

引用格式:王朴,周晴. 基于龙芯 1E 的嵌入式 Linux 实时性的优化与可靠性设计[J]. 微电子学与计算机, 2019, 36(11): 11-15.
[WANG P, ZHOU Q. Real-time optimization and reliability realization of embedded linux based on loonson 1E [J]. Microelec-
tronics & Computer, 2019, 36(11): 11-15]

基于龙芯 1E 的嵌入式 Linux 实时性的优化与可靠性设计

王 朴^{1,2}, 周 晴²

(1 中国科学院大学 计算机科学与技术学院, 北京 100190; 2 中国科学院 国家空间科学中心, 北京 100190)

摘 要: 为了满足航天领域对嵌入式操作系统实时性需求, 采用实时抢占补丁的方法来优化 Linux 系统的实时性, 并对移植了 RT-preempt Linux 系统的龙芯 1E 平台进行实时性测试, 分别测试了系统进程切换上下文时间、中断响应时间、中断恢复时间以及系统延时. 结果表明实时抢占补丁能满足航天任务对操作系统实时性的需求. 为了实现航天领域对嵌入式操作系统的可靠性, 通过为每个异常设计异常处理策略对 MIPS CPU 的异常处理机制进行优化, 最后通过对典型的异常处理程序进行测试, 验证了该方法的有效性.

关键词: 龙芯; Linux; 实时抢占补丁; 系统可靠性

中图分类号: TP316. 2

文献标识码: A

文章编号: 1000—7180(2019)11—0011—05

Real-time optimization and reliability Realization of embedded
Linux based on Loonson 1E

WANG Pu^{1,2}, ZHOU Qing²

(1 School of Computer Science and Technology, University of Chinese Academy of Sciences,
Beijing 100190, China; 2 National Space Science Center, CAS, Beijing 100190, China)

Abstract: In order to meet the real-time requirements of the embedded operating system in the aerospace field, the RT-preempt patch method is adopted to optimize the real-time performance of the Linux system, and test the real-time performance of the Loongson 1E platform transplanted with the RT-preempt Linux system. The system process switching context time, interrupt response time, interrupt recovery time, and system delay are tested separately. The results show that the RT-preempt patch can meet the real-time requirements of the aerospace mission. In order to realize the reliability of the embedded operating system in the aerospace field, the exception handling mechanism of the MIPS CPU is optimized by designing an exception handling strategy for each exception. Finally, the effectiveness of the method is verified by testing typical exception handling procedures.

Key words: loongson; Linux; RT-preempt; reliability

1 引言

龙芯 1E 处理器是一款高性能抗辐射 32 位处理器, 基于 MIPS 架构, 采用 180 nm 工艺, 主频最高可达 100 MHz, 功耗小于 3 W, 提供 PCI、UART、I2C、SPI 等一系列通用的对外接口. 该款芯片于

2014 年推出, 主要应用在航天领域^[1]. 目前在市场上商用龙芯计算机运行的是基于 Linux 内核的操作系统标准的 Linux 是分时系统, 不能满足航天领域对实时性的需求, 且其异常处理机制不适用于航天应用. 因此, 需要对 Linux 实时性能进行优化, 并针对异常处理进行可靠性设计^[1].

2 几种实时操作系统简介

对目前国内外流行 Vxworks、RTAI、RTLinux/GPL、RT-Preempt 等实时操作系统进行简单的介绍与对比。

Vxworks 是一款实时操作系统,由美国风河公司(WindRiver)开发^[2]。Vxworks 性能优秀,可裁剪性强,具有丰富的接口资源,在军工、航天市场的份额达到 90%,但 Vxworks 并不开源,且价格昂贵,不易获取^[3]。RTLinux/GPL 由新墨西哥矿业及科技学院的 V. Yodaiken 开发,是一款开源的实时操作系统,但目前 RTLinux/GPL 被风河谷公司收购,其开源版本已停止更新。RT-Preempt Patch 是用于提高 Linux 实时性的实时抢占补丁,由 Ingo Molnar 和 Thomas Gleixner 维护,OSADL 实验室对其进行稳定性测试^[4]。实时抢占补丁采用的是整体式内核实时改造方法,使用了中断线程化、临界区可抢占以及优先级继承等技术,可以令 Linux 系统达到硬实时的要求^[5]。RT-Preempt 作为一款实时抢占补丁,与 Linux 主线内核一同更新发布,支持包括 MIPS 在内的多种体系结构。尽管 RT-Preempt 的响应时间要低于 Vxworks、RTLinux 等实时系统,但 RT-Preempt 在 OSADL 和 Linux 社区的支持下不断的更新,补丁的内容也会逐渐并入到新版本的主线内核中。不同于 RTLinux 和 RTAI 使用双内核方法对 Linux 进行实时化改造,实时补丁直接对 Linux 内核内部改造来实时,给用户提供了熟悉的 Linux/POSIX 编程环境,使用户能够充分利用 Linux 系统的丰富资源编写应用程序,各种 RTOS 比较如表 1 所示。

