嵌入式实时操作系统 eCos 内存分配策略的分析

董明峰 谷建华

(西北工业大学计算机学院,陕西 西安 710072)

摘 要: 本文全面深入的探讨了 eCos 中内存分配策略的实现。首先,对 eCos 内核及内存管理方法进行了介绍,然后对四种内存分配策略进行了阐述和性能比较。对于嵌入式系统和应用的开发具有一定的参考价值。

关键词: 嵌入式,实时操作系统,eCos,内存管理

中图法分类号:TP316 文献标识码:A 文章编号:1000-7180(2004)07-120-04

The Analysis of Memory Allocation in the Embedded Real-time Operating System eCos

DONG Ming-feng, GU Jian-hua

(College of Computer, Northwestern Polytechnical University, Xi'an 710072 China)

Abstract: With the embedded system developing rapidly, the real-time operating system eCos has been widely used in embedded application. The implementation method of memory allocation is analyzed in this paper. Firstly, it introduces the eCos kernel and memory management package. Then it explains and compares the four apposite policies of memory allocation. It is a good reference for embedded system development.

Key words: Embedded, RTOS, eCos, Memory management

1 引言

近些年来,随着嵌入式系统的飞速发展,嵌入式操作系统在网络通信,工业控制,通讯,国防,航空航天等各个领域得到了广泛的应用,并越来越引起人们的重视。目前,国内比较流行的嵌入式操作系统有 Windows CE, Nucleus, VxWorks, Psos, eCos 等等。因为嵌入式设备存储器的容量较小,而对系统性能和可靠性的要求比较高,其中内存的管理与分配在嵌入式操作系统中的地位尤为重要,所以剖析 eCos 的内存分配算法并对其进行优化就具有了重要的意义。

2 eCos 内核简介

eCos(embedded configurable operating system)是 Red hat 公司开发的一个可裁剪的嵌入式实时操作系统。与其它商业操作系统相比,其主要优点是源代码完全开放,开发人员可以重新编译或修改系统的任一部分。像 Linux 一样,全世界有众多的自由软件爱好者在支持着 eCos,eCos 的源码几乎每天都在更新。

收稿日期: 2004-01-13

基金项目: 国家 863 资助项目(2003 AA1z2100)

西北工业大学研究生创新基金支持(M016110)

eCos 支持大部分主流的 CPU,目前已经成功移植到 ARM7、PowerPC、MIPS 等多种 CPU 的体系结构上。eCos 还给用户提供了从操作系统配置到源代码编译、应用程序调试的图形化集成开发环境,极大的方便了嵌入式应用程序的开发和调试。例如 eCos 提供了裁剪配置工具 Configuration Tool, 已经作为 eCos 发行版本的有机组成部分,它为用户管理配置信息提供了可视化的用户界面,极大地减轻了配置工作的复杂性。

eCos 的核心是一个功能全面的,灵活的,可配置的实时内核。这个内核提供了多线程支持,多种调度器的选择,一组丰富的同步原语,内存分配原语和线程管理原语。在这个内核中,改变或替换其中的某些部分(比如调度器)而不会影响内核中的其他模块。

eCos 内核的一个的重要特征是可以选择合适的内存分配算法。

3 eCos 内存管理机制

作为一个嵌入式操作系统,eCos 的内存管理相对简单,既不分段也不分页。eCos 的内存管理通过内存池对象来实现。系统提供了模板类 Cyg_Mempolt2,Cyg_Mempolt2 在多线程访问内存时提供了线

程安全的保护,数字 2 是为了区别过去曾经使用过模板类 Cyg_Mempoolt[1]。模板类实例化了两种可选的内存池对象,eCos 通过两种内存池类来实现两种不同的内存管理方法，一种是固定块大小的内存池，即内存池以固定大小的块为单位进行分配,使用位图进行管理；另一种是可变块大小的内存池,即内存池根据申请的内存尺寸分配,使用链表进行管理。

eCos 根据用户的选择,生成一个预编译宏指示相应的头文件。以可变块大小内存管理为例，如果用户选择该管理方式，宏 CYGBLD_MEMALLOC_MALLOC_IMPLEMENTATION_HEADER 就指向头文件 memvar.hxx，在该头文件里，声明了模板类 Cyg_Mempolt2 的一个对象，并用宏 CYGCLS_MEMALLOC_MALLOC_IMPL 确定内存管理的实现采用可变块内存管理算法[2]。

eCos 允许对拥有所有可用内存的内存池进行自动定义,可以自动分配堆的大小。要实现这一功能,必须使用 eCos 配置工具中的内存布局工具对用户定义段(user-defined section)进行定义。这些段的名字都是以“heap”为前缀,如“heap1”、“heap3”、“heapram”等等。用户定义段可以是固定大小,也可以是一个未知的大小。

