

## luofuchong

2007年12月						
日	一	二	三	四	五	六
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	<a href="#">18</a>	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

### 常用链接

[我的随笔](#)  
[我的评论](#)  
[我参与的随笔](#)

### 留言簿(29)

[给我留言](#)  
[查看公开留言](#)  
[查看私人留言](#)

### 随笔档案

[2008年4月 \(4\)](#)  
[2008年3月 \(1\)](#)  
[2008年1月 \(2\)](#)  
[2007年12月 \(1\)](#)  
[2007年11月 \(1\)](#)  
[2007年10月 \(1\)](#)  
[2007年9月 \(5\)](#)  
[2007年8月 \(8\)](#)  
[2007年7月 \(2\)](#)  
[2007年1月 \(5\)](#)

### 搜索

### 最新评论 XML

1. re: [yaffs2源代码情景分析](#)  
 很好的文章,持续关注。。谢谢。  
 --dAsh

2. re: [根文件系统的制作](#)  
 thanks very much!  
 --wwlhZ

3. re: [s3c2410触摸屏驱动\(2.6内核\)分析](#)  
 评论内容较长,点击标题查看  
 --hxlIn

4. re: [s3c2410触摸屏驱动\(2.6内核\)](#)

[IT博客网](#) [首页](#) [新随笔](#) [联系](#) [聚合](#) XML [管理](#)

随笔-30 评论-55 文章-4 trackbacks-0

## udev规则编写(更新中)

Daniel Drake(dsd)编写

版本: 0.73

luofuchong译

该文档最新版本可以在这找到:

[http://www.reactivated.net/writing\\_udev\\_rules.html](http://www.reactivated.net/writing_udev_rules.html)

### 介绍

### 关于本文档

udev旨在用于linux2.6或更高版本的内核上,为用户空间提供使用固定设备名的动态/dev目录的解决方案。以前的/dev目录实现,devfs,现在已经被抛弃,udev取代了其地位。有关udev和devfs的争论是个敏感的话题,在作出对比前你可以参考一下该文档:

[http://kernel.org/pub/linux/utils/kernel/hotplug/udev\\_vs\\_devfs](http://kernel.org/pub/linux/utils/kernel/hotplug/udev_vs_devfs)

Over the years, the things that you might use udev rules for has changed, as well as the flexibility of rules themselves. On a modern system, udev provides persistent naming for some device types out-of-the-box, eliminating the need for custom rules for those devices. However, some users will still require the extra level of customisation.

本文档假设你已经安装好udev并使用默认配置成功运行。通常你所使用的linux发行版已经帮你做好了这一切。

本文档不会包括所有编写udev规则的细节,而旨在介绍所有主要的概念。更详尽的细节可以从udev帮助手册中找到。

本文档使用多个例子去说明概念和想法,其中有些例子是完全虚构的。不是所有的语法都会在随后的文档中详细描述,所以请仔细阅读例子程序以便完全的去理解它。

### History

\* December 3rd 2007 v0.73: Update for new udev versions,

and some miscellaneous improvements.

- \* October 2nd 2006 v0.72: Fixed a typo in one of the example rules.

- \* June 10th 2006 v0.71: Misc changes based on recent feedback - thanks!

- \* June 3rd 2006 v0.7: Complete rework, to be more suited for the modern-day udev.

- \* May 9th 2005 v0.6: Misc updates, including information about udevinfo, groups and permissions, logging, and udevtest.

- \* June 20th 2004 v0.55: Added info on multiple symlinks, and some minor changes/updates.

- \* April 26th 2004 v0.54: Added some Debian info. Minor corrections. Re-reverted information about what to call your rule file. Added info about naming network interfaces.

- \* April 15th 2004 v0.53: Minor corrections. Added info about NAME{all\_partitions}. Added info about other udevinfo tricks.

- \* April 14th 2004 v0.52: Reverted to suggesting using "udev.rules" until the udev defaults allow for other files. Minor work.

- \* April 6th 2004 v0.51: I now write suggest users to use their own "local.rules" file rather than prepending "udev.rules".

- \* April 3rd 2004 v0.5: Minor cleanups and preparations for possible inclusion in the udev distribution.

- \* March 20th 2004 v0.4: General improvements, clarifications, and cleanups. Added more information about writing rules for usb-storage.

- \* February 23rd 2004 v0.3: Rewrote some parts to emphasise how sysfs naming works, and how it can be matched. Updated rule-writing parts to represent udev 018s new SYSFS{filename} naming scheme. Improved sectioning, and clarified many points. Added info about KDE.

