[UDEV规则参数详细解释使用](http://blog.itpub.net/29065182/viewspace-1267927/)

 Mdev与udev的联系与区别：

mdev是udev的简化版本,是busybox中所带的程序,最适合用在嵌入式系统,而udev一般用在PC上的linux中,相对mdev来说要复杂些,devfs是2.4内核引入的,而在2.6内核中却被udev所替代,他们有着共同的优点,只是devfs中存在一些未修复的BUG,作者也停止了对他的维护,最显著的一个区别,采用devfs时,当一个并不存在的设备结点时,他却还能自动的加载对应的设备驱动,而udev则不能,udev认为当加载了不存在的对应的设备驱动的时候不应加载对应的驱动模块，因为加载也没有,浪费了资源.  
  从本质上来说，udev与mdev他们都是一个应用程序，配置了就可以使用,为了方便使用,我们可以使用busybox自带的mdev,当然也可以去下载udev的源码去编译和移植

Udev 的使用--linux系统创建设备节点

udev 和mdev 是两个使用uevent 机制处理热插拔问题的用户空间程序，两者的实现机理不同。udev 是基于netlink 机制的，它在系统启动时运行了一个deamon 程序udevd，通过监听内核发送的uevent 来执行相应的热拔插动作，包括创建/删除设备节点，加载/卸载驱动模块等等。

mdev 是基于uevent\_helper 机制的，它在系统启动时修改了内核中的uevnet\_helper 变量（通过写/proc/sys/kernel/hotplug），值为“/sbin/mdev”。这样内核产生uevent 时会调用uevent\_helper 所指的用户级程序，也就是mdev，来执行相应的热拔插动作。udev 使用的netlink 机制在有大量uevent 的场合效率高，适合用在PC 机上；而mdev 使用的uevent\_helper 机制实现简单，适合用在嵌入式系统中。另外要说明的一点是,uevent\_helper 的初始值在内核编译时时可配置的，默认值为/sbin/hotplug。如果想修改它的值，写/proc/sys/kernel/hotplug 文件就可以了

Linux 里都是以设备文件的形式存在。在早期的 Linux 版本中，/dev目录包含了所有可能出现的设备的设备文件。但因为这样 Linux 用户很难在这些大量的设备文件中找到匹配条件的设备文件。现在 udev 只为那些连接到 Linux 操作系统的设备产生设备文件。并且 udev 能通过定义一个 udev 规则 (rule) 来产生匹配设备属性的设备文件，这些设备属性可以是内核设备名称、总线路径、厂商名称、型号、序列号或者磁盘大小等等。

UDEV实现：

动态自动管理设备信息：当有设备添加 / 删除时，udev 的守护进程侦听到来自内核的 uevent 的事件，用来添加或者删除 /dev下的设备文件，所以 udev 可以只为已经连接的设备产生设备文件，

而不会象 2.4 内核一样在 /dev下产生大量设备文件。另外可以使用这个功能 ，当有设备加入时运行外部的程序，比如鼠标加入时自动禁用触摸板之类

使用自定义命名和管理设备：使用 Udev 规则文件，udev 在 /dev/ 里为所有的设备定义了内核设备名称，比如 /dev /sda、/dev/hda、/dev/fd等等。

由于 udev 是在用户空间 (user space) 运行，Linux 用户可以接下来对这些信息进行操作，比如可以通过自定义的规则文件，生成人性的设备标识，比如 /dev/my\_disk、/dev/nameusb 等,还能对设置进行参数成员用户组权限之类的修改。

开始之类需要了解

? sysfs：sysfs是 Linux 2.6 内核里的一个虚拟文件系统 (/sys)。它把设备和驱动的信息从内核的设备模块导出到用户空间 (userspace)。从该文件系统中，Linux 用户可以获取很多设备的属性。

? devpath：本文的 devpath是指一个设备在 sysfs文件系统 (/sys)下的相对路径，该路径包含了该设备的属性文件。udev 里的多数命令都是针对 devpath操作的。例如：sda的 devpath是 /block/sda，sda2 的 devpath是 /block/sda/sda2。

