

Linux/Android开发记录

学习、记录、分享Linux/Android开发技术

目录视图

摘要视图

RSS 订阅

个人资料



liuhaoyutz

访问: 80609次

积分: 1673分

排名: 第7877名

原创: 83篇

转载: 0篇

译文: 0篇

评论: 59条

博客声明

本博客文章均为原创，欢迎转载交流。转载请注明出处，禁止用于商业目的。

博客专栏



Android应用开发学习笔记本

文章: 30篇

阅读: 17067



LDD3源码分析

文章: 17篇

阅读: 29965

文章分类

LDD3源码分析 (18)

ADC驱动 (1)

触摸屏驱动 (1)

LCD驱动 (1)

Linux设备模型 (8)

USB驱动 (0)

Android架构分析 (12)

Cocos2d-x (1)

C陷阱与缺陷 (3)

Android应用开发 (30)

Linux设备驱动程序架构分析 (8)

有奖征资源，博文分享有内涵

5月推荐博文汇总

大数据读书汇--获奖名单公布

2014 CSDN博文大赛

LDD3源码分析之简单休眠

分类: LDD3源码分析

2012-03-23 17:30

1045人阅读

评论(1)

收藏

举报

module

struct

file

脚本

测试

ubuntu

作者: 刘昊昱

博客: <http://blog.csdn.net/liuhaoyutz>

编译环境: Ubuntu 10.10

内核版本: 2.6.32-38-generic-pae

LDD3源码路径: examples/misc-modules/sleepy.c

本文分析LDD3第六章中关于简单休眠的示例代码sleepy.c。

首先列出sleepy.c的完整代码:

[html]

01. 1/*

02. 2 * sleepy.c -- the writers awake the readers

03. 3 *

04. 4 * Copyright (C) 2001 Alessandro Rubini and Jonathan Corbet

05. 5 * Copyright (C) 2001 O'Reilly & Associates

06. 6 *

07. 7 * The source code in this file can be freely used, adapted,

08. 8 * and redistributed in source or binary form, so long as an

09. 9 * acknowledgment appears in derived source files. The citation

10. 10 * should list that the code comes from the book "Linux Device

11. 11 * Drivers" by Alessandro Rubini and Jonathan Corbet, published

12. 12 * by O'Reilly & Associates. No warranty is attached;

13. 13 * we cannot take responsibility for errors or fitness for use.

14. 14 *

15. 15 * \$Id: sleepy.c,v 1.7 2004/09/26 07:02:43 gregkh Exp \$

16. 16 */

17. 17

18. 18#include <linux/module.h>

19. 19#include <linux/init.h>

20. 20

21. 21#include <linux/sched.h> /* current and everything */

22. 22#include <linux/kernel.h> /* printk() */

23. 23#include <linux/fs.h> /* everything... */

24. 24#include <linux/types.h> /* size_t */

25. 25#include <linux/wait.h>

26. 26

27. 27MODULE_LICENSE("Dual BSD/GPL");

28. 28

29. 29static int sleepy_major = 0;

30. 30

31. 31static DECLARE_WAIT_QUEUE_HEAD(wq);

32. 32static int flag = 0;

33. 33

34. 34ssize_t sleepy_read (struct file *filp, char __user *buf, size_t count, loff_t *pos)

35. 35{

36. 36 printk(KERN_DEBUG "process %i (%s) going to sleep\n",

<http://blog.csdn.net/liuhaoyutz/article/details/7388163>

1/5

最新评论

- [LDD3源码分析之内存映射](#)
wzw88486969:
@jhlhlonng:unsigned long offset
= vma->vm_pgoff <v...
- [Linux设备驱动程序架构分析之I2](#)
teamos:看了你的i2c的几篇文章，真是受益匪浅，虽然让自己写还是ie不出来。非常感谢
- [LDD3源码分析之块设备驱动程序](#)
electfan2011: 感谢楼主的精彩讲解，受益匪浅啊！
- [LDD3源码分析之slab高速缓存](#)
donghuwuwei: 省去了不少修改的时间，真是太好了
- [LDD3源码分析之时间与延迟操作](#)
donghuwuwei: jitc代码需要加上一个头文件。
- [LDD3源码分析之slab高速缓存](#)
捧灰: 今天学到这里了，可是为什么我没有修改源码一遍就通过了额。。。内核版本是2.6.18-53.el5-x...
- [LDD3源码分析之字符设备驱动程序](#)
捧灰: 参照楼主的博客在自学~谢谢楼主！
- [LDD3源码分析之调试技术](#)
fantasyhujian: 分析的很清楚，赞一个！
- [LDD3源码分析之字符设备驱动程序](#)
fantasyhujian: 有时间再好好读读，真的分析的不错！
- [LDD3源码分析之hello.c与Makef](#)
fantasyhujian: 写的很详细，对初学者很有帮助！！

