【信息科学与控制工程】

基于S3C2440的操作系统实现

蔡亮明

（福州大学 至诚学院，福州 350002）

摘要：利用嵌入式CPU三星S3C2440的硬件体系结构，实现操作系统的一般性功能。实现进程管理，内存管理，中断管理，系统调用等功能。

关键词：嵌入式；S3C2440；操作系统

中图分类号：TP302 **文献标识码**：A **文章编号**：1006－0707(2012)11－0097－03

随着系统复杂度的不断增加，操作系统[1]的实现也越来越庞大，而另一方面，在大学里相关的操作系统课程，却停留在理论层面上的，并未涉及具体的实现方式。在嵌入式系统的应用开发中，采用嵌入式实时操作系统（简称RTOS[2]）能够支持多任务，使得程序开发更加容易，同时能够提高系统的稳定性和可靠性。ARM处理器[3]是一款市场占有率相当大的处理器芯片。

1 操作系统内核的设计

1.1 整体设计思路

利用文件系统来管理外存（NAND FLASH）[4]上的可执行文件；通过进程管理模块来管理装载进内存的可执行程序，并通过时钟中断来划分时间片以此实现多进程的分时系统；使用内存管理模块来实现应用程序或内核对内存堆的申请；使用MMU[5]来实现虚拟地址到物理地址的转换，以此来保证进程有独立的运行空间并提高内存的使用效率。操作系统的总体流程图如图1所示。

1.2 模块设计

1.2.1 系统态与用户态设计

1）ARM处理器有7种不同的处理器模式[6]。其中SYS与USR公用寄存器，其他6种有独立的SP寄存器，LR寄存器，SPSR寄存器，在切换模式的同时，会自动切换其独有寄存器。IRQ，FIQ中断时切换到IRQ，FIQ模式；取指或取数据异常时，切换到ABT[7]模式；无效指令切换到Undef模式。USR模式具有最低的权限，无法进行模式切换，其他7种的权限相同。因此选择USR作为用户态。

2）其他模式到SVC模式[8]的转换。在进入其他模式的时候，保存该模式独有寄存器，然后切换到SVC模式执行系统代码，最后在退出时恢复独有寄存器。这样就保证了系统代码运行在系统态，用户代码运行在用户态。

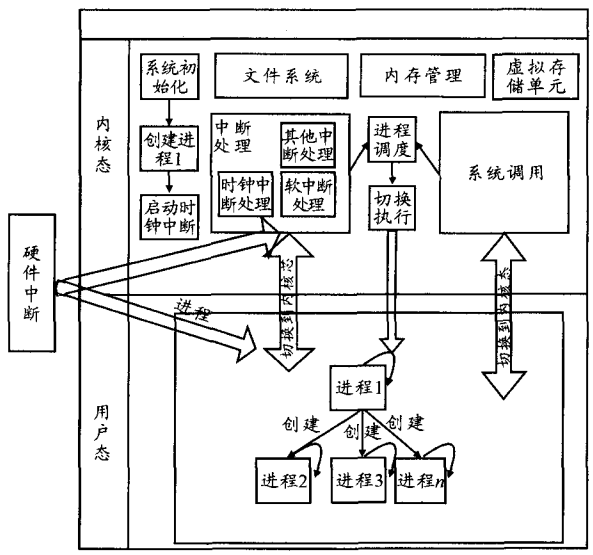

图1 系统流程

1.2.2 进程管理设计

进程切换

在ARM体系结构[9]中，进程的切换是把R0～R15寄存器的值压入进程的栈中，再从被切换的进程的栈中，读出R0～R15的值，这样就完成了进程的切换。具体实现的时候需要预先恢复该进程的用户态SP，LR。