在配置目标操作系统时,用户不仅可以自行配置内存池的大小，也可以利用 Cyg_Mempool_Joined 类模版将多个物理上不连续的内存池封装成一个内存池。在具有多个不连续的内存并且没有存储管理单元(MMU)将它们连接成一个连续的内存空间时,可以定义多个堆段。系统将每一个堆段的定义形成一个列表,根据该表可以确定使用的是哪一个内存池。

eCos 的内存管理不仅提供了内存的分配 try_alloc( ),改变内存块大小 resize_alloc( ),获取内存信息 get_status( )等内核 API,也提供 ANSI C 的标准内存操作接口如 malloc( ),realloc( ),free( )等等。默认情况下,eCos 的 malloc C 库函数使用可变块大小内存池来实现内存分配的。

eCos 支持四种内存分配策略的实现，分别是:固定块大小内存分配 (fixed block memory allocators)，简单可变块大小内存分配 (simple variable block memory allocators),分离数据元可变块大小内存分配 (sepameta variable block memory allocators)，Doug Lea 内存分配算法 (Doug Lea's memory allocators)。

4　内存分配策略的实现

4.1　固定块大小内存分配

固定块大小的内存池使用位图来进行管理。系统将一个内存池分为固定大小（可在配置时设定）的块,根据应用程序的需求,计算出所需内存的总块数。从内存池的基址开始,初始几个内存块建立一个位图数组,它的每一个元素是 32 位的整型数,每一个整型数的一位对应内存池的一块,未分配则该位为 0,已分配该位为 1。一个位图的状态如图 1 所示。

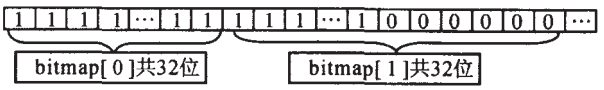

图1 内存块位图的结构

在实现类 Cyg_Mempool_Fixed_Implementation 中附加定义了位图基址,块大小等信息,由于是固定块大小的内存分配，在内存池创建程序 Cyg_Mempool_Fixed_Implementation 中,计算并确定可分配的块数和位图数组的大小。内存分配函数 try_alloc 分配一块 blocksize 大小的内存，返回该块内存指针 p,简略程序如下。

```
do {
    if ( 0xffffffff ! = bitmap[ i ] ) {
        //条件成立,说明在此 bitmap[ i ]中有指向空闲块的位
        register cyg_uint32 j, k;
        k = ~bitmap[ i ];        // 各位取反,这样未分配的块标记便为 1
        HAL_LSBIT_INDEX( j, k );  //返回 k 各位中的每一个为 1 的位的序号
        bitmap[ i ] |= (1 << j); // 把该位设置为已分配
        firstfree = i;
        freeblocks--;
        p = &mempool[ ((32 * i) + j) * blocksize ];
        break;  }
        if ( ++i >= maptop )    //i 增 1,对下一位 bitmap 进行考察
        i = 0;                  // 内存分配完
  } while (i ! = firstfree);// 利用 i 值对位图数组循环遍历
```

4.2　简单可变块大小内存分配

可变块大小内存池使用链表来进行管理。eCos 系统为空闲内存定义了一个双向链表的结构,如图 2 所示。内存中每一个空闲块都有一个 memdq 结构头指针,它包含着指向前后空闲内存块的指针和该内存块的大小信息，它存储在每一内存块的头几个字节当中。类 Cyg_Mempool_Variable_Implementation

中定义了链表的节点数据结构：

```
struct memdq {
    struct memdq *prev, *next;
    cyg_int32 size;    }
```