阅读排行

- [LDD3源码分析之字符设](#) (3143)
- [LDD3源码分析之hello.c](#) (2701)
- [S3C2410驱动分析之LCI](#) (2527)
- [Linux设备模型分析之kse](#) (2435)
- [LDD3源码分析之内存映](#) (2336)
- [LDD3源码分析之与硬件](#) (2333)
- [Android架构分析之Andrc](#) (2093)
- [LDD3源码分析之时间与](#) (1987)
- [LDD3源码分析之poll分](#) (1972)
- [S3C2410驱动分析之AD](#) (1948)

评论排行

- [LDD3源码分析之字符设](#) (12)
- [S3C2410驱动分析之触](#) (7)
- [LDD3源码分析之内存映](#) (5)
- [LDD3源码分析之hello.c](#) (4)
- [Linux设备模型分析之kot](#) (4)
- [LDD3源码分析之slab高](#) (4)
- [S3C2410驱动分析之LCI](#) (3)
- [LDD3源码分析之阻塞型I](#) (3)
- [LDD3源码分析之时间与](#) (3)
- [LDD3源码分析之poll分](#) (2)

文章存档

- [2014年06月](#) (1)
- [2014年05月](#) (4)
- [2014年04月](#) (1)

```
37. 37         current->pid, current->comm);
38. 38     wait_event_interruptible(wq, flag != 0);
39. 39     flag = 0;
40. 40     printk(KERN_DEBUG "awoken %i (%s)\n", current->pid, current->comm);
41. 41     return 0; /* EOF */
42. 42}
43. 43
44. 44ssize_t sleepy_write (struct file *filp, const char __user *buf, size_t count,
45. 45         loff_t *pos)
46. 46{
47. 47     printk(KERN_DEBUG "process %i (%s) awakening the readers...\n",
48. 48         current->pid, current->comm);
49. 49     flag = 1;
50. 50     wake_up_interruptible(&wq);
51. 51     return count; /* succeed, to avoid retrial */
52. 52}
53. 53
54. 54
55. 55struct file_operations sleepy_fops = {
56. 56     .owner = THIS_MODULE,
57. 57     .read = sleepy_read,
58. 58     .write = sleepy_write,
59. 59};
60. 60
61. 61
62. 62int sleepy_init(void)
63. 63{
64. 64     int result;
65. 65
66. 66     /*
67. 67      * Register your major, and accept a dynamic number
68. 68      */
69. 69     result = register_chrdev(sleepy_major, "sleepy", &sleepy_fops);
70. 70     if (result < 0)
71. 71         return result;
72. 72     if (sleepy_major == 0)
73. 73         sleepy_major = result; /* dynamic */
74. 74     return 0;
75. 75}
76. 76
77. 77void sleepy_cleanup(void)
78. 78{
79. 79     unregister_chrdev(sleepy_major, "sleepy");
80. 80}
81. 81
82. 82module_init(sleepy_init);
83. 83module_exit(sleepy_cleanup);
```

在模块初始化函数中，注册字符设备“sleepy”时，指定了该设备的读写函数分别是sleepy_read和sleepy_write。当某个进程对sleepy执行读操作时，会进入休眠。当某个进程对sleepy执行写操作时，会唤醒相应等待队列中的所有休眠进程。

为了管理休眠进程，需要建立等待队列，等待队列就是一个进程链表，其中包含等待某个特定事件的所有进程。等待队列通过“等待队列头”来管理，等待队列头是一个类型为wait_queue_head_t的结构体。可以静态初始化一个等待队列头：

```
DECLARE_WAIT_QUEUE_HEAD(name);
```

也可以动态初始化一个等待队列头：

```
wait_queue_head_t my_queue;
```

```
init_waitqueue_head(&my_queue);
```

一个进程要进入休眠，最常用的函数是：

```
wait_event_interruptible(queue, condition);
```

queue是等待队列头，condition是一个条件表达式，进程进入休眠前和被唤醒后，都会检查condition的值是否为真，如果不为真，则进程会进入休眠。

对应wait_event_interruptible的唤醒函数是：

```
wake_up_interruptible(wait_queue_head_t *queue)
```

[2014年01月](#) (1)[2013年12月](#) (6)[展开](#)

文章搜索

推荐文章

sleepy.c第31行定义了等待队列头wq:

```
[html]
01. 31static DECLARE_WAIT_QUEUE_HEAD(wq);
```

在sleepy_read函数中, 38行调用wait_event_interruptible(wq, flag != 0)进入休眠。所以只要有进程对sleepy执行读操作, 就会进入休眠。

在sleepy_write函数中, 49行将flag设置为1, 然后调用wake_up_interruptible(&wq)将等待在wq上的进程唤醒。

注意, 因为在sleepy_read函数中, 休眠进程被唤醒后, 会把flag重新设置为0, 所以虽然全部休眠进程都会被唤醒, 但一次只有一个进程能真正继续执行, 其它进程会重新休眠。但是为简单起见, 这里没考虑并发处理等问题。

