Linux/Android开发记录 学习、记录、分享Linux/Android开发技术

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

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]
 1/*
 2 * sleepy.c -- the writers awake the readers
 3 *
 4 * Copyright (C) 2001 Alessandro Rubini and Jonathan Corbet
 5 * Copyright (C) 2001 O'Reilly & Associates
 6 *
 7 * The source code in this file can be freely used, adapted,
 8 * and redistributed in source or binary form, so long as an
 9 * acknowledgment appears in derived source files.  The citation
10 * should list that the code comes from the book "Linux Device
11 * Drivers" by Alessandro Rubini and Jonathan Corbet, published
12 * by O'Reilly & Associates.   No warranty is attached;
13 * we cannot take responsibility for errors or fitness for use.
14 *
15 * $Id: sleepy.c,v 1.7 2004/09/26 07:02:43 gregkh Exp $
16 */
17
18#include <linux/module.h>
19#include <linux/init.h>
20
21#include <linux/sched.h>  /* current and everything */
22#include <linux/kernel.h> /* printk() */
23#include <linux/fs.h>     /* everything... */
24#include <linux/types.h>  /* size_t */
25#include <linux/wait.h>
26
27MODULE_LICENSE("Dual BSD/GPL");
28
29static int sleepy_major = 0;
30
31static DECLARE_WAIT_QUEUE_HEAD(wq);
32static int flag = 0;
33
34ssize_t sleepy_read (struct file *filp, char __user *buf, size_t count, loff_t *pos)
35{
36    printk(KERN_DEBUG "process %i (%s) going to sleep\n",
```

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

sleepy.c第31行定义了等待队列头wq：

```
[html]
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]
#!/bin/sh
# $Id: complete_load,v 1.4 2004/11/03 06:19:49 rubini Exp $
module="sleepy"
device="sleepy"
mode="666"

# Group: since distributions do it differently, look for wheel or use staff
if grep -q '^staff:' /etc/group; then
    group="staff"
else
    group="wheel"
fi

# invoke insmod with all arguments we got
# and use a pathname, as insmod doesn't look in . by default
/sbin/insmod ./$module.ko $* || exit 1

# retrieve major number
major=$(awk "\$2==\"$module\" {print \$1}" /proc/devices)

# Remove stale nodes and replace them, then give gid and perms
# Usually the script is shorter, it's scull that has several devices in it.

rm -f /dev/${device}
mknod /dev/${device} c $major 0

chgrp $group /dev/${device}
chmod $mode  /dev/${device}
```

sleepy_unload脚本的内容如下：

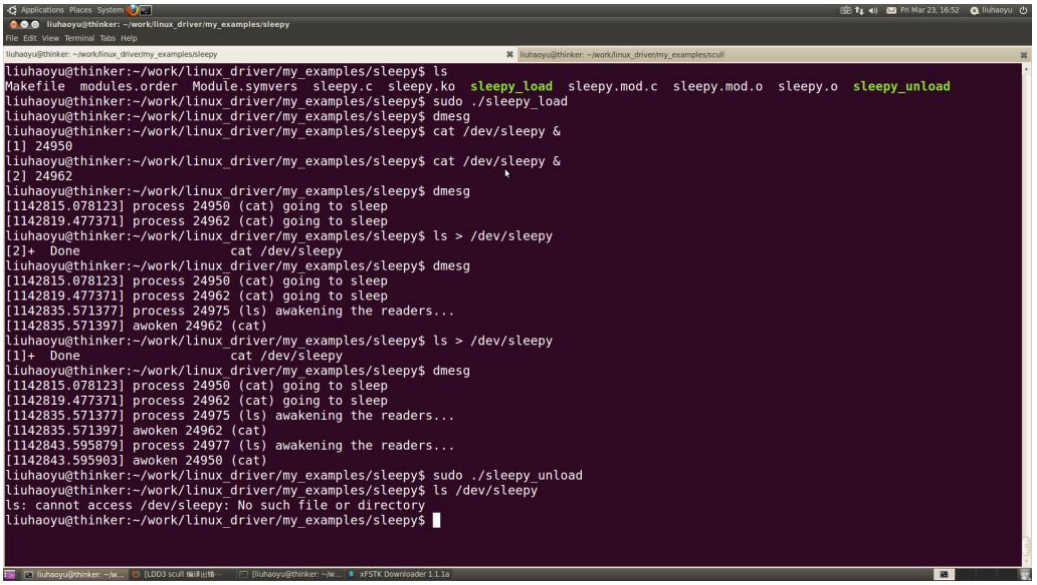
```
[html]
#!/bin/sh
module="sleepy"
device="sleepy"

# invoke rmmod with all arguments we got
/sbin/rmmod $module $* || exit 1

# Remove stale nodes

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

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。

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