- \* February 18th 2004 v0.2: Fixed a small omission in an example. Updated section on identifying mass-storage devices. Updated section on nvidia.

- \* February 15th 2004 v0.1: Initial publication.

## 概念

术语: devfs, sysfs, nodes, etc.

只是基本的介绍, 未必完全的正确。

在典型的基于linux的系统, /dev目录用于存储与系统设备关联的设备节点文件。每个节点指向系统的一部分(一个设备), 无论它存在或不存在。用户空间的应用程序可以通过这些设备节点与系统硬件设备进行交互, 例如: X服务器会监听/dev/input/mice设备, 移动虚拟的鼠标指针以反映用户鼠标的移动。

传统的/dev目录存在所有系统可能会用到的设备文件。因为这样, 典型的/dev目录非常的巨大。devfs应用而生以提供更为易于管理设备文件的方法(值得注意的是, 它只在/dev目录下放置接入系统的相关设备文件)和一些其它功能, 然而, devfs被证实存在不可轻易解决的问题。

udev是管理/dev目录的新技术, designed to clear up some issues with previous /dev implementations, and provide a robust path

forward. 为了生成和命令/dev目录下与存在于系统中的设备相应的设备节点文件, udev依赖于sysfs提供的匹配信息和用户提供的规则的结合。 This documentation aims to detail the process of rule-writing, one of the only udev-related tasks that must (optionally) be performed by the user.

sysfs是2.6内核下新的子系统。它由内核来管理, 用于导出当前接入系统的设备的基本信息。sysfs被挂载到/sys目录下, 用户可见。 You may wish to investigate some of the files stored there before getting to grips with udev. 纵观全文, 我将会交替的使用/sys和sysfs这个词。

### Why?

udev规则相当灵活、强大。通过使用udev的规则,你可以实现:

重命名设备节点默认的名字为其它名字。

通过为默认的设备节点建立链接实现可变/固定的设备节点名。

基于程序的输出来命名设备节点。

改变设备节点的访问权限和所有权。

当设备节点被建立和删除的时候调用一个脚本(典型的当设备接入或被拔除的时候)。

重命名网络接口。

Writing rules is not a workaround for the problem where no device nodes for your particular device exist. 就算没有匹配的规则, udev仍会使用内核提供的默认名字来生成设备节点。

设备节点名字固定拥有不少的好处。假设你拥有两个USB storage设备: 一个数码相机和一个USB接口的flash存储器。这些设备典型的被分配设备节点/dev/sda和/dev/sdb,然而确切的分配依赖于他们接入的顺序。这样可能会为用户带来一些麻烦,如果每个设备每次都采用固定的名字,例如:/dev/camera和/dev/flashdisk的话,用户可能将会大大的收益。

### Built-in persistent naming schemes

udev provides persistent naming for some device types out of the box. This is a very useful feature, and in many circumstances means that your journey ends here: you do not have to write any rules.

udev为storage设备在/dev/disk目录下提供out-fo-the-box的固定命名。想要查看其为你的storage设备生成的固定名字,可以施用如下命令:

```
# ls -lR /dev/disk
```

这对于所有的storage类型均适用。举个例子:

udev为我的主分区生成固定名字的符号链接/dev/disk/by-id/scsi-SATA\_ST3120827AS\_4MS1NDXZ-part3。

当我插入USB flash 存储器的时候,udev为我生成/dev/disk/by-id/usb-Prolific\_Technology\_Inc.\_USB\_Mass\_Storage\_Device-part1,同样固定的名字。

### 规则编写

#### 规则文件和语法

udev通过读取一系列的规则文件来决定如何命名设备和执行附加动作。这些规则文件保存在/etc/udev/rules.d目录下，均使用.rules的后缀。

缺省的udev规则存放在/etc/udev/rules.d/50-udev.rules文件中。仔细阅读，你会觉得非常有意思—它包含一些例子，和一些默认的规则以提供一种devfs风格的/dev层。然而，你不应该往里面添加规则。

Files in /etc/udev/rules.d/ are parsed in lexical order, and in some circumstances, the order in which rules are parsed is important.

通常，希望你自己的规则先于默认的规则被运行，所以我建议你新增/etc/udev/rules.d/10-local.rules，往里面添加你自己的规则。

在规则文件中，以字符#开头的被认为是注释。其它非空的行被视为规则。规则不能跨越多行。

一个设备可以与多于一个的规则匹配。这有其实用的好处，比如：我们可以为相同的设备编写两个规则，每一个规则为该设备提供适合自身的名字。所有别名都将会被创建，即使规则存在独立的文件中。udev不会因为找到匹配的规则而停止运行，他会继续寻找并尝试应用每一条它所能识别的规则，明确这一点是非常重要的。

### 语法规则

每个规则由一系列有逗号隔开的关键字构成。match keys are conditions used to identify the device which the rule is acting upon.

