间来保证任务在死限之前完成。

表 3　分区 1 的任务示例

任务	周期(ms)	执行时间(ms)
1	20	2
2	80	4
3	80	4
4	80	4
5	80	4

表 4　分区 2 的任务示例

任务	周期(ms)	执行时间(ms)
1	40	4
2	40	8
3	80	8

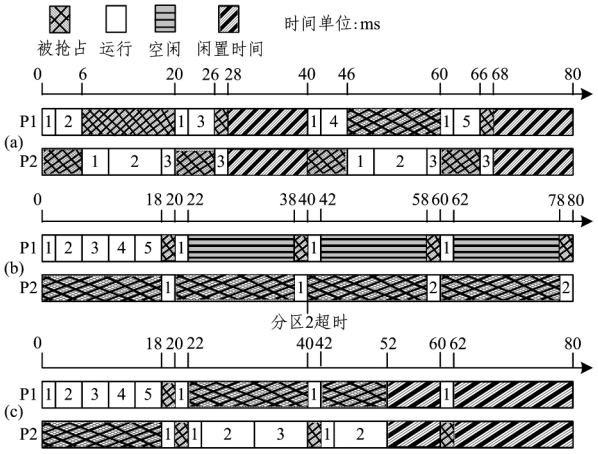

图 4　分区调度示意图

(2)简单的防止关键级别反转的算法:为了防止关键级别的反转,需要使高优先级分区中的任务比低优先级分区中的任务更早运行,这可以通过给高优先级分区预留足够的运行时间来实现。其算法如式(3)和式(4)所示。

$$\lambda^j = u \times p^j_{\min} \tag{3}$$

$$\theta^j = \lambda^j - I_u \tag{4}$$

其中,I_u 表示第 $u(1 \leqslant u \leqslant R^j)$ 个周期的剩余空闲时间。u 是使得 $I_u > 0$ 或者 $I_{u+1} = 0$ 的最小值,而当 $I_{R^j} = 0$ 时 $u = R^j$。I_u 的计算公式如式(5)所示。

$$I_u = p^j_{\min} - \sum_{i=1}^{n}(e_i^j \times f_i) - L_u \tag{5}$$

f_i 代表任务 T_i^j 会或不会在当前周期内运行,其在运算时假设所有的任务都是在其周期一开始的时候就想要运行。f_i 的计算如下:

$$f_i = \begin{cases} 1, & p_i^j = p^j_{\min} \\ 1, & r \bmod (\dfrac{p_i^j}{p^j_{\min}}) = 1 \\ 0, & \text{上述条件不成立} \end{cases} \tag{6}$$

式(5)中的 L_u 则表示前一周其没有运行完的任务负载。之所以会有多余的任务负载并不是由于任务超过了死限时间,而是由计算 f_i 时假设条件引起的。L_u 的计算如下:

$$L_u = \begin{cases} 0, & u = 1 \\ -I_{u-1}, & I_{u-1} < 0 \\ 0, & I_{u-1} \geqslant 0 \end{cases} \tag{7}$$

表 3 和表 4 中的分区由式(3)、式(4)可以得到 $P1(20, 18)$和 $P2(40, 20)$,很显然资源的使用率已经大于 1,所以这种方法将导致部分任务无法得到有效执行,其调度过程如图 4(b)所示。可以看到由于式(4)为了防止关键级别反转预留了太多的资源,导致分区 2 中的任务 3 和任务 4 无法得到执行。因此,这种方法无法有效地利用系统资源。

3.2　周期执行时间可变的分区调度算法

上述算法在计算分区的周期和执行时间时仅仅考虑使当前分区满足约束条件,因此产生了关键级别反转和资源预留过多的问题。

Hyun-Wook Jin 和 Sanghyun Han[13] 提出了使分区在各个周期的执行时间可变的方法,将其算法运用于表 3 和表 4 的分区可以得到图 4(c)所示的分区调度,其算法虽然解决了关键级别反转和预留资源过多的问题,但是其算法对关键级别执行顺序的要求过于苛刻而导致可调度性降低,例如当改变表 4 中的任务 $T_1^2(40,4)$为 $T_1^2(20,4)$时,其算法将导致 T_1^2 超出死限时间,如图 5(a)所示,T_1^2 在 20ms 的时候超出了死限时间。