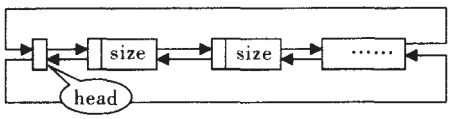

图2 简单可变内存块链表结构

空闲链表是一个有序的循环双链表，链表中的每一个节点都代表一个空闲块。空闲链表的头指针有同样的结构但它的大小域为零。

eCos 可变块大小分配模式采用的分配策略是最先适应(First fit)算法，空闲内存块按地址大小递增排列。对于要求分配的分区容量 size，从头开始比较，直至找到满足大小≥size 的块为止，并从链表中相应块中分配出相应 size 大小的块指针。

内存分配函数 try_alloc 的参数是需要内存块的大小 size，返回分配的内存块首地址。它首先查找空闲链表，找一块足够大的块。如果找到一块大小正合适的块，就把这一块从链表中卸下来，并返回一个指向该块的指针，程序如下：

```
dq->prev->next = dq->next;
dq->next->prev = dq->prev;
alloced = (cyg_uint8 *)dq;
```

如果找到一块比所需内存大的块，我们将这块内存从末端开始分为两块，返回后一块的指针。所有的被分配的内存块的前后指针都指向地址 0xd530d53，以便于对分配的内存块统一管理。

4.3 分离数据元可变块大小内存分配

eCos 系统为对具有分离数据元的可变块大小的内存进行管理，运用了两个双向循环链表的结构，一个是已分配块的，另一个是空闲块的链表，如图 3 所示。同时还有一个数据元池中的无用数据元链表。内存中每一个空闲块都有一个结构头指针，结构中有指出它的所在的已分配或者空闲链表中前一块的指针，还包括它实际内存中它的前一块和后一块的地址。类 Cyg_Mempool_Sepmeta_ Implementation 中定义了内存块节点的数据结构。

```
struct memdq {
    struct memdq *prev, *next;        //指出逻辑上相邻的已分配的或空闲的块
    struct memdq *memprev, *memnext; // 指出实际内存中相邻前一块和后一块
    cyg_uint8 *mem;        // 这一块内存的相关首地址
}
```

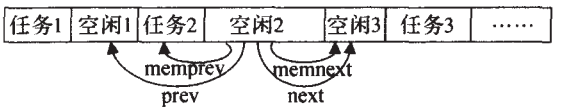

图3 分离数据元可变内存块链表结构

内存分配的算法和可变块大小内存分配的思想相似，try_alloc 在找到合适的内存块后将其从空闲链表中取下后，然后紧接着要将其插入已分配的链表中，程序如下所示。

```
dq->next = dq->memnext;        // 再把这一块插入到已分配的链表当中
dq->prev = dq->next->prev;
alloced = dq;
```

这里应注意，因为 dq 是空闲块，那么它紧跟着的内存块，即：dq->memnext 块一定是已分配的，否则，它在插入之前就已经和 dq 块合并为一。

4.4 Doug Lea 内存分配

Doug Lea 内存分配算法是当前比较经典的内存分配算法，eCos 将它移植到系统中来。该算法的思想如图 4 描述所示。将所有的内存块都放在容量仓(bin)中，系统一共有 128 个固定大小的容量仓。对于小于 512 字节(默认值)的容量仓，每个仓内存放固定大小的内存块，内存块的大小从 16 字节开始，以 8 字节递增，直到 512 字节为止。大于 512 块的内存块，按块的大小由小到大排列。

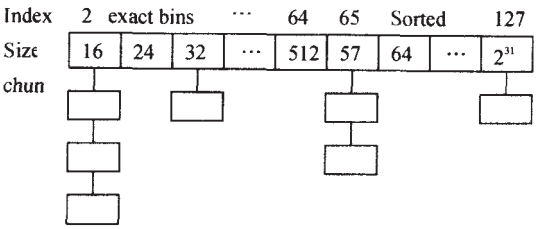

图4 Doug Lea内存分配算法容量仓

系统在执行 try_alloc 分配内存时，先检测有没有正好合适的容量仓中的内存块可以分配，如果没有，则检测最近使用的内存块的剩余部分有没有足够大的空间来分配。这样的分配方法充分利用了程序的局部性特征，有利于减少内存碎片。

对于大块(默认≥512 字节)的内存的请求，内存块按从小到大的顺序排列，按最佳适应算法(best-fit)这种方式来搜索内存块。系统剩余的靠近内存池边界的空闲内存(wildness memory)可以很大(取决于系统的限制)，所以把它看作是一个比所有的内存块都大的块并运用在最佳适应算法中[3]。