表 1 各种 RTOS 比较

RTOS	优点	缺点
VxWorks	性能优秀;可裁剪性强;接口资源丰富	价格昂贵,不易获取
RTAI	响应速度快;较多的用户和社区支持	不支持 POSIX;不支持 MIPS 架构
RTLinux/GPL	开源;性能优秀	已停止更新
RT-preempt	强大的社区支持;支持大量平台;支持 POSIX,可移植性强	性能略弱 VxWorks 实时操作系统

基于以上的对比分析,本文选择使用安装 RT-preempt 的 Linux 内核来进行移植工作。

3 RT-preempt 补丁原理

实时抢占补丁采用中断线程化、临界区可抢占、

高精度时钟等方法来提高标准 Linux 内核的实时性。

3.1 中断线程化

中断在标准的 Linux 系统中具有最高优先级,当中断发生时系统会暂停当前的任务进入中断处理程序,而且在此期间有可能进行关中断的操作,系统无法进行任务的调度,这样就无法确定任务的最大延迟时间,系统实时性。而中断线程化技术则是将中断服务程序转变成可被操作系统调度的线程,并且给中断服务程序分配优先级,减少关中断的操作,以此来避免实时任务得不到调度的情况^[6]。

需要注意的是,并非所有的中断都能够被线程化。例如时钟中断是整个操作系统的脉搏,不被及时处理会造成整个系统的崩溃。另外一些耗时很少的中断如果被线程化,反而会增加系统开销。Linux 实现中断线程化的流程如图 1 所示。

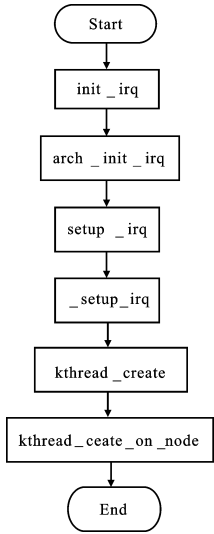


图 1 中断线程化过程

3.2 临界区可抢占

标准 Linux 不支持抢占,内核使用自旋锁和大内核锁来确保数据的一致性。当前进程进入临界区操作数据时,其它进程只能等待,任务完成的时间确定。实时补丁使用优先级可继承的 rt_mutex 替换自旋锁和大内核锁,高优先级任务可以抢占临界区,如果此时低优先级的任务正持有 rt_mutex 锁,高优先级的任务会加入到 rt_mutex 锁的 wait_list 优先级队列中,等待锁的释放。内核拥有全抢占的能力,而且解决了优先级反转的问题。但内核中依然存在不能被抢占的区域如非线程化的中断等,则不能使用 rt_mutex 锁代替^[7]。

3.3 高精度时钟

实时系统需要具有高精度的时钟系统,而标准 Linux 使用 jiffies 记录从系统启动到目前总共的时钟中断次数,系统的时钟频率单位为 HZ, Linux 内核自 2.5 版本开始将频率从 100 Hz 调到 1 000 Hz,时钟分辨率即为 1 ms,这对于实时任务来说是不够的,而且系统时钟频率不能盲目的提高,因为这样会增加时钟中断的次数,降低系统的性能.实时补丁为内核设计了一个新的时钟管理系统^[8].

4 实时抢占补丁性能测试

对移植了 RT-Preempt Linux 系统的龙芯 1E 平台进行性能测试,测试分为编写测试程序进行上下文切换时间,使用示波器测试中断响应时间和中断恢复时间,以及使用专门的实时工具 Cyclicttest 程序进行系统延时的测试.

4.1 测试环境

本次实验的测试环境如表 2 所示.

