UC/OS-II 中动态内存管理方案的改进与实现

梁乘铭，韩坚华，夏成文，覃毅

（广东工业大学，广东省广州市 51006）

批注 [U1]: 该文研究 UC/OSII 实时操作系统 DSA 的改进，具备一定的实用性，在 TLSF 算法的移植过程中，根据 RTOS 的需求做了定制，文章结尾给出了移植实验结果数据。文章具有一定的参考价值。

摘要：UC/OS-II 原有内存管理模块存在着较为普遍的内存碎片，无法提供接口查找合适尺寸的内存块，且扩充动态内存量不灵活，总的功能较弱。该文根据 RTOS 中 DSA（Dynamic Storage Allocation）的高效、快速、可靠、方便与节约的需求，分析、采纳并定制了 TSLF(Two-level Segregated Fit)在 UC/OS-II 上的应用。最后，做了对比实验，实验数据表明了 UC/OS-II 新的 DSA 较原有 DSA 具有较大的改进效果。

关键字：UC/OS-II；动态内存管理；动态存储算法；TLSF；实时操作系统

中图分类号：TP316.2　　**文献标识码：**A

Improvement and Realization of DSA in UC/OS-II

LIANG CHENG-MING, HAN JIAN-HUA, XIA CHENG-WEN, QIN YI

（GuangDong University of Technology, 510006,China）

Abstract: It exists common memory fragmentation, can't provide methods to find out the suitable memory block to allocate, and is hard to extend the amount of memory, provides weak function wholly in the original UC/OS-II's DSA. This paper analyses, adopts and designs DSA of TLSF(Two-level Segregated Fit) in UC/OS-II, according to requirements of high efficiency, fast response time, reliability, convenience to use, and saving. Experiment data are presented, and the result shows the harvest of improving on the DSA in UC/OS-II.

Key words: UC/OS-II; Dynamic Memory Management; DSA; TLSF; RTOS

1、 引言

嵌入式系统的内存资源相当有限，所以需对其进行合理的规划和管理，即需要满足其管理特点：[1]快速性、可靠性和高效性。

随着嵌入式应用软件规模的增长，人们希望 DSA 在满足以上特性的同时，更能够被方便而充分地使用。而 UC/OS-II 的 DSA 功能较弱，所以对其进行改进是很有必要的。

2、 RTOS 的 DSA 概览【2】【3】【4】【5】

按照记录以及合并空闲内存块的方法，可将 RTOS 的 DSA 分成顺序搜索、索引搜索、分类搜索、位图搜索以及伙伴算法等。这些 DSA 算法都具有分配真实内存，立即合并空闲内存等特点。

2．1 顺序搜索算法

顺序搜索算法采用单向或双向链表维护空闲内存。该算法的时间花费与空闲内存链表长度成正比，时间花费是有界的，但会随着空闲内存块增多而增加，在嵌入式实时系统中并不宜使用。

2．2 索引搜索算法

索引搜索算法用一种比链表复杂的数据结构来记录空闲内存。常见的如排序二叉树，树中每一个节点代表某一个尺寸的空闲内存，存储该尺寸的空闲内存链表指针。索引搜索算法的数据结构和分配、合并内存较为复杂。

2．3 分类搜索算法

分类算法把所有空闲内存按其尺寸范围划归不同的类，同一类内的内存块链接成一个空闲自由内存链表。所有的空闲内存头指针统一由另一个数组链表维护，每个空闲内存头指针对应该数组一个元素。值得注意的是，属同一类的自由内存，并不要求其物理上是相邻的。

分类搜索算法中的链表可以是按空闲内存尺寸排序的，也可以是不排序的。分类算法较为复杂，但不必搜索即可查找合适空闲内存，时间花费不随空闲内存的数量而变化，适合于嵌入式系统采用。

2．4　位图搜索算法

位图搜索算法用一个位图来查找空闲内存，该算法查找空闲内存所需的信息全部存储在一小块内存中，查找响应速度很快。

3、　UC/OS-II 的 DSA 不足之处

UC/OS-II 中的内存管理模块把动态管理的内存分成多个内存区，每一个内存区又分成一定数量相同尺寸的内存块。具体的 UC/OS-II 中，DSA 由 OS_MEM.c 实现，总共只包含 5 个函数 OSMenInit,，OSMemCreate，OSMemGet，OSMemPut 与 OSMemQuery，约 100 行代码，十分精炼。正是由于其精炼，UC/OS-II 的 DSA 提供的功能十分有限，存在以下不足：