进程调度

动态优先级，模仿LINUX使用NICE来奖励I/O消耗型进程，惩罚处理器消耗型进程。NICE值的范围：－20～20。

进程睡眠与唤醒

同样借鉴LINUX2.6内核[10]的实现。

1）在等待某个事件时，把进程加入到等待队列，设置成睡眠态，调用调度函数，把进程移除可执行队列。

2）在事件到达时，唤醒进程，即把进程设置成运行态，

收稿日期：2012－07－26

作者简介：蔡亮明（1975—），男，高级工程师，主要从事物联网与嵌入式研究。

把进程加入到可执行队列。

当恢复进程运行时,把进程移出睡眠队列。

1.2.3 内存管理设计

1)内存划分如图2所示。

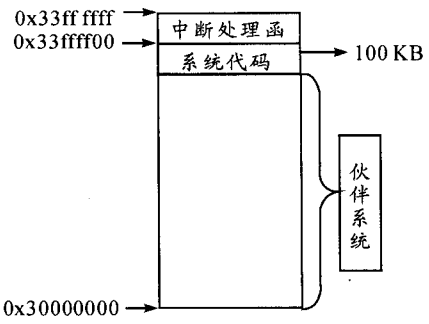

图2 内存划分图

2)虚拟进程空间

每个进程都有一个独立的地址空间,各个进程之间不相互影响。虚拟地址经过MMU时,MMU查询页表项[11](页表存储在内存中),找到相对应的物理地址,从而访问内存,读取数据。现在,只需要每个进程都维护一个自己的页表,就能实现独立的虚拟地址空间。

3)动态内存分配

在物理内存的分配与回收中有多种方法,总的来说分为连续分配与离散分配。在连续分配中,伙伴算法有较好的性能,而在离散分配中,经常使用到的是分页分配。使用分页的前提是需要存在MMU这样的分页单元。

4)伙伴算法的改进

伙伴算法的缺点:

a)合并的要求太过严格:只能是满足伙伴关系的块。比如第1,2块就不能合并。

b)算法中的浪费现象:伙伴算法是按2的幂次方分配内存区的,当需要257(2^8+1)个页面时,不得不申请2^9的页面。于是就有255个页面被浪费掉了。

c)算法的效率问题:但在块的释放过程中,块的回收过程是一个递归过程,伙伴算法把满足条件的两个块合并为一个块,如果合并后的块还可以跟相邻的块进行合并,那么该算法就继续合并。这会花费大量的时间。另外当分配与释放交替进行时,将可能导致对同一内存块的不断分割和合并这样的合并又立即拆分的过程是无效率的。

改进方式:

1)针对a缺点,在内存资源充分的情况下,仍然使用伙伴算法的策略,在内存资源紧张的情况下,合并相邻的非伙伴块来形成大块。

2)针对b缺点,把需要申请的页面数与伙伴算法中对应的页面数相除,若小于阈值,则继续拆分该块,直到大于等于阈值。即把2^9拆分成2^8,2^7—2^1。

3)针对c的改进方法,释放的分区并不马上合并,而是维持一定数目的各种大小的空闲分区;当空闲分区超过一定数目时,才合并。

1.2.4 文件系统设计

为了实现简单,选择了单目录多文件的实现方式。

1)以HASH表的方式实现,文件节点号到文件节点的映射。

2)把文件节点链成一个链表,这样就可以存储该文件系统的所有文件。

3)每个文件节点有一个子节点链表,每个子节点存储该节点在NAND FLASH[12]上的页号。

2 操作系统内核的实现

2.1 系统启动过程

① GPIO寄存器初始化;② 中断寄存器初始化,清除中断,装载中断处理函数;③ 伙伴算法初始化,SLAB算法初始化;④ lcd初始化;⑤ mmu初始化:清除Cache,无效TLB,映射内核页表,把SDRAM的0x30000000 - 0x33f00000,外设寄存器0x40000000 - 0x5af00000,映射到相同的位置,并设置访问这些地址的权限为系统态允许访问,用户态禁止访问,这就实现了用户进程无法直接使用硬件;⑥ 文件系统初始化;⑦ 进程1创建;⑧ 启动时钟中断。