表 2 测试环境	
处理器	龙芯 1E
内存	256 MB
norflash	4 MB
内核版本	Linux-3. 10. 105
RT-Preempt	patch-3. 10. 105-rt120
宿主机	Ubuntu 14. 04
交叉编译器	gcc-4. 3-ls232

4.2 进程上下文切换时间

进程上下文切换时间是指 cpu 从一个任务切换到另一个任务所耗费的时间,是衡量系统实时性的一个标准^[9]. 本次测试设计了两个进程通过管道通信完成 5 000 次进程切换,进程的优先级设置为 99,采用 SCHED_RR 调度策略,记录结果并绘制图表进行分析.测试上下文切换时间伪代码如下:

```
pipe_init()//初始化管道
setPriority(99);
while(x= fork()){
父进程:
sched_setscheduler(getpid,)//设置调度策略
t1 =gettimeofday();
for(i form 0 to 500){
    read(pipe);
    write(pipe);
}
子进程:
sched_setscheduler(getpid,...)
```

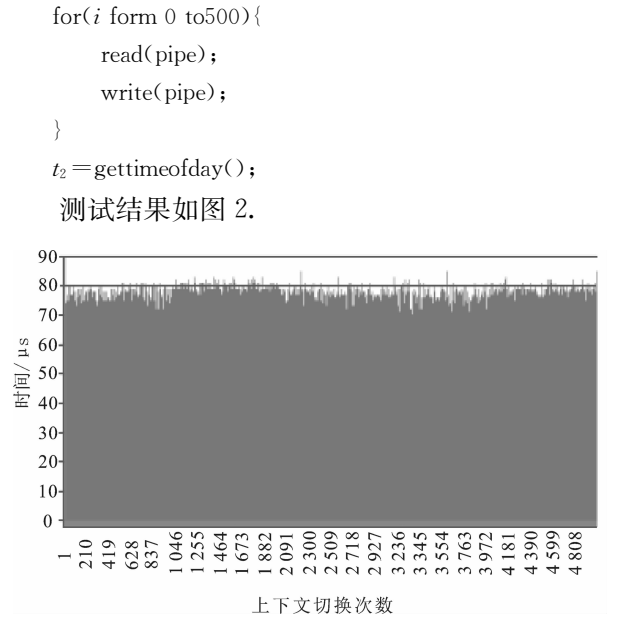


图 2 标准 Linux 测试结果

4.3 中断响应和恢复时间

中断响应时间是指 Linux 系统接外部信号陷入异常到中断服务程序开始执行这一段时间,中断恢复时间是指中断服务程序执行结束系统从异常处理状态恢复的时间,是反映实时系统实时性能的重要标准^[10]. 本次测试是 I2C 中断. 测试程序通过配置龙芯处理器的 CP0 协处理器的 INT_SET 等寄存器来产生 I2C 中断,在触发中断前拉高 GPIO 的第 5 号引脚的电平,进入中断之后中断服务程序将 I2C 中断标志位置 0,然后拉低 GPIO 引脚电平并返回,不断循环这个过程来产生脉冲,用示波器显示波形,观察波形记录中断响应时间和中断恢复时间.测试程序流程如图 3 所示.

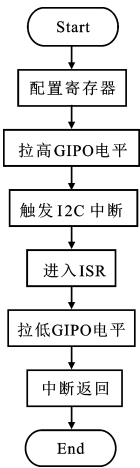


图 3 图中断测试流程

根据示波器的显示,如表 3 所示,中断响应时间

为 11.2 μs , 中断恢复时间为 31.2 μs .

表 3 RT-preempt 与 VxWorks 性能对比

	进程切换	中断响应	中断
VxWorks	33	3.86	18.8
RT-preempt	85	11.2	31.2

其中运行 VxWorks 进行测试的龙芯 1E 频率为 66 MHz, 运行 RT-preempt Linux 的 1E 频率为 33 MHz. 对比可知, RT-preempt 性能虽然弱于 VxWorks, 但能够满足航天任务的需求.

4.4 Cyclictest 测试

Cyclictest 是实时测试组件 rt-tests 的一部分可以准确地测量任务的预期唤醒时间与实际唤醒时间之间的差异, 以提供有关系统延迟的统计信息, 也可以测量由硬件、固件和操作系统引起的实时系统的延迟^[11]. 本次实验使用在内核官网 (<https://mirrors.edge.kernel.org/pub/linux/utils/rt-tests/>) 下载的 1.0 版本的 rt-tests, 通过交叉编译生成 cyclictest 程序, 编译到内核中, 分别在实时和非实时的 Linux 系统下运行. 测试的命令为: cyclictest -l10000 -m -a0 -t1 -n -p99 -i200 -h300 -q > result 其中, -l: loops, 即为循环的个数, 缺省的情况下为 0, 本次实验设为 10000; -m: 锁定当前和将来的内存分配; -a0: 在 0 处理器上运行线程; -t1: threads[num], 线程个数为 1, -n: nanosleep, 使用 clock_nanosleep 来精确执行线程睡眠时间; -p99: 为线程设置的优先级为 99; -i: 线程之间的间隔, 缺省为 1000, 本次实验设置为 200; -h300: 只记录延时在 300 μs 之内结果; -q: quiet, 在程序运行结束时打印结果. 测试结果如图 4、5 及表 4 所示.

