第35卷 第11期 计 算 机 工 程 2009年6月
Vol.35 No.11 Computer Engineering June 2009
·开发研究与设计技术· 文章编号：1000—3428(2009)11—0271—03 文献标识码：A 中图分类号：TP316.81

一种基于改造时钟系统的Linux实时化方案

胡 强，蔡自兴
(中南大学信息科学与工程学院，长沙 410083)

摘 要： Linux是一种通用操作系统，但不适合实时应用。针对上述问题，通过对Linux时钟系统管理方面的研究，提出一种针对时钟的改进策略，以此为基础，根据实时应用的特点设计高精度定时器。实验证明，改进方案能有效提高Linux的时钟精度，满足实时方面的需求。

关键词： 实时操作系统；时钟源；时钟事件设备

Linux Real-time Schme Based on Enhanced Time System

HU Qiang, CAI Zi-xing
(School of Information Science and Engineering, Central South University, Changsha 410083)

【Abstract】 Linux is designed to be a general-purpose operating system, it is not suitable for real-time applications, but because of its open source, many solutions have been put forward to enhance its real-time ability. Inspired mostly by Gettimeofday(GTOD), the paper gives a method based on timer management improvement for high resolution timers. Experiments on the modified OS show that the improvement gives a better result for applications with real time constraints than original Linux kernel.

【Key words】 real-time OS; clock source; clock event device

对于实时系统而言，“实时”不仅意味着要求逻辑结果正确，更重要的是响应时间，而且要保证最坏情况下的响应时间，即结果必须产生在截止时间之前。Linux是一个通用操作系统，将它应用于嵌入式实时环境有许多不足，要将Linux用于实时环境必须进行实时化改造[1]。本文所研究的Linux2.6.16内核的时钟粒度在i386体系下被设置为1 ms，而实时应用一般都需要微秒甚至纳秒级的响应精度，1 ms的时钟粒度显然不能满足实时应用的需求。本文改造Linux内核的计时体系，并在此基础上实现了一种高精度定时器。实验结果表明，改造后的时钟系统能明显降低响应延迟，满足实时任务需要。

1 基于GTOD的时钟系统改造

1.1 Linux2.6.16的timeofday子系统

Linux2.6.16内核采用jiffies计时，但是时间精度太低，在频率为1 000 Hz时，只能达到1 ms。为了提高精度，可以采用一个高精度时钟源记录自上一次时钟中断以来经过的时间，从而使Linux达到子节拍的分辨度，内核就能够提供比节拍周期更高的精度来测定当前的时间[2]。这种工作模式可以用下面的伪代码描述：

```
timer_interrupt():
hi_res_base=read_timesource()
xtime+=NSECS_PER_TICK+ntp_adjustment
gettimeofday():
now=read_timesource()
return xtime+cycles2ns(now-hi_res_base)
```

插值函数cycles2ns(now-hi_res_base)提供了2个tick间的精确时间，但同时存在如下问题：可能引起时间倒流，这主要是由时钟源的频率变化和时钟中断延迟或丢失引起的。

内核使用的计时策略有相当多的bug，必须加以改进，才能满足实时需要。内核时钟系统的实现代码分散在time.c, timer.c, time.h, timer.h, times.h, timex.h中，这些文件并没有确切的组织目的，新代码只是随意地加入这些文件，结构相当混乱，因此，代码的组织体系也需要重新整理。

1.2 GTOD的timeofday实现方式

实现方式可以用下面的伪代码表示：

```
nsec_t system_time
nsec_t wall_time_offset
cycle_t offset_base
int ntp_adj
strvuct timesource_t ts
monotonic_clock():
now=read_timesource(ts)
return system_time+cycles2ns(ts,now-offset_base,ntp_adj)
gettimeofday():
retrun monotonic_clock()+wall_time_offset
periodic_hook():
now=read_timesource(ts)
interval=cycles2ns(ts,now-offset_base,ntp_adj)
system_time+=interval
offset_base=now
ntp_adj=ntp_advance(interval)
```

计时系统的主要目的是提供一个单调增加的系统时间和墙上时间，在GTOD中，这2种时间分别通过monotonic_clock()和gettimeofday()获取。从伪代码描述中可以看出：gettimeofday()并没有使用插值函数，在periodic_hook()中的offset_base, system_time变量值和monotonic_clock()中是完全相同的，这就意味着timeofday不再依赖时钟节拍计时，所以，