如果规则中所用的关键值均与所操作的设备匹配的话，规则将被应用，指定的处理脚本将被调用。

每条规则至少包含一个匹配选项、一个指定响应值。

以下是一个很好的解释上面所指情况的例子：

```
KERNEL=="hdb", NAME="my_spare_disk"
```

以上规则包含一个匹配关键字(KERNEL)和一个指定值(NAME)。The semantics of these keys and their properties will be detailed later.

It is important to note that the match key is related to its value through the equality operator (==), whereas the assignment key is related to its value through the assignment operator (=).

注意udev不支持续行符。不要在你的规则中插入任何空行，这将导致udev把你的一条规则视为多条规则，从而导致你的规则不能像预期那样执行。

### 基本规则

udev提供一些不同的匹配规则供用户编写和设备严格匹配的规则。以下介绍一些最常见的匹配值，其它的将会在文档的后续部分介绍。如果想要详细了解udev的应用，请参考udev的帮助手册。

.KERNEL - 和设备的内核名字匹配

.SUBSYSTEM - 和设备所属的子系统匹配

## .DRIVER - 与设备的驱动程序匹配

当你采用一系列的匹配规则去准确匹配设备后, udev允许你通过a range of assignment keys, 很好的去控制后续的操作。

想要了解所有可能的分配值, 请查看udev的帮助手册。以下介绍最基本的分配值。其它的将会在文档的后续部分介绍。

.NAME - 为设备节点所使用的名字

.SYMLINK - 作为设备节点别名的符号链接链表

正如上面所提到的, udev仅为每个设备生成唯一的设备节点。如果你想要为该设备节点提供可变的名字, 你可以使用符号链接的功能。

With the SYMLINK assignment, you are actually maintaining a list of symbolic links, all of which will be pointed at the real device node.

To manipulate these links, we introduce a new operator for appending to lists: +=. You can append multiple symlinks to the list from any one rule by separating each one with a space.

```
KERNEL=="hdb", NAME="my_spare_disk"
```

以上规则可以这样理解: 匹配一个内核名字为hdb的设备, 并改用my\_spare\_disk来命名这个设备节点。

该设备节点将会出现在/dev/my\_spare\_disk。

```
KERNEL=="hdb", DRIVER=="ide-disk", SYMLINK+="sparedisk"
```

以上规则可以这样理解: 匹配一个内核名字为hdb并使用ide-disk驱动程序的设备, 使用其默认设备名并建立名为sparedisk的符号链接指向它。

由于我们并没有指定设备节点的名字, 所以udev使用默认值。为了保留标准的/dev层, 你所编写的规则应该保证设备名独立, 通过建立一些符号链接并、或执行其它分配:

```
KERNEL=="hdc", SYMLINK+="cdrom cdrom0"
```

The above rule is probably more typical of the types of rules you might be writing. 它生成两个符号链接/dev/cdrom和/dev/cdrom0, 全部指向/dev/hdc。

再次, 没有指定设备名, 所以沿用默认的内核名hdc。

## sysfs属性匹配

目前为止所介绍的匹配关键字只提供有限的匹配功能. 实际上我们需要更好的控制: 我们想要通国更高级的属性, 例如vendor codes, exact product numbers, 序列号, 存储容量, 分区数目等等来识别设备。

许多驱动程序把类似这样的信息导出到sysfs, udev允许我们在编写规则的时候, 通过使用SYSFS关键字这么一个稍微有点特别的语法, 引入sysfs的匹配条件。

以下是一些使用sysfs数据作为匹配条件的例子. 更详尽的内容将会在本文档的后续部分进行介绍, 它将为你基于sysfs属性来编写规则起到一定的帮助作用。

```
KERNEL=="sda", SYSFS{model}=="ST3120827AS",
```

```
SYMLINK+="my_hard_disk"  
SUBSYSTEM=="block", SYSFS{size}=="234441648",  
SYMLINK+="my_disk"  
BUS=="usb", SYSFS{manufacturer}=="OLYMPUS",  
SYSFS{product}=="X250,D560Z,C350Z", SYMLINK+="camera"
```

### 字符替换

编写可能处理多个类似的设备的规则时,udev类似于printf的字符替换功能变得非常有用.You can simply include these operators in any assignments your rule makes,但它们被执行的时候,udev会充分的对它们进行评估.