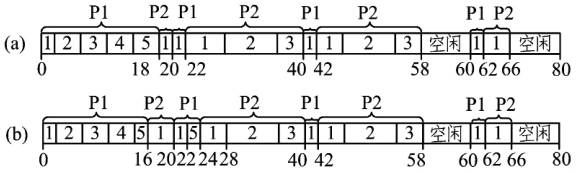

图 5　T_1^2 的周期为 20ms 时的分区调度

针对上述问题,本文提供了一种新的算法,这种算法在为关键级别高的分区按需预留足够资源的同时,通过允许较低关键级别分区中的紧迫任务抢占较高关键级别中的非紧迫任务达到较高的可调度性,从而充分利用系统的资源。用 $P^j(\lambda^j, \theta_r^j)$ 来表示分区,其中 θ_u^j 代表分区 j 在第 u 个周期的执行时间,其计算方法如式(8)和式(9)所示。

$$\lambda^j = p^{all}_{\min} \tag{8}$$

$$\theta_u^j = m_u^j + L_u + Min(p^{all}_{\min} - \sum_{i=1}^{j-1}\theta_u^j - m_u^j - L_u - \sum_{i=j+1}^{m} m_u^i, s_u^j) \tag{9}$$

其中,m_u^j 表示在第 u 周期分区 j 必须要执行完的任务的执行时间之和,其计算公式如下:

$$m_u^j = \sum_{i=1}^{n}(e_i^j \times fm_i) \tag{10}$$

其中,fm_i 表示该任务是否在第 u 个周期必须执行完,其计算公式如下:

$$fm_i = \begin{cases} 1, & p_i^j = p^{all}_{\min} \\ 0, & \text{上述条件不成立} \end{cases} \tag{11}$$

式(9)中的 s_u^j 的计算公式如下:

$$s_u^j = \sum_{i=1}^{n}(e_i^j \times fs_i) \tag{12}$$

fs_i 表示当前周期是否是该任务的起始周期,$p_i^j = p^{all}_{\min}$ 的情况除外,其计算公式如下:

$$fs_i = \begin{cases} 1, & u \bmod (\dfrac{p_i^j}{p^{all}_{\min}}) = 1 \ \&\ p_i^j \neq p^{all}_{\min} \\ 0, & \text{上述条件不成立} \end{cases} \tag{13}$$

式(9)中的 L_u 的含义与 3.1 节的相同,计算公式如下:

$$L_u = \begin{cases} 0, & r = 1 \\ -I_{u-1}, & I_{u-1} < 0 \\ 0, & I_{u-1} \geqslant 0 \end{cases} \tag{14}$$

I_u 表示第 u 个周期的剩余空闲时间,其计算公式如式(15)和式(16)所示。

$$I_u = p^{all}_{\min} - \sum_{i=1}^{j-1}\theta_u^j - \sum_{i=1}^{n}(e_i^j \times f_i) - L_u - \sum_{i=j+1}^{m} m_u^i \tag{15}$$

$$f_i=\begin{cases}1, & p_i^j=p_{\min}^{all}\\ 1, & r \bmod(\dfrac{p_i^j}{p_{\min}^{all}})=1\\ 0, & \text{上述条件不成立}\end{cases} \tag{16}$$

运用本文的算法得出的分区调度如图 5(b)所示，可以看到所有的任务均按时完成，而且也为关键级别较高的分区 1 预留了足够的资源和缓冲时间。各个分区在每个周期的执行时间的计算过程如表 5 和表 6 所列。

表 5　分区 1 执行时间的计算过程

u	fm_1	fs_1	f_1	fm_2	fs_2	f_2	fm_3	fs_3	f_3	fm_4	fs_4	f_4	fm_5	fs_5	f_5	L_u	I_u	θ_u^1
1	1	0	1	0	1	1	0	1	1	0	1	1	0	1	1	0	−2	16
2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	2	16	4
3	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	14	2
4	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	14	2

表 6　分区 2 执行时间的计算过程

u	fm_1	fs_1	f_1	fm_2	fs_2	f_2	fm_3	fs_3	f_3	L_u	I_u	θ_u^2
1	1	0	1	0	1	1	0	1	1	0	−16	4
2	1	0	1	0	0	0	0	0	0	16	−4	16
3	1	0	1	0	1	1	0	0	0	4	2	16
4	1	0	1	0	0	0	0	0	0	0	14	4