基金项目： 国家“863”计划基金资助项目(A1420060159)
作者简介： 胡 强(1984－)，男，硕士，主研方向：智能控制，嵌入式系统；蔡自兴，教授、博士生导师
收稿日期： 2008-11-10 **E-mail：** cophu@163.com

中断丢失、中断延迟都不会影响计时的准确度，特别是这段代码可以被所有的体系结构共享，代码冗余度大幅降低，代码维护简单方便。

1.3 时钟源(clock source)管理

GTOD 需要维护一个时钟源抽象层，计时系统通过该抽象层计算过去某个时间点距离现在的时间。为了计算流逝的时间，把系统中的计时硬件抽象出来，然后在通用的计时代码中通过一个指针选择相应的时钟源作为计时硬件。时钟源可以用如下一个简单的结构体表示：

```
struct clock_source{
char *name;
int rating;
cycle_t (*read)(void *data);
int mask;
int mult,shift;
}
```

clock_source 的名字在系统中应该唯一，rating 作为系统运行时选择时钟源的优先级，使用 read 函数返回硬件时钟源的 cycle，或者是一个软件计数器(如 jiffies)，mask 主要用于数学辅助计算，防止减法运算时发生溢出错误。mult 和 shift 也是数学辅助运算函数，它们之间的关系是：mult/2^shift ≈ nanoseconds per cycle，一般 shift 是确定的，只要知道时钟源的频率，就可以计算出 mult。一般而言，对于简单的时钟源，只使用上述域就可以了，如可以定义一个简单的时钟源 simple_clock：

```
struct clock_source simple_clock{
.name=”simple_clock”,
.rating=200,
.mask=0xFFFFFFFF,
.mult=0,
.shift=10};
```

然后编写一个读时钟源函数：

```
cycle_t simple_clock_counter_read(void)
{
  cycle_t ret = readl(simple_clock_ptr);
  return ret;
}
```

使用 register_clocksource(&simple_clock)把 simple_clock 注册到系统中，然后就可以在计时系统中使用 simple_clock。

通过时钟源抽象层，在运行阶段(如插入一个 module)也可以改变时钟源，不需要在内核编译阶段就确定硬件时钟源。

1.4 时钟事件设备的抽象

时钟源的抽象可以实现对时钟源计数器的访问，但是计时系统还需要实现对时钟事件设备的抽象。时钟事件源设备主要用于调度产生下一个时钟中断，Linux2.6.16 中的时钟中断是固定的、周期性产生的时钟“嘀嗒”，并且周期值在内核编译阶段就已确定[3]。内核中与具体时钟服务相关的时钟事件设备的设置和选择在各个体系结构中都被写死了，这不仅造成大量的代码冗余，而且使新添时钟事件设备相当复杂。因此，有必要对时钟事件设备进行抽象，同时钟源类似，可以设计一个简单的结构体来表示时钟事件设备：

```
struct clock_event {
const char* name;
unsigned int capabilities; /*事件源的功能属性*/
unsigned long max_delta_ns; /*事件源的最大中断间隔*/
unsigned long min_delta_ns; /*事件源的最小中断间隔*/
u32 mult;
u32 shift;
void (*set_next_event)(unsigned long evt);
void (*set_mode)(int mode); /*中断处理函数的运行方式*/
int (*suspend)(void);
int (*resume)(void);
void (*event_handler)(struct pt_regs *regs);
void (*start_event)(void *priv);
void (*end_event)(void *priv);
unsigned int irq;
void *priv;
};
```

其中关键的成员是 set_next_event 和 event_handler，set_next_event 可以对时钟事件设备重新编程，调度产生下一个时钟中断，event_handler 是相应时钟中断的回调函数，所以，现在系统可以有不同的时钟中断处理函数，而在没有引入 clock_event 之前，时钟中断处理函数是固定的，所有与周期性节拍相关的服务都在这个函数中实现。