最通用的操作莫过于%k和%n.%k等同于设备的内核名,例如:"sda3"代表一个默认出现在/dev/sda3目录下的设备.%n等同于设备的内核号(设备的分区号),例如:"3"代表/dev/sda3.

udev同样为更高级的应用提供一些其它的替换操作.读完这篇文章后可以参考一下udev的帮助文档.同样,它为这些操作提供了一个可选的语法-\$kernel and \$number for the examples above. 正应为如此,如果你在规则中匹配%符号的话,那你不得不使用%%,同样,如果你想要匹配符号\$,那你必需使用\$\$.

为了更好的说明字符替换的概念,以下举一些例子规则进行说明:

```
KERNEL=="mice", NAME="input/%k"  
KERNEL=="loop0", NAME="loop/%n", SYMLINK+="%k"
```

第一个规则确保mice设备接点唯一的建立在/dev/input目录下(默认它应该建立在在/dev/mice).第二条规则保证名为loop0的设备接点被创建于/dev/loop/0的同时,建立符号链接/dev/loop0.

以上规则的编写可能会引起你的质疑,因为他们完全可以重写为不使用任何字符替换来实现.字符替换的真正用法将会在下面的章节中为你介绍.

### 字符匹配

除了确切的匹配字符以外,udev允许你使用shell的模式匹配风格.有以下三种模式可供选择:

\*匹配任意多个字符,0个或多个均可.

?匹配任意单个字符.

[]匹配方括号中指定的任意单个字符,允许设置范围.

以下例子包含上面所提到模式.留意字符替换操作的用法.

```
KERNEL=="fd[0-9]*", NAME="floppy/%n", SYMLINK+="%k"  
KERNEL=="hiddev*", NAME="usb/%k"
```

第一条规则匹配所有的软盘驱动,确保设备节点位于/dev/floppy目录,同时建立一个使用默认名字的符号链接.第二条规则保证hiddev设备节点建立于/dev/usb目录下.

### 从sysfs中查找信息

#### sysfs树

The concept of using interesting information from sysfs was briefly touched upon above. 要想编写基于sysfs信息的规则,首先你必需知道属性名和他们的当前值.

sys实际上是一个非常简单的结构.它逻辑上被划分为目录.每个目录包含一些属性文件,每个文件典型的只包含一个值.存在一些符号链接,把sysfs树的不同部分链接起来.

一些目录被作为顶层设备路径.These directories act as the top-level glue which chain other parts of sysfs to the device in question. Top-level device paths can be classified as sysfs directories which contain a dev file,一些命令将为您展示:

```
# find /sys -name dev
```

例如,在我的系统上,/sys/block/sda目录是我的硬盘设备目录.它通过符号链接/sys/block/sda/device和控制器建立联系,通过符号链接/sys/block/sda/device/driver/和设备驱动建立联系.

当你基于sysfs信息编写规则时,you are simply matching attribute contents of some files in one part of the chain. 比如,我可以通过以下方式来读取我的硬盘大小:

```
# cat /sys/block/sda/size
234441648
```

在一条udev规则中,我可以使用SYSFS{size}=="234441648"来识别硬盘.因为udev在整个设备链中进行递归,我们可以有选择的匹配该设备链的任何部分的属性来进行匹配(例如:/sys/class/block/sda/device/下的属性),however there are some caveats when dealing with different parts of the chain which are described later.

Although this serves as a useful introduction as to the structure of sysfs and exactly how udev matches values, manually trawling through sysfs is both time consuming and unnecessary.

posted on 2007-12-18 16:10 [lfc](#) 阅读(480) [评论\(1\)](#) [编辑](#) [收藏](#) [引用](#)

#### 评论:

# 高手有个问题 2008-03-01 19:04 | 驱动开发

我用的是au1200 2.6.11

我在用request\_firmware这个函数后该创建的节点都有了,可是过10秒返回谁败

我看了udev 安装没有问题

请高手指点

[givemeliu@yahoo.com.cn](mailto:givemeliu@yahoo.com.cn)


[回复](#) [更多评论](#)

[刷新评论列表](#)

标题

姓名

主页

验证码  \* 

内容(提交失败后,可以通过“恢复上次提交”恢复刚刚提交的内容)

Remember Me?

[登录](#) [使用高级评论](#) [新用户注册](#) [返回首页](#) [恢复上次提交](#)

[使用Ctrl+Enter键可以直接提交]

**[Free application access](#)**

2X ApplicationServer - Free 5 connection version download now:

[www.2x.com/ApplicationServer/](http://www.2x.com/ApplicationServer/)

Google 提供的广告

Powered by: [博客园](#) 模板提供: [沪江博客](#) Copyright ©2008 Ifc