2.2 文件系统的实现

1)文件系统中的使用到得结构体

dev_info:用来描述正NAND FLASH设备见图3。

```
//this struct store in ram,not in nand flsh;
typedef struct _dev_info
{
    unsigned short  totalBlocks;
    block_info      *blocks;          //blocks array[2048];
    struct list_head free_block_list;  //a list to store free blocks;
    struct list_head full_block_list;  //a list to store the block which is use up free pages;
    block_info      *allocBlock;      //current allocation block;
    unsigned long   freeBlocks;       //available free  block num;
    unsigned long   *bitmap;          // 1 bit for 1 page, value 1 for dirty, 0 for free or used;
    unsigned long   totalFiles;       //total files in this dev;
}dev_info;
```

图3 dev_info结构体

2)文件系统组织结构见图4。

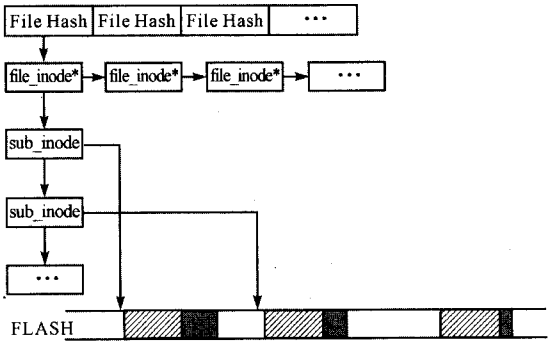

图4 文件系统组织结构

2.3 进程管理的实现

1)进程管理相关的结构体见图5。

```
//process struct
typedef struct _os_pcb
{
    U32 *sp;          //process kernel mod stack;
    U32 pid;          //process id;
    U32 timeSlice;    //process time slice;
    U8  stat;         //process stat;
    U8  prio;         //process priority;
    U32 ttBase;       //mmu page table;
    int sleepTime;    //sleep time from switch out to switch in;
    int runTime;      //run time from switch in to switch out;
    int sleep_avg;    //depend on this to judge this process if is a interactive process;
    U32 timestamp;    //store the moment when the process switch;
    int nice;         //nice range: -20 to 20;
    file *executable; //process code from this excutable file;
    struct list_head list;    //process list;
}os_pcb;
```

图 5 os_pcb 结构体

2）进程的创建

a）创建 PCB,初始化其进程内核堆栈(程序入口地址,CPSR,用户态栈)。

b）拷贝内核的页表项到进程的页表中。

c）把该 PCB 加入到 runqueue 的活动链表中,等待调度。

3）进程的切换

a）切换页表:switch_mm( );

b）切换进程上下文:switch_context( );

4）进程的睡眠与唤醒

用户进程调用 sleep( )系统调用,睡眠 n ms 时间;

```
void sleep(U32 ms)
{
    timer tm;
    DECLARE_WAITQUEUE(wait,list_entry(g_runqueue->cu...,os_pcb,list));

    tm->ticks = cal_ticks(ms);
    tm->wait = &wait;
    add_timer(&tm);
    add_wait_queue(&delay_time_queue, &wait);

    while(!wait.flag)
    {
        set_curr_pcb_stat(PS_SLEEP);
        process_sche(void);
    }
    remove_wait_queue(&wait);
    del_timer(&tm);
}
```

图 6 sleep 函数

2.4 伙伴算法的实现

1）首先,使用 struct page 结构体来表示一页(4KB)。把 n 个页链起来表示一个块,使用该块的首页来表示该块。下图是一个阶数为 2 的块,即 4 个页。如图 7 所示。