对比式(3)和式(4)的算法可以看出，本文的算法有着更高的系统利用率，由图 4 可以看到前者为了保证分区 1 的优先执行而为分区 1 预留了过多的资源，结果使得分区 2 无法得到有效执行。虽然文献[13]的算法通过按需分配执行时间避免了这一问题，但是其严苛的分区执行顺序导致了较差的分区可调度性，而本文的算法不仅在时间充裕的情况下可以保证高关键级别的所有任务都先于低关键级别的分区中的任务执行，而且在时间资源紧张的情况下仍然可以通过允许较低关键级别分区中的紧迫任务抢占较高关键级别中的非紧迫任务达到较高的可调度性，图 5 的调度示例展示了这一情况。本文开头提过，将关键级别较高的分区的任务提前执行是为了在任务运行超期时有足够的缓冲时间让系统进行处理以保证关键任务在死限时间之前完成。本文的算法为提高可调度虽牺牲了一点缓冲时间，但是其在保证可调度的情况下尽可能地为高关键级别分区的任务预留了缓冲时间，在图 5(b)的调度中，T_5^1 任虽然在第一个周期被抢占，但任有 56ms 的缓冲时间。如果系统对缓冲时间有特定要求，也可以使用预先保证缓冲时间的方法，即预留固定的时间进行缓冲，将剩余时间用来调度。

4　基于位运算的访问控制算法

第 3 节讨论了分区调度算法，本节将分析如何在内核中实现一个高效的、可以保障信息的完整性和机密性的访问控制算法，下面首先在 4.1 节中定义一个基于集合的访问控制模型，然后在 4.2 节中给出基于该模型的访问控制权限判断算法。

4.1　集合形式表示的强制访问控制模型

首先先来看一下传统的两个强制访问控制模型。

(1)BLP 模型：BLP 模型是一个适用于军事安全策略的计算机系统安全模型，其在 1973 年由 D. Elliott Bell 和 Leonard J. La Padula 提出。BLP 模型的安全策略包括两个部分：自主访问控制(Discretionary Access Control，DAC)策略和强制访问控制(Mandatory Access Control，MAC)策略。其 MAC 主要通过访问主体和客体的安全级别来表现，设定 $c(s)$ 表示主题的安全级别，$c(o)$ 表示客体的安全级别，符号 $\leqslant$ 表示安全级别之间的高低关系，那么 BLP 模型的 MAC 可以用下面的两个性质表示：

1)读请求控制：主体 s 对可以 o 的读请求只有在 $c(o)\leqslant c(s)$ 的时候才可以通过；

2)写请求控制：主体 s 对可以 o 的写请求只有在 $c(s)\leqslant c(o)$ 的时候才可以通过。

BLP 的读请求控制可以确保没有“向上读”的情况，写请求控制可以确保没有“向下写”的情况，满足这两个特性可以使信息不会从高安全级别的主体或客体流向低安全级别的主体或客体。但是 BLP 模型这种信息流向的控制只考虑了信息的机密性而没有考虑到信息的完整性。

(2)Biba 模型：Biba 模型可以用来保护信息的完整性，设定 i 为主体和客体的完整性标识，则 Biba 模型可以用以下两个性质来表示：

1)读请求控制：主体 s 对可以 o 的读请求只有在 $i(s)\leqslant i(o)$ 的时候才可以通过；

2)写请求控制：主体 s 对可以 o 的写请求只有在 $c(s)\leqslant c(o)$ 的时候才可以通过。

满足 Biba 模型的这两个特性可以防止信息从完整性较低的、不可靠的主体或客体流向完整性较高的客体或主体。

BLP 模型和 Biba 模型各有优点，它们都是通过主客体的安全属性级别(完整性或机密性)来决定信息的流向。下面给出一种综合考虑完整性和机密性，并将安全属性之间的比较转化为集合之间的包含关系的访问控制模型。

设定 S 为主体集合，O 为客体集合，$SO=S\cup O$ 为计算机系统的元素集合。用 CL 表示机密性级别的集合，IL 代表完整性级别的集合，CK 代表主体或客体的类别集合，$SC=CL\cup IL\cup CK$ 表示一个安全属性集合，集合 CL 和 IL 中的内部元素之间都存在偏序关系 $\leqslant$，而 CK 中的元素之间没有这样的关系。举例来说，假若 $CL=\{$绝密(α)，机密(β)，秘密(χ)，非涉密(δ)$\}$，$IL=\{$非常重要(a)，重要(b)，一般(c)$\}$，那么有 $\delta\leqslant\chi\leqslant\beta\leqslant\alpha$ 和 $c\leqslant b\leqslant a$。CK 可能为一些部门、功能的划分，比如 $CK=\{$作战部，指挥部，后勤部$\}$。下面根据上述这些集合的概念来定义访问控制模型和规则。