另外 Doug Lea 内存分配算法考虑到了很多特殊的应用。对于小块(默认≤64 字节)的内存请求，它

采用了一种依靠缓冲存储器的分配算法,充分考虑了延迟合并和预分配等情况从而提高系统效率。对于非常大的内存块请求(默认≥128KB),Doug Lee 算法可以支持内存图的方法,允许为应用程序分配一块非连续的内存区域, 目前移植入 eCos 系统 Doug Lea 算法暂不支持内存图的应用。

5 性能比较

固定大小内存块的实现显然最简单,malloc ( ) 和 free( )函数的执行时间是固定的,并且消除了内存碎片的问题,在一些实时性要求较高的系统中有一定的应用。但不同应用环境下分配固定内存块可能会对系统内存造成浪费。

简单可变块大小内存分配和分离数据元的内存分配都采用最先适应算法, 它在释放内存分区时,能够很方便的使相邻的空闲块合并,这一点在函数 insert_free_block( )函数中可以很清楚看到。算法的实质是尽可能利用了存储器的低地址部分,在高地址部分则保留较多的或较大的空闲区,以后如果需要较大的内存,比较容易满足。然而,不难看出,算法的缺点是在低地址部分很快集中了许多非常小的内存块,因而在内存分配时,搜索次数增加,影响了工作效率[4]。

Doug Lea 内存分配采用了多种内存分配策略的组合,尽可能得达到了各种指标的平衡。针对不同情况,算法会很快的使用尽可能最好的分配方法来满足内存的需求的。它主要采用了最佳适应算法,相比较其他的算法产生的内存碎片最少,适合于比较复杂的嵌入式应用。但 Doug Lea 内存分配算法的实现相对复杂,占用资源比较多,最佳适应算法也可能会在系统中产生过多的小的空闲区。

6 结束语

eCos 一是种源码开放的嵌入式实时操作系统,它的应用大大提高了嵌入式系统开发的效率。作为一个实时的嵌入式系统,eCos 内核不仅要占用尽量少的内存开销,内存分配占用尽量少的资源和 CPU 周期,并选择适当的分配算法。本文剖析了 eCos 内存管理中多种内存分配策略的实现,并进行了分析和比较,希望对有从事嵌入式系统应用和开发的同行们有所帮助。

参考文献

[1] Memory allocation package – Implementation Notes,http://sources.Redhat.com/ecos/

[2] 秦怀峰.面向嵌入式 PC 的 eCos 支持与完善[D],西北工业大学硕士毕业论文,2002.

[3] Doug Lee.A Memory Allocator[J].http://g.oswego.edu/dl/html/malloc.html

[4] 徐甲同,陆丽娜,谷建华,计算机操作系统教程[M],西安,西安电子科技大学出版社,2001.2

董明峰　男,(1980–),硕士研究生。主要研究方向为嵌入式计算,嵌入式操作系统。

谷建华　男,(1965–),教授。主要研究方向为嵌入式计算,网络和分布式计算。

(上接第 119 页)

对新一代无线系统的研制、实现,有一定参考作用。

参考文献

[1] Bhagwat P, Perkins, and Tripathi S. Network layer mobility: an architecture and survey, IEEE Personal Communication [J], June 1996,3(3):54–64.

[2] J Mitola., Software Radio Architecture: A Mathematical Perspective [J]. IEEE Journal on Selected Areas in communications, 1999,17(4):514–538.

[3] K.Ducatel, M.Bogdanowicz,et. Secenarios for Ambient Intelligence in 2010, Feb. 2001 IPTS–Selle, http://www.cordis.lu

[4] Peter M. Corcoran et al, Mapping home–network application to TCP/IP sockets using a three–tiered home gateway architecture, IEEE Trans. on consumer electronics [J], Aug 1998,44(3):729–736.

[5] Evangelos Topalis et al, A generic network management architecture targeted to support home automation networks and home internet connectivity [J], IEEE Trans. on consumer electronics, Feb. 2000,46(1):44–50.

[6] 谢谦等.计算机网络实验教程[M].北京:电子工业出版社,2000:20–85.

陈旭辉　(1972–),博士研究生。主要研究方向为无线网络通信、家庭网络技术、无处不在计算等。

侯义斌　博士生导师。主要研究方向为广义人机交互计算机网络理论与应用。