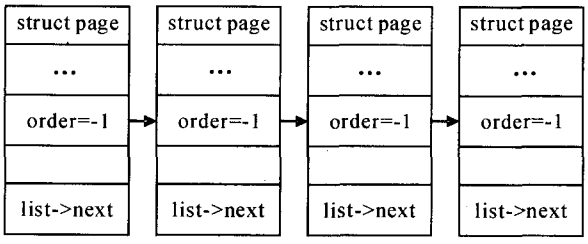

图 7 伙伴算法中的页块

2）由于伙伴算法是折半来降阶的,因此 0 阶即伙伴算法的最小单元,即 1 页。我们使用一个数组来管理每个阶数的块,数组的下标就是对应的阶数。如图 8 所示。

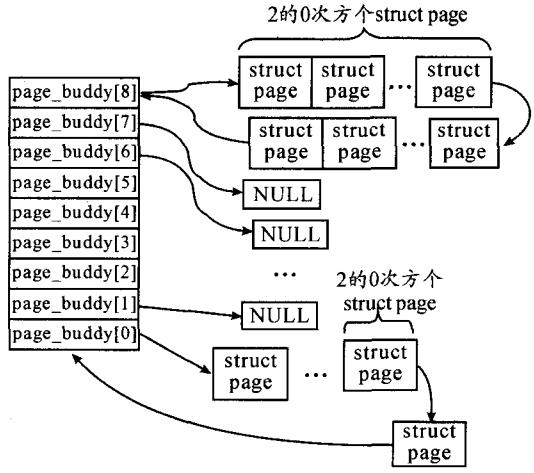

图 8 伙伴算法的管理结构

2.5 MMU 与页表的实现

MMU 的作用过程,如图 9 所示。

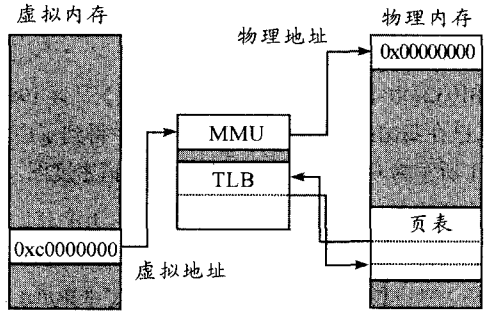

图 9 MMU 的作用过程图

页表基地址 0x33ff8000(存储在 CP15 的 C2 寄存器中),我们访问一个虚拟地址 0xc0033f00,MMU 会从虚拟地址的前 12 位即 0XC00 作为页表基地址的偏移。与页表基地址相加,获得对应的页表项地址。即 0x33ff8c00 (0x33ff8000 | 0xC00)。从该地址取得段描述符,该描述符记录了相对应的 12 位物理段基地址(0x000),由该基地址与虚拟地址的后 20 位偏移相加获得最终的物理地址(0x00033f00)。

3 结束语

本文通过对操作系统内核的设计与实现,论述了什么是操作系统内核,内核由哪些部分组成,要如何工作,如何实现的。操作系统内核的组成成分:进程管理、内存管理、虚拟内存、文件系统、中断机制、驱动程序、输入输出系统、定时器与系统时间。

(下转第 110 页)

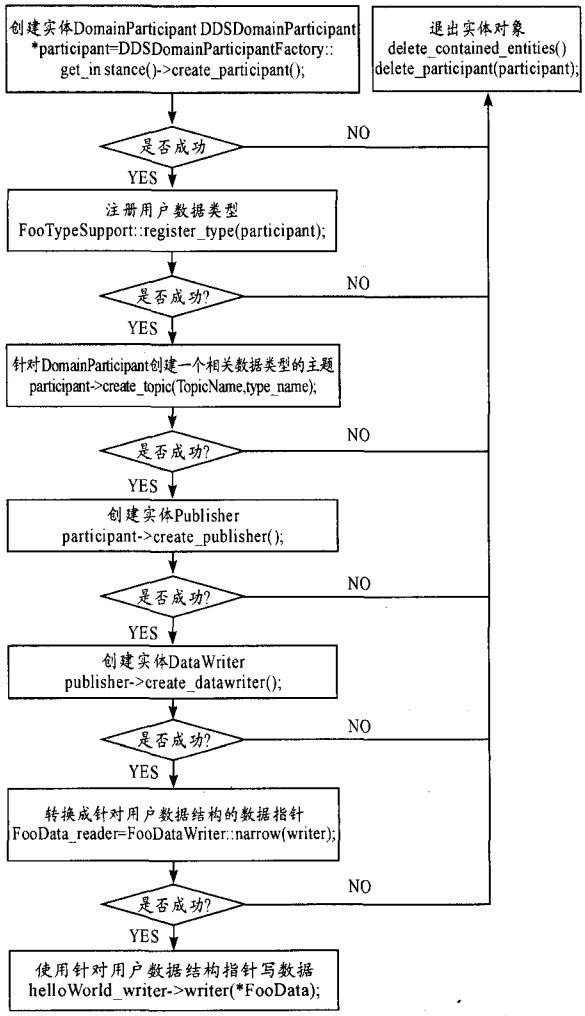

图 5 发布端代码框架

f) 构件组装:使用构件组装工具对多个构件进行定制、连接(与其它构件)和分割,并形成构件组装集存档文件。构件集中包含有多个可定制的构件包以及该构件集的 XML 描述器(包括构件实例和它们的互连信息以及构件集的逻辑组成)。产生的构件集同样支持与其它构件包或构件集一共组装成为更大的构件集。