定义 1　三元组 $AC=(SO,SC,IR)$ 为访问控制上下文，其中 SO 为元素集合，SC 为关键属性集合，IR 是集合 SO 和 SC 之间的一个二元关系，对于任意一个属于 SO 的元素 x，以

及一个属于 SC 的安全属性 y，$xIRy$ 表示 x 拥有安全属性 y。

定理 1 $\forall x \in SO$ 和属于 SC 的安全属性即 $\forall y_1, y_2 \in SC$，如果 $y_1, y_2 \in CL$ 且 $y_1 \leqslant y_2$，那么当 $xIRy_2$ 成立时 $xIRy_1$ 也成立；如果 $y_1, y_2 \in IL$ 且 $y_1 \leqslant y_2$，那么当 $xIRy_1$ 成立时 $xIRy_2$ 也成立。

定理 1 将安全属性之间的比较转化为了集合的包含关系，对于 CL 来说，定理 1 意味着如果一个元素包含了一个较高机密级别的属性，那么它就自动包含了比此级别低的机密级别属性；而对于 IL 来说，当一个元素包含了一个较低完整级别的属性，那么它就也包含了比此级别高的完整级别的属性。现在设定函数 $f(x)$ 表示元素 x 的所有安全属性的集合，函数 $ck(x)$ 表示元素 x 的类别属性集合，$R(s,o)$ 表示主体 s 对客体 o 的读请求，$W(s,o)$ 表示写请求，那么就可以得到综合完整性和机密性的访问控制规则的集合表示形式，如表 7 所列。

表 7　访问控制规则

访问请求	允许条件	集合表示形式
R(s,o)	c(o)≤c(s) i(s)≤i(o) ck(o)⊆ck(s)	f(o)⊆f(s)
W(s,o)	c(s)≤c(o) i(o)≤i(s) ck(s)⊆ck(o)	f(s)⊆f(o)

4.2　权限判断算法

根据 4.1 节的访问控制模型，可以很方便地建立基于位运算的权限判断算法。假设在一个无人机的嵌入式系统中，有 4 个主体和 3 个客体。主体分别为整体系统检测进程 1、控制飞行角度的进程 2、接受远程战斗命令的进程 3、采集传感设备数据并回传的进程 4；客体分别为飞行控制器件、攻击武器、红外传感器，$ck=\{$战斗模块(FM)，飞控模块(FCM)，数据采集模块(DCM)$\}$，各个元素的安全属性标识如表 8 所列。

表 8　各元素的安全属性标识

主题或客体	机密级别	完整级别	类别
进程 1	α	c	战斗模块、飞控模块、数据采集模块
进程 2	β	b	飞控模块
进程 3	β	b	战斗模块
进程 4	δ	b	数据采集模块
飞行控制器件	χ	b	飞控模块
攻击武器	χ	b	战斗模块
红外传感器	χ	b	数据采集模块

本文的权限判断算法用布尔值来表示一个元素是否拥有某个安全属性，这样就可以用一个数值来表示元素对所有安全属性的拥有情况，根据定理 1，表 8 中各元素的安全属性标识可以用表 9 表示。

表 9　各元素的安全属性的拥有情况

名称	α	β	χ	δ	a	b	c	FM	FCM	DCM
进程 1	1	1	1	1	1	1	1	1	1	1
进程 2	0	0	1	1	1	1	0	0	1	0
进程 3	0	0	1	1	1	1	0	1	0	0
进程 4	0	0	0	1	1	1	0	0	0	1
飞行控制器件	0	0	1	1	1	1	0	0	1	0
攻击武器	0	0	1	1	1	1	0	1	0	0
红外传感器	0	0	1	1	1	1	0	0	0	1

为每个主客体都分配一个安全属性标识 SL，其为一个 2^n 位的数值，具体位数由表示的安全属性的个数决定，在上述的例子中有 10 个安全属性，因此可以设定 SL 为 16 位的数值，以进程 1 的 SL 为例，其各位的表示如图 6 所示。