1.5 高精度定时器的实现

Linux2.6.16 中的时间概念都使用 jiffies 表示，jiffies 表示的时间与具体的体系结构相关，所以，用 jiffies 表示的时间并不是绝对的。如果内核中以纳秒表示时间，那么无论在哪种体系结构下，都是绝对的。在 GTOD 的基础上，采用纳秒作为计时单位，可以设计出一种新型的高精度定时器 hrtimer，用如下的结构体来定义：

```
struct hrtimer{
struct rb_node node; /*红黑树节点*/
ktime_t expires; /*定时器到期时间*/
enum hrtimer_state state;
int(*function)(void *); /*定时器回调函数*/
void   *data;
struct list_head   list; /*到期高精度定时器链表*/
};
```

hrtimers 采用红黑树组织定时器，主要用于向应用层提供 nanosleep, posix-timers 和 itimer 接口，当然驱动程序和其他子系统也会用到高精度的定时器。内核里原先每秒周期性地产生 *n* 个 tick(中断)，被在下一个过期的 hrtimer 的定时点上产生中断代替，也就是说中断不再是周期性的，而是靠 clcok_event 的 set_next_event 接口来设置下一个事件中断，只要没有 hrtimer 加载，就没有中断，但是为了保证系统时间(进程时间统计、jiffies 的维护)更新，需要加载一个 hrtimer 来保证周期性中断的产生。

hrtimer 在到期时调用 hrtimr_interrupt()将过期的 hrtimer 从红黑树中删除，放到一个挂起链表中，然后根据剩余的最早过期的 timer 设置事件源下一个中断触发时间，再激活 HRTIMER_SOFTIRQ，这个软中断会执行挂起链表中定时器的回调函数。

2 改造后的时钟系统性能测试

一个实时操作系统实时性能的主要测评标准和指标包括系统响应时间、上下文切换时间、中断延迟时间、调度延迟时间等[4]，而定时器的定时精度可以综合反映这些实时性能指标。为了验证改造后系统的定时精度，笔者编写了一个测试程序 SleepTest，主要是测试 nanosleep()的睡眠精度。测试操作系统是 fedora 5(内核版本号是 2.6.15-1.2054_FC5)，硬件平台是 Genuine Intel CPU 2140 @1.60 GHz。为了完成测试，

需在 fedora 上编译 2.6.16 内核和改造后的 Linux 内核，本文的测试程序分别在 2.6.16 内核和改造后的内核上运行。测试程序的基本思想是：假定进程需要睡眠一定时间以等待某种事件发生(如 I/O 操作的结束)，由于时钟系统的精度问题，因此实际睡眠时间和期望睡眠时间会有一定的差值。时钟系统精度越高，这个差值就越小。SleepTest 进行 1 000 次睡眠循环，每次睡眠 1 s，最后统计出最大、最小以及平均的睡眠延迟时间。SleepTest 的伪代码如下：

```
do{
clock_gettime(now)      /*获取现在的时间，以μs 为单位*/
nanosleep(1000*USEC_PER_SEC)      /*睡眠 1 s */
clock_gettime(next)      /*获取睡眠 1 s 后的时间*/
 diff = calcdiff(now+1000, next)      /*计算实际睡眠时间与期望睡眠时间 1 s 的差值，以μs 为单位*/
 if (diff < stat->min)
 stat->min = diff      /*1 000 次循环中睡眠延迟最小的时间*/
 if (diff > stat->max)
 stat->max = diff      /*1 000 次循环中睡眠延迟最大的时间*/
 stat->avg += (double) diff      / *1 000 次循环中睡眠的平均延迟时间*/
}while(loop<1000)      /*1 000 次循环*/
```

测试结果如表 1、表 2 所示。

表 1　没有负载的结果　μs

内核	最大延迟时间	最小延迟时间	平均延迟时间
2.6.16	1 145	2 152	1 652
改造后的内核	8	8	14

表 2　以 2 个 find 命令作为负载的结果　μs

内核	最大延迟时间	最小延迟时间	平均延迟时间
2.6.16	2 321	10 210	6 569
改造后的内核	5	50	10

从测试结果可以看出：改造后内核的延迟比改造前有明显的降低：延迟从毫秒级降到微秒级，特别是在高负载的情况下，优势更加明显：改造前内核的延迟有大幅上升，最少增加了 102.7%，最多增加了 374.4%，平均延迟增加了 297.6%，而改造后的内核延迟则相对稳定，保持在低延迟状态。