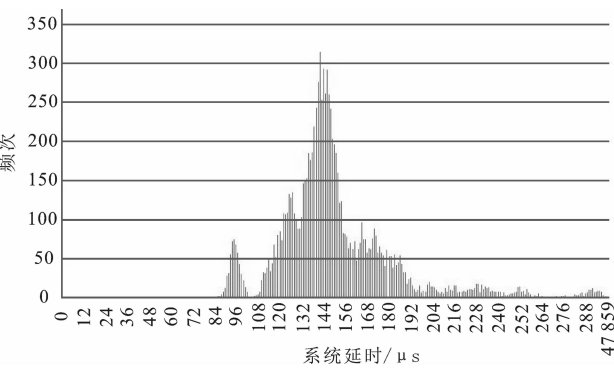


图 4 标准 Linux 测试结果

表 4 实时和非实时内核 cyclictest 测试结果对比

内核模式	Min	Avg	Max
标准 Linux	64	154	47 859
实时 Linux	29	169	288

结果表明, 标准 Linux 的 cyclictest 测试的 Max

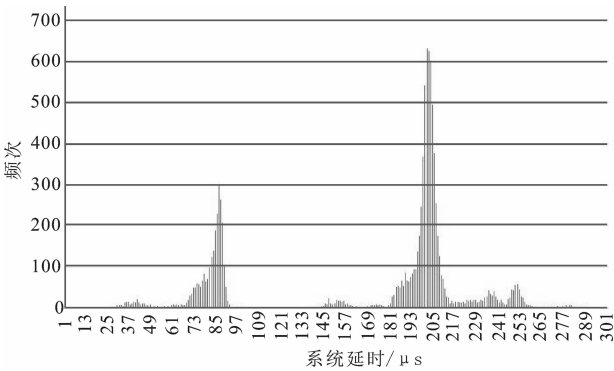


图 5 实时 Linux 测试结果

值要远大于实时 Linux, 可以得出结论, RT-preempt 能够极大的提高 Linux 的实时性.

5 系统可靠性的设计

航天领域嵌入式系统对可靠性具有很高的要求. 航天器所处的太空中存在大量的高能粒子和宇宙射线, 能够对航天器内的电子器件产生影响, 如单粒子翻转效应会使 CPU 寄存器和 RAM 存储器的内容发生改变, 导致系统产生异常. MIPS CPU 的异常一共可以处理 16 种异常, 并非每种异常在处理时都需要休眠当前任务, 不必要的任务休眠严重影响应用程序的执行, 不满足航天应用的可靠性需求. 因此本文对 Linux 系统的异常处理机制进行了改进, 使其能够满足航天任务的对操作系统的需求.

5.1 MIPS 异常处理简介

MIPS 体系结构将中断、自陷、系统调用以及其它打断 CPU 正常执行流的事件统称为异常, 并且采用同一种机制来处理. CPU 的 Cause 寄存器是异常处理的关键寄存器, 其中的 ExcCode 域用来判断异常的类型, 如表 5 所示.

表 5 Cause 寄存器

31	30	29 28	27	26	25 24	23
BD	TI	CE	DC	PCI	0	IV
22	21 16	15 10	9 8	7	6 2	1 0
WP	0	IP7-2	IP1-0	0	ExcCode	0

异常处理流程包括: (1) 设置 EPC 指向重新开始的地址; (2) 设置 Status 寄存器的 EXL 位, 使 CPU 进入内核模式 (高特权级) 并且关闭中断; (3) 设置 Cause 寄存器使软件可以获取异常发生的原因; (4) CPU 从异常入口点取指令, 接下来由软件进行异常处理操作; (5) 保护现场, 对各个寄存器的值进行压栈; (6) 根据 Cause 寄存器的 ExcCode 值判断具体异常并执行相应的异常处理例程; (7) 恢复现

场,写回栈中保存的寄存器的值. 7. 从异常返回, CPU 执行 eret 指令在清除 Status 寄存器 EXL 位的同时跳转到 EPC 中保存的地址^[12].

CPU 的异常列表如表 6 所示.