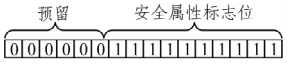

图 6　进程 1 的 SL 的示例

现在就可以使用 SL 之间的位运算来重新定义访问控制规则。

(1)读请求控制：主体 s 对可以 o 的读请求只有在 $S_{SL}\&O_{SL}=O_{SL}$ 的时候才可以通过；

(2)写请求控制：主体 s 对可以 o 的写请求只有在 $S_{SL}\&O_{SL}=S_{SL}$ 的时候才可以通过。

5　实验分析与验证

为了验证实时系统内核和算法的性能，在 Virtex-5 FXT FPGA ML507 评估平台上对系统内核进行了测试，ML507 使用的是 PowerPC440 的 CPU，有 256MB 的内存，PowerPC440 完整地实现了 32-bit Book-E Enhanced PowerPC 架构，并且有完善的内存管理单元(MMU)。在系统的测试过程中，系统内核占有 64MB 的内存，其余的内存按照系统配置分配给相应的分区，系统的主要时间参数如表 10 所列，其数据都是多次测试的平均结果。结果表明，其可满足大多数实时嵌入式系统的要求。

在任务运行的过程中，如果任务运行超过死限时间，则系统的健康监控模块就会调用相应的错误处理程序，不同的错误和其处理程序之间的映射关系都体现在系统级、模块级和分区级的 3 张健康监控映射表中，在系统测试过程中注入的不同级别的多个错误均得到了正确的处理。

表 10　时间参数

参数名称	时间(μs)
分区切换时间	12.4
任务切换时间	3.2
中断响应时间	0.7

为了验证访问控制算法的性能，测试了 20 个不同 API 在加入访问控制前后的运行时间，如图 7 所示。加入访问控制之后，平均运行时间增加了 0.12μs，各 API 增加时间的方差为 1.84×10^{-6}。因此可以看出，与文献[8,9]的算法相比，本文提出的算法的执行时间更稳定，效率更高。

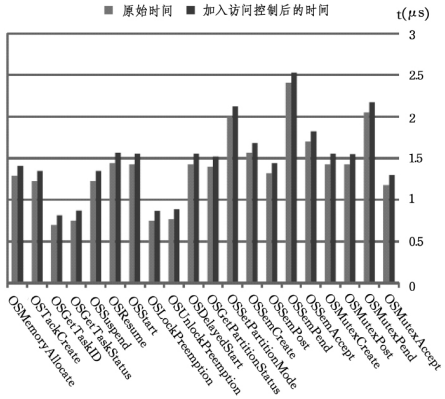

图 7　访问控制时间对比

(下转第 550 页)

的提高，从表 1 即可得到例证。因此，运用本文算法来预测时用水量具有较高的精度和可靠性。

结束语 本文分析了 BP 神经网络算法作为时用水量预测算法的不足，并提出使用遗传算法对 BP 神经网络进行优化的思路；分析历史用水数据的变化规律和影响因素，选取了适当的输入变量组成样本集，并且经过多次尝试和比较，选取了合适的网络参数；最后，将深圳某用水大户的用水数据分别应用于本文建立的预测模型、传统的 BP 神经网络预测模型以及小波神经网络这种改进的 BP 神经网络预测模型中进行仿真。结果表明，本文算法相对于传统 BP 神经网络方法收敛速度快，且其预测精度明显高于 BP 神经网络模型和其他改进的 BP 神经网络模型，克服了 BP 神经网络算法的不足，有较高的可靠性和较好的应用性。

参 考 文 献

[1] 刘洪波，郑博一，蒋博龄. 基于人工鱼群神经网络的城市时用水量预测方法[J]. 天津大学学报(自然科学与工程技术版)，2015(4)：373-378

[2] Cheng Q，Ni-Bin C. System dynamics modeling for municipal water demand estimationin an urban region under uncertain economic impacts.[J]. Journal of Environmental Management，2011，92(6)：1628-1641

[3] 董长虹. Matlab 神经网络与应用[M]. 北京：国防工业出版社，2003：35-48

[4] 张宏伟，牛志广. 神经网络法建立城市供水管网宏观模型的研究[J]. 系统工程理论与实践，2003，23(10)：121-126

[5] 周艳春，李树平，赵子威，等. 基于 BP 神经网络工具箱的城市短期用水量预测[J]. 给水排水，2015(S1)：375-377

[6] 陆健. 基于 BP 神经网络和遗传算法的城市供水系统优化调度模型研究[D]. 河海大学，2007

[7] 储诚山，张宏伟，郭军. 基于遗传算法和 BP 神经网络的用水量预测[J]. 中国农村水利水电，2006(4)：36-38

[8] Dybowski R，Weller P，Chang R，et al. Prediction of outcome in critically ill patientsusing artificial neural network synthesised by genetic algorithm [J]. Lancet，2008，18(7)：545-9