1） 动态管理的内存块尺寸须在编译时指定，运行时不能更改，限制了系统以后扩展应用程序的灵活性，也造成内存浪费。

2） 由于同一分区只能提供唯一尺寸的内存块，而应用中一般需使用到不同尺寸的内存块。为了减少资源浪费，此时则需建立两个以上的内存区，加大了维护开销。

3） 不可能提供确定不同内存区的内存块之间尺寸差距的方案，使内存的浪费不可避免。这是由于系统中可能的应用千变万化，而他们申请的内存块尺寸也不尽相同。

4） UC/OS-II 的 DSA 可以归类为 2.3 中的分类搜索算法，但其并未提供如何搜索到合适分类的方法，也未提供向某一分类申请内存失败后如何向下一分类申请内存的方法，而需要程序员自己提供，加重了程序员负担的同时更是降低了程序的可靠性与稳定性。

4、　TSLF 的数据结构介绍和算法分析【2】

TLSF 是 M.Masmano 在 IEEE 的 Euromic 的会议上提出的，用于支持嵌入式实时系统的动态内存管理，它结合了 2.3 分类搜索算法与 2.4 位图搜索算法的优点，速度快、内存浪费少，所用的数据结构如图 1。

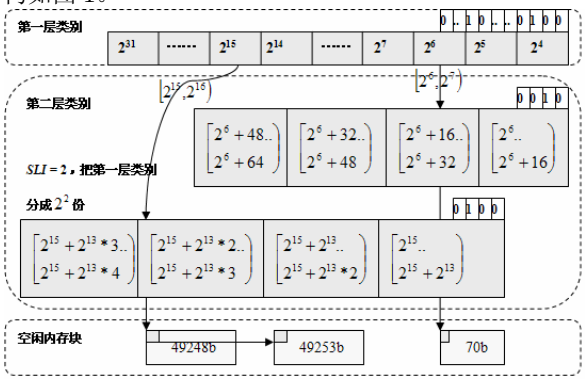

图 1 TLSF 数据结构

TLSF 用两个层次的分类对不同尺寸的内存块进行分类。第一层次的类别目录为 2^n，n 为 4，5，……，31 的整数，称为 FLI（First-level Segregated Fit）。每一个 FLI 类别又根据第二层的 *SLI* 细分为 2^{SLI} 个子类别。第二层的每个类别，都对应一条属于该类别尺寸范围内的内存块链表。为了加快分配与合并内存块的速度，链表是不排序的。所有的链表头指针用数组元素尺寸为 32 位的二维数组存储起来。各个类别所表示的内存块尺寸范围可参见图 1。第一层次、第二层次都使用位图指示该类别有无空闲内存块，有则该类别对应的位为 1，否

则为 0。

4．1　TLSF 位图与 TLSF 头指针

TLSF 中每一个第一或第二层次的类别对应位图中的 1 位，位为 1 表示该类别有空闲内存块，为 0 则表示没有。可根据第一、第二类别的总数确定总的位图存储空间大小。位图存储在内存池的开始位置。

TLSF 第二层次的每一类别皆对应一个头指针。若该类别的空闲内存块链表非空，则类别头指针指向该链表，否则头指针为空。

4．2　TLSF 块头

TLSF 的空闲内存块与使用中的内存块块头并不相同，如图 2 所示。

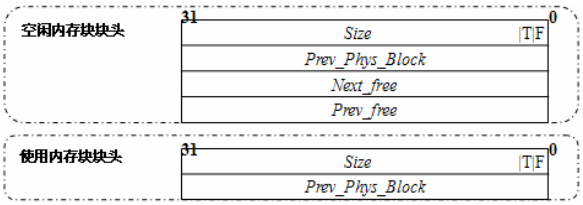

图 2　内存块块头数据结构

TLSF 中所用的内存块由两条链表组织。逻辑链表：每个第二层次的类别可有 0 条或 1 条，它是一个双向链表，把属于该类别的所有内存块不排序地逻辑上链接在一起；物理链表：把所有空闲与非空闲内存块按物理地址相邻链接起来。