g) 构件部署与安装:根据构件集的描述符文件把构件集(或构件)安装在网络上指定的节点上、激活构件实例并连接构件。一旦构件被配置和安装成功,构件实例通过标准的 CORBA ORB 机制被激活运行。

图 6 订阅/发布模型

4 结束语

针对分布式系统的特点,介绍了一种分布式系统的集成设计框架,分析了它的工作原理和特点,给出了基于该框架的分布式系统体系结构,并结合 ACE + TAO + CIAO 及 RTI DDS 环境对系统功能构件开发、部署等做了说明。

参考文献:

[1] 韩乐平,薛军敖,孟洛明. OMG. CORBA 系统结构原理与规范[M]. 北京:电子工业出版社,2000.

[2] 张军本,宁伟,王强. 基于构件的分布式软件体系结构设计[J]. 哈尔滨理工大学学报,2001(6):11 - 14.

[3] 曹建福,周理琴. 基于构件的软件开发模型及其实现[J]. 小型微型计算机系统,2002(23):739 - 742.

(责任编辑 杨继森)

(上接第 99 页)

参考文献:

[1] Samsung. S3C2440A 32 - BIT RISC MICROPROCESSOR USER'S MANUAL PRELIMINARY Revision 0. 14 [EB/OL]. [2004 - 07 - 12]. http://www. kamami. pl/dl/s3c2440. pdf.

[2] 任哲. uCOSⅡ实时操作系统[M]. 北京:北京航空航天大学出版社,2009:215 - 260.

[3] 博韦. 深入理解 LINUX 内核[M]. 北京:中国电力出版社,2008:350 - 433.

[4] Robert Love. linux 内核设计与实现[M]. 北京:机械工业出版社,2010:211 - 250.

[5] 于渊. 自己动手写操作系统[M]. 北京:电子工业出版社,2005:112 - 170.

[6] 陈守孔. 算法与数据结构[M]. 北京:机械工业出版社,2008:57 - 89.

[7] 汤小丹. 计算机操作系统[M]. 西安:西安电子科技大学出版社,2007:320 - 350.

[8] Thomas H. Cormen. 算法导论[M]. 北京:机械工业出版社,2008:133 - 150.

[9] 赵炯. LINUX 内核完全剖析[M]. 北京:机械工业出版社,2009:700 - 780.

[10] Robert Love. linux 内核设计与实现[M]. 2 版. 北京:机械工业出版社, 2010.

[11] 任哲. uCOSII 实时操作系统[M]. 2 版. 北京:北京航空航天大学出版社,2009.

(责任编辑 杨继森)

基于S3C2440的操作系统实现

作者： 蔡亮明
作者单位： 福州大学至诚学院,福州,350002
刊名： 四川兵工学报
英文刊名： Journal of Sichuan Ordnance
年，卷(期)： 2012,33(11)

参考文献(11条)

1. Samsung $3C2440A 32-BIT RISC MICROPROCESSOR USER' S MANUAL PRELIMINARY Revision 0,14 2004

2. 任哲 uCOSlI实时操作系统 2009

3. 博韦 深入理解LINUX内核 2008

4. RobertLove linux内核设计与实现 2010

5. 于渊 自己动手写操作系统 2005

6. 陈守孔 算法与数据结构 2008

7. 汤小丹 计算机操作系统 2007

8. ThomasH.Cormen 算法导论 2008

9. 赵炯 LINUX内核完全剖析 2009

10. Robert Love linux内核设计与实现 2010

11. 任哲 uCOSII实时操作系统 2009