[9] 周明. 遗传算法原理及应用[M]. 北京：国防工业出版社，1999：23-48

[10] 王德明，王莉，张广明. 基于遗传 BP 神经网络的短期风速预测模型[J]. 浙江大学学报(工学版)，2012(5)：837-841，904

[11] 金龙，吴建生，林开平，等. 基于遗传算法的神经网络短期气候预测模型[J]. 高原气象，2005，24(6)：981-987

[12] 刘洪波，张宏伟，田林. 人工神经网络法预测时用水量[J]. 中国给水排水，2002，18(12)：39-41

[13] 袁伟，陈晓东. 基于 GA-BP 神经网络与 LSSVM 支持向量机的日用水量组合预测模型[J]. 水电能源科学，2015(10)：33-37

[14] 韩力群. 人工神经网络理论、设计及应用[M]. 北京：化学工业出版社，2007：47-66

[15] 崔丽杰. 基于 BP 神经网络的训练函数选取研究[J]. 科技创新导报，2014(36)：27-27

[16] 陈丽琳. 基于多嵌入维数的时用水量 LSSVM 组合预测[J]. 机电工程，2012，29(7)：869-872

(上接第 546 页)

结束语 本文在 μC/OS-II 的基础上加入分区机制和访问控制机制，扩展了原有系统的任务调度功能，并在此基础上提出了新的分区调度算法。该算法不仅可以保证低关键级别分区中的任务在时间充裕的情况下不会抢占高关键级别分区中的任务，而且可以提供较高的系统吞吐量和任务可调度性，使得整个系统变得更加可靠、安全、耐用。因此，本文提出的内核设计具有很大的应用价值，同时也存在一定的进步空间。可以对以下几个方面进行更深入的研究：1)更加复杂的任务模型，以及任务间协作和资源共享等复杂场景下的系统可靠性；2)适用范围更广的访问控制模型。

参 考 文 献

[1] Han S，Jin H W. Resource partitioning for Integrated Modular Avionics：comparative study of implementation alternatives[J]. Software Practice & Experience，2014，44(12)：1441-1466

[2] Ananda C M，Nair S，Mainak G H. ARINC 653 API and its application-An insight into Avionics System Case Study[J]. Defence Science Journal，2013，63(2)：223-229

[3] Xiao G，Qu Z，He F. Design and realization of IMA simulation platform based on CPCI bus using VxWorks653 RTOS[C]// 2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC). IEEE，2015：10A1-1-10A1-8

[4] Wan M，Tian S. Research on Schedulability of Partition Scheduling for IMA[C]// 2011 Fourth International Symposium on Computational Intelligence and Design (ISCID). IEEE，2011：322-325

[5] Kurowski K，Oleksiak A，Piatek W，et al. Hierarchical scheduling strategies for parallel tasks and advance reservations in grids[J]. Journal of Scheduling，2013，16(4)：349-368

[6] Carnevali L，Pinzuti A，Vicario E. Compositional Verification for Hierarchical Scheduling of Real-Time Systems[J]. Publication，2013，39(5)：638-657

[7] Boudjadar A，David A，Kim J H，et al. Hierarchical scheduling framework based on compositional analysis usinguppaal[M]// Formal Aspects of Component Software. Springer International Publishing，2014：61-78

[8] 邓集波，洪帆. 基于任务的访问控制模型[J]. 软件学报，2004，14(1)：76-82

[9] Liu Y，Chen X. A new information security model based on BLP model and Biba model[C]// 2004 7th International Conference on Signal Processing，2004(ICSP '04). IEEE，2004：2643-2646

[10] 崔可明. uC/OS-II 的安全访问控制关键技术研究[D]. 哈尔滨：哈尔滨工程大学，2006

[11] 李大明，曹万华，张焕. 基于可变标签的访问控制策略设计与实现[J]. 计算机科学，2012，39(12)：290-294

[12] Labrosse J J. μC/OS-II：a Real Time Kernel[M]. Electronic Engineering & Product World，2007

[13] Jin H W，Han S. Temporal partitioning for mixed-criticality systems[C]// 2011 IEEE 16th Conference on Emerging Technologies & Factory Automation (ETFA). IEEE，2011：1-4