4．3　TLSF 算法分析

参考文献[2]的推导，TLSF 的 malloc，free 的时间复杂度并不随空闲内存块的数量而变化，都是 O(1)。

4．4　TLSF 的碎片

由于 TLSF 的分类内的自由空闲内存链表是不排序的，分配时也不搜索，所以申请尺寸属于某一类别的内存块时，却要从下一个类别分配内存[2]。TLSF 内存碎片的计算公式为：

$$fragmetation = 2^{FLI} / 2^{SLI} - 2$$

4．5　TLSF 参数的控制

TLSF 有 3 个可以配置的参数常量。

FLI：是第一层次类别的数目，类别都是 2 的 n 次方。FLI 最大可去到 31，

$$FLI = \min(\log_2(memory\_pool\_size), 31)$$

SLI：是第二层次类别的数目。出于性能考虑，*SLI* 必须是 2 的 n 次方，并且范围在[1, 32]之间，以便用单字处理指令。一般，*SLI* 用第二层次类别数目的 $\log_2$ 来表示，如 *SLI*＝2 则表示第二层次类别把对应的第一层次类别分为 4 份。

MBS(Minimum block size)：最小块的尺寸，一般为 16Bytes。

5、　TLSF 向 UC/OS-II 的移植定制

为了克服 UC/OS-II 原有 DSA 的不足，本文引进了 TLSF 动态内存管理算法，并做了适当的修改以便适合于 UC/OS-II。

由于 TLSF 可以在同一内存池分配不同尺寸的内存块，为了充分发挥 TLSF 这一优点、减少管理开销，在其移植后只使用物理地址连续的一块内存。在 TLSF 的移植过程中，仿照了 UC/OS-II 系统的风格，把其定制成可裁减的模块，只有配置了相关常数后，才编译该模块。提供的编程接口函数与配置常数如表 1。

函数	功能	时间复杂度	在 OS_CFG.H 配置常数为 1，表示允许
tlsfmalloc	类似 c 语言的标准函数 malloc		

tlsf_free	类似 c 语言的内存函数 free	O(1)	
tlsf_init	初始化内存池	O(1)	
tlsf_destroy	销毁内存池	O(1)	OS_MEM_DESTROY

表 1　UC/OS-II 的 TLSF 编程接口函数与配置常数

根据 UC/OS-II 的一般应用，还在 OS_CFG.h 中配置了 FLI=20（动态管理 1MB 内存），*SLI*=2,MSB=16。根据需要动态管理的内存池大小可以调整 FLI，要降低内存块粒度减少内存浪费可以增大 *SLI*。

由于 TLSF 的内存块是动态分割与合并的，因此其尺寸在运行时可变，所以其内存利用率高，内存分配可靠。

6、 UC/OS-II 中的 TLSF 动态内存管理模块实验分析

本文做了一个对比实验，比较 UC/OS-II 使用原有的 DSA 与使用 TLSF 的性能差异。为了便于和 TLSF 对比，实验条件为：1）原有 DSA 只使用一个内存分区，其只能提供唯一尺寸的内存块。2）实验所用程序为一个 DSA 基准测试专用程序 cfrac [6]，该程序分解给定数字的因数，在此过程中向系统申请/释放内存，本实验要分解的数是 23533。3）用于动态管理内存区 heap 的尺寸为 256KB。4）两个 DSA 都做 10 次实验，最后对结果求平均值。

对比表 2 的分配的内存块最大尺寸与分配的内存块平均尺寸可以看出，虽然 UC/OS-II 原有 DSA 与 TLSF 均使用了一块 256KB 的内存作为动态内存，但只有 TLSF 可以提供不同尺寸的内存块。这就让 TLSF 获得更高的内存利用率。从 TLSF 的内存最大使用量标志与内存平均使用量标志均是 UC/OS-II 原有 DSA 的 10 倍，也可以得出以上结论。TLSF 的 *SLI* 是 2，粒度是比较细的，所以平均分配尺寸较小。