表 6 异常表

ExcCode	符号	异常名	ExcCode	符号	异常名
0	Int	中断	8	Sys	系统调用
1	Mod	TLB 修改	9	Bp	断点异常
2	TLBL	TLB 异常	10	RI	保留指令
3	TLBS	TLB 异常	11	CpU	协处理器
4	AdEL	地址异常	12	Ov	算数溢出
5	AdES	地址异常	13	Tr	陷阱异常
6	IBE	总线异常	15	FPE	浮点异常
7	DBE	总线异常	23	WATCH	Watch 异常

5.2 异常处理的与实现

查询龙芯 1E 的手册中所有异常的原因以及处理方式根据异常的原因和具体的处理方式将异常分为 3 类;最后编写代码并对内核相关部分进行修改. 下面以第 4、8、号异常为例来进行介绍.

4 号异常即取地址异常发生的原因包括引用非法地址,和取一个非对齐的地址. Linux 对取一个非对齐地址的处理方式是分两次取这个地址引用非法地址会对任务造成致命, Linux 会输出调试信息, 然后调用 do_exit 函数退出进程. 对一种情况,需要记录异常号、EPC、BadVAddr 寄存器的内容,重新运行任务. 8 号异常是系统调用,是用户进入内核模式的正规方式, Linux 会运行相应的处理例程,该类异常不会对任务产生影响. 因此只需要记录 EPC 即可. 12 号异常是算数溢出异常, Linux 会退出进程并输出调试信息,改进策略即为记录异常号、EPC、重启任务. 表 7 列出所有异常的处理策略.

表 7 异常处理策略

异常号	异常处理方案	分类
0	EPC	1
1,2,3	EPC	1
4,5	保存计数、异常号、EPC、BadVAddr	2
6,7	保存计数、异常号、EPC 重启任务	3
8	EPC	1
9、13	保存计数、异常号、EPC、bcode 重启任务	3
10	保存计数、异常号、EPC 重启任务	3
11	保存计数、异常号、EPC、Cause 的 CE 域重启任务	3
12	保存计数、异常号、EPC 重启任务	3
15	保存计数、异常号、EPC、si_code 重启任务	3
23	EPC	2

根据异常处理策略修改内核代码,如断点指令的异常处理代码在 arch/mips/kernel/traps.c 的 do

_be 函数中,修改之后编写应用程序产生除零错误来进行测试.

6 结束语

本文嵌入式 Linux 系统,对系统的进程上下文切换时间、中断响应时间、中断恢复时间、系统延时进行测试,结果表明基于 RT-preempt 的 Linux 系统能够满足航天领域对实时性的要求. 另一方面设计新的异常处理策略.

参考文献:

[1] 邹玉龙,刘彬,田小莉,等. 基于龙芯 1E1F 航天应用平台与 VxWorks 系统的 VxBus 型驱动设计[J]. 计算机测量与控制, 2018, 26 (5):108-112.

[2] 黄江泉. 面向航天应用的高可靠系统软件设计研究[D]. 北京:中国科学院研究生院, 2012.

[3] 肖和龙,唐文胜. 基于 RTAI 改进的 Linux 实时调度算法[J]. 计算机工程与应用, 2012, 48(1):81-83.

[4] WU Z J, MC GUIRE N. Porting RT-preempt to Loongson2F[C]//Eleventh Real-Time Linux Workshop. Dresden, Germany,2009;169.

[5] FAYYAD-KAZAN H, PERNEEL L, TIMMERMAN M. Linux preempt-rt v2. 6.33 versus v3. 6.6: better or worse for real-time applications? [J]. ACM Sigbed Review, 2014, 11(1): 26-31.

[6] HART D, STULTZ J, TS'O T. Real-time Linux in real time[J]. IBM Systems Journal, 2008, 47(2): 207-220.

[7] 吴章金. Linux 实时抢占补丁的研究与实践[D]. 兰州:兰州大学, 2010.

[8] 刘冀. Tickless 技术研究及其在嵌入式系统中的实现[D]. 成都:电子科技大学, 2009.

[9] 刘宇帅,苏宇,王金波,等. 航天嵌入式 Linux 实时性能优化研究[J]. 航天控制, 2018, 36(3):58-63.

[10] 张晓龙,郭锐锋,陶耀东,等. Linux 实时抢占补丁研究及实时性能测试[J]. 计算机工程, 2014, 40(10): 304-307.

[11] Ebugden. Cyclicttest[EB/OL]. [2018-10-15]. <https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/cyclicttest/start>

[12] SWEETMAN D. See MIPS run[M]. Amsterdam, Netherlands ,Elsevier, 2010.

作者简介:

王 朴 男,(1993-),硕士研究生. 研究方向为计算机应用技术. E-mail:wangpiao16@mails.ucas.ac.cn.

周 晴 女,(1972-),硕士,正研级高工. 研究方向为软件系统架构、软件操作系统、软件测试.