要测试sleepy模块, 我们先创建sleepy_load和sleepy_unload脚本。

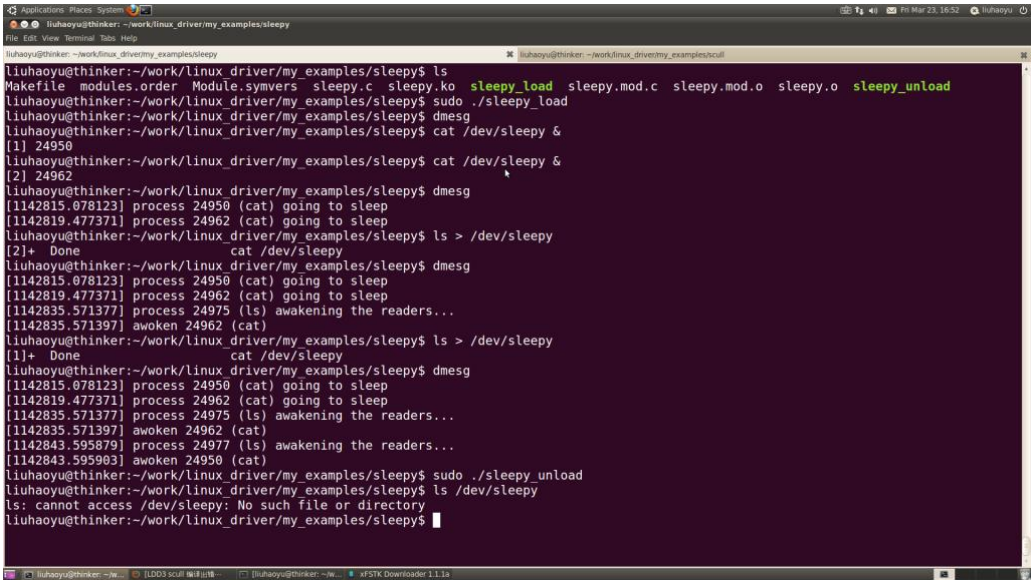
sleepy_load脚本的内容如下:

```
[html]
01. #!/bin/sh
02. # $Id: complete_load,v 1.4 2004/11/03 06:19:49 rubini Exp $
03. module="sleepy"
04. device="sleepy"
05. mode="666"
06.
07. # Group: since distributions do it differently, look for wheel or use staff
08. if grep -q '^staff:' /etc/group; then
09.     group="staff"
10. else
11.     group="wheel"
12. fi
13.
14. # invoke insmod with all arguments we got
15. # and use a pathname, as insmod doesn't look in . by default
16. /sbin/insmod ./module.ko $* || exit 1
17.
18. # retrieve major number
19. major=$(awk "\$2==\"$module\" {print \$1}" /proc/devices)
20.
21. # Remove stale nodes and replace them, then give gid and perms
22. # Usually the script is shorter, it's scull that has several devices in it.
23.
24. rm -f /dev/${device}
25. mknod /dev/${device} c $major 0
26.
27. chgrp $group /dev/${device}
28. chmod $mode /dev/${device}
```

sleepy_unload脚本的内容如下:

```
[html]
01. #!/bin/sh
02. module="sleepy"
03. device="sleepy"
04.
05. # invoke rmmod with all arguments we got
06. /sbin/rmmod $module $* || exit 1
07.
08. # Remove stale nodes
09.
10. rm -f /dev/${device}
```

sleepy模块的测试过程如下图所示:



更多 0

上一篇 LDD3源码分析之ioctl操作
下一篇 LDD3源码分析之阻塞型I/O

主题推荐 源码 source code 脚本 管理 测试

猜你在找

- LDD3源码分析之调试技术
- LDD3源码分析之内存映射
- LDD3源码分析之时间与延迟操作
- LDD3源码分析之slab高速缓存
- LDD3源码分析之内存映射
- LDD3源码分析之时间与延迟操作
- LDD3源码分析之并发与竞态
- LDD3源码分析之阻塞型I/O
- LDD3源码分析之hello.c与Makefile模板
- LDD3源码分析之与硬件通信&中断处理

免费学习IT4个月,月薪12000

中国[官方授权]IT培训与就业示范基地,学成后名企直接招聘,月薪12000起!

查看评论

1楼 liuyao550 2013-10-05 23:05发表



赞!
如果能报 read中进程唤醒后应当做的并发进程处理做一个简要的介绍就好了。

您还没有登录,请[登录](#)或[注册](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

- 全部主题
- Java VPN Android iOS ERP IE10 Eclipse CRM JavaScript Ubuntu NFC
- WAP jQuery 数据库 BI HTML5 Spring Apache Hadoop .NET API HTML SDK IIS
- Fedora XML LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE
- Cassandra CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace
- Web App SpringSide Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate
- ThinkPHP Spark HBase Pure Solr Angular Cloud Foundry Redis Scala Django
- Bootstrap