虽然 TLSF 的执行每个 malloc 平均指令数与执行每个 free 平均指令数均是 UC/OS-II 原有 DSA 的 5 倍以上，即 TLSF 要比 UC/OS-II 原有 DSA 耗时，但这是以牺牲内存空间为代价的。这是由于此实验的 UC/OS-II 原有 DSA 并未使用搜索合适内存区的相关算法，内存资源浪费很大，在程序规模不断增长而要求更节约内存时，UC/OS-II 原有 DSA 这两个指标也会有明显增大。

DSA	TLSF	原有 DSA	DSA	TLSF	原有 DSA
内存最大使用量 maxUsed（Byte）	1191	10680	内存最大使用量标志（heap/maxUsed）	220.1	24.5
内存平均使用量 avgUsed（Byte）	558	5819	内存平均使用量标志（heap/avgUsed）	469.4	45.0
分配的内存块最大尺寸（Byte）	268	280	执行每个 malloc 平均指令数	526.0	60
分配的内存块平均尺寸（Byte）	30.5	280	执行每个 free 平均指令数	325.9	58

表 2　TLSF 实验结果数据

此外，对比文献【2】【7】【8】中 TLSF 相关的数据，也可知在 UC/OS-II 中实现的 TLSF 动态内存管理算法获得较好的性能。

7、 结论

本文创新地在 UC/OS-II 中实现了一种使用方便、高效、快速、可靠与节约的动态内存管理方案 TLSF，它是可以裁减的模块，具有优秀的空间利用率与较好的时间响应速度，提供方便的内存管理函数，改善了 UC/OS-II 原有内存管理方案程序接口不够友好、不够规范的缺点，减少了内存浪费，对于一般的实时系统特别是软实时的、应用程序规模大的场合具有较大的应用价值。

参考文献：

【1】 黄贤英，王越，陈媛.嵌入式实时系统内存管理策略[J].计算机工程与设计. 2004 年 10 月,第 25 卷第 10 期.

【2】 M. Masmano, I. Ripoll, A. Crespo, and J. Real.TLSF: a New Dynamic Memory Allocator for Real-Time Systems[R]. Proceedings of the 12th 16th Euromicro Conference on Real-Time Systems (ECRTS'04).1068-3070/04

【3】 M. Masmano. TLSF Implementation and Evaluationin OCERA[R]. Technical Report OCERA D4.4, RealTime Research Group. Universidad Politecnica de Valencia,2004. Available at: http://www.ocera.org andhttp://rtportal.upv.es.

【4】 吴晓勇,曾家智.操作系统内核中动态内存分配机制的研究[J]. 成 都 信 息 工 程 学 院 学 报. 2005 年 2 月,第 20 卷第 1 期.

【5】 杨雷, 吴珏, 陈汶滨.实时系统中动静结合的内存管理实现[J]. 微计算机信息.(嵌入式与 SOC)2005 年第 21 卷第 10-2 期。

【6】 [EB/OL]ftp://ftp.cs.colorado.edu, directory/pub/cs/misc/malloc-benchmarks

【7】 I. Puaut. Real-Time Performance of Dynamic MemoryAllocation Algorithms[R]. Technical Report 1429,Institut de Recherche en Informatique et SystemesAleatories, 2002.

【8】 K. Nilsen and H. Gao. The real-time behavior of dynamicmemory management in C++[R]. In Proceedings of the 1995 Real-Time technology and Applications Symposium, pages142－153, Chicago, Illinois, June 1995.

作者简介：梁乘铭（1981-04），男，汉，广西省岑溪市人，研究生，硕士，主要研究方向为嵌入式系统软件应用。韩坚华（1955-10），女，汉，湖北省人，副教授，硕士生导师，工学硕士，主要研究方向为软件工程、智能控制。夏成文，男，硕士生。覃毅，男，硕士生。

Biography：LIANG CHENG-MING(1981-04), male, han, CenXi city GuangXi province, master, major in embedded system software application. HAN JIAN-HUA(1955-), female, han, HuBei province, vice professor, Master of Engineering, major in software engineer and intelligent control. XIA CHENG-WEN, male, master. QIN YI, male, master.

资金资助项目：广东省科技计划项目（2003C101013）

联系方法：
广州大学城广东工业大学大学城校区计算机学院 G-166 信箱 04 级研究生 梁乘铭 (510006)
lcm242@163.com