3　结束语

通过 GTOD 接口以及 clock_source 和 clock_event 2 个抽象层，可以明显降低代码的冗余度，使系统在运行阶段能够动态改变时钟源和时钟事件设备，并且为高精度定时器的实现提供了基础。hrtimer 以纳秒为定时单位，可以满足实时应用的定时需求。但是，高精度定时器对系统也有一些负面影响，比如时钟中断频繁，增加系统负担，降低了系统的吞吐量，所以，下一步的研究目标是在满足实时需求和提高吞吐量之间寻找一个合适的平衡点。

参考文献

[1] 赵明富，李太福，陈鸿雁，等. Linux 嵌入式系统的实时性分析[J]. 计算机工程, 2003, 29(18): 89-91.

[2] Gleixner T, Niehaus D. Hrtimers and Beyond: Transforming the Linux Time Subsystems[C]//Proceedings of the 22nd Linux Symposium. Ottawa, Canada: [s. n.], 2006-07.

[3] Love R. Linux 内核设计与实现[M]. 2 版. 陈莉君，康　华，张　波，译. 北京: 机械工业出版社, 2006

[4] 林　浒，蔡光起，李凤宪. 实时化的 Linux 系统及其实时性能的研究[J]. 小型微型计算机系统, 2004, 25(8): 454-1457.

编辑　张正兴

（上接第 270 页）

过药物配伍产生的。现存古方中只简单记载了方剂的组成与主治，缺乏对方剂功效的描述，这对方剂的临床应用和研究极为不利。

方剂功效不是凭空产生的，它有着一定的物质基础，而这个物质基础就是方剂组成药物，即组方药物决定了方剂的功效。一首方往往由几味到几十味药物组成，每味药物都有自身特定的功效集合。组成方剂后，组方药物功效之间有的是相互协同，有的是相互制约，有的既有协同又有制约。因此，方剂功效不是组方药物功效的总和，而是组方药物之间综合作用的结果。此外，方剂功效还受组方药物的剂量、方剂剂型等因素的影响。

5.2.2　功效量化方案[6]

中药功效由文字描述，各功效之间有千丝万缕的联系，为了正确表示各功效间的关系，本系统对功效采用了如下量化方案：

(1)将每个标准功效看作为基于 1 个多维空间的单位矢量，即它的长度为 1。

(2)将每个方剂视为这些单位矢量的加权和矢量，权重可以通过频数加药物的相对剂量组合得到。

(3)将已获取的功效间的相似信息，根据相近程度，定义为单位矢量间的距离。

(4)将单位矢量间的距离转换为单位矢量间夹角的余弦。

(5)最后的功效集合就是和矢量在每个功效上的投影。

通过这样的量化，可以很好地利用空间矢量的性质和计算公式来表示功效之间和方剂之间的相关程度。

6　结束语

中药特性信息数据挖掘系统从传统中药验方入手，挖掘出蕴含在其中的一些规律性知识，根据这些知识，推断出新药方的组合性、味、归经、功效、证、症等药理信息，这些信息可以有效地辅助医生临床用药、新药产品开发和科研人员对中药的研究。

参考文献

[1] 龚燕冰，倪　青，王永炎. 中医证候研究的现代方法学述评[J]. 北京中医药大学学报, 2006, 29(12): 797-801.

[2] 阴小雄，唐常杰，曾令明，等. 基于GEP 的缺剂量古中药复方药剂配伍关系挖掘[J]. 计算机科学, 2004, 31(10): 287-289.

[3] 曾令明，唐常杰，阴小雄，等. 基于位图矩阵和双支持度的中药配伍挖掘技术[J]. 四川大学学报: 自然科学版, 2005, 42(1): 57.

[4] 胡建军. 中药特性信息数据挖掘系统中预处理设计[J]. 计算机工程, 2008, 34(21): 284-285.

[5] 胡　波. 中药方剂性味的多维宏观量化表达方法研究[D]. 成都: 成都中医药大学, 2003.

[6] 彭　京，唐常杰，曾　涛，等. 基于神经网络和属性距离矩阵的中药方剂功效归约算法[J]. 四川大学学报: 工程科学版, 2006, 38(1): 92-97.

编辑　顾姣健