? 内核设备名称：设备在 sysfs里的名称，是 udev 默认使用的设备文件名。

udev 应用程序主配置文件：

主要的udev 主配置文件是 /etc/udev/udev.conf。这个文件通常很短，他可能只是包含几行#开头的注释，然后有几行选项：

udev\_root=/dev/ # 设置的绝对路径，相当于创建 chroot 的根。

udev\_rules=/etc/udev/rules.d/ #规则的存放地址

udev\_log=err # 日志的输入级别

udev 的规则配置文件实例

默认的规则配置文件存放在 /etc/udev/rules.d/ 中，我们进入这个可以看到 RedHat 默认对设备建好的一些规则和一些硬件公司写好的规则。

进入目录，可以见到以二位数字开头的前缀的配置文件，可以使用 vi 进入配置文件中查看，一行是一条规则,默认是从小数字到大数字，这些表示生效的顺序。

我们在使用 udev 写规则前，先来看一个例子

KERNEL==sd\*, PROGRAM=/lib/udev/scsi\_id -g -s %p, RESULT==123456, SYMLINK=%k\_%c

该规则的执行：如果有一个内核设备名称以 sd 开头，且 SCSI ID 为 123456，则为设备文件产生一个符号链接“sda\_123456”. %p %k %c 请看后面的“udev 的值和可调用的替换操作符 ”

udev 的规则配置文件

在规则文件里，除了以“#”开头的行（注释），所有的非空行都被视为一条规则，但是一条规则不能扩展到多行。规则都是由多个 键值对（key-value pairs）组成，并由逗号隔开，键值对可以分为 条件匹配键值对( 以下简称“匹配键 ”) 和 赋值键值对( 以下简称“赋值键 ”)，一条规则可以有多条匹配键和多条赋值键。匹配键是匹配一个设备属性的所有条件，当一个设备的属性匹配了该规则里所有的匹配键，就认为这条规则生效，然后按照赋值键的内容，执行该规则的赋值。

规则文件里的规则有一系列的键/值对组成，键/值对之间用逗号(,)分割。

通过上面例子中也能看出，这些配置，但我想大家可能会产生疑惑，为什么 KERNEL 是匹配键，而 NAME 和 MODE 是赋值键呢？这由中间的操作符 (operator) 决定。

仅当操作符是“==”或者“!=”时，其为匹配键；若为其他操作符时，都是赋值键。

匹配键和赋值键操作符解释见下表：

操作符 匹配或赋值t 解释

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

== 匹配 相等比较

!= 匹配 不等比较

= 赋值 分配一个特定的值给该键，他可以覆盖之前的赋值。

+= 赋值 追加特定的值给已经存在的键

:= 赋值 分配一个特定的值给该键，后面的规则不可能覆盖它。

udev 规则的匹配键 :

键 含义

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

ACTION 事件 (uevent) 的行为，例如：add( 添加设备 )、remove( 删除设备 )。

KERNEL 在内核里看到的设备名字，比如sd\*表示任意SCSI磁盘设备

DEVPATH 内核设备录进，比如/devices/\*

SUBSYSTEM 子系统名字，例如：sda 的子系统为 block。

BUS 总线的名字，比如IDE,USB

DRIVER 设备驱动的名字，比如ide-cdrom

ID 独立于内核名字的设备名字

SYSFS{ value} sysfs属性值，他可以表示任意

ENV{ key} 环境变量，可以表示任意

PROGRAM 可执行的外部程序，如果程序返回0值，该键则认为为真(true)

RESULT 上一个PROGRAM调用返回的标准输出。

NAME 根据这个规则创建的设备文件的文件名。

注意：仅仅第一行的NAME描述是有效的，后面的均忽略。

如果你想使用使用两个以上的名字来访问一个设备的话，可以考虑SYMLINK键。

SYMLINK 为 /dev/下的设备文件产生符号链接。由于 udev 只能为某个设备产生一个设备文件，

所以为了不覆盖系统默认的 udev 规则所产生的文件，推荐使用符号链接。

OWNER 设备文件的属组

GROUP 设备文件所在的组。

MODE 设备文件的权限，采用8进制

RUN 为设备而执行的程序列表

LABEL 在配置文件里为内部控制而采用的名字标签(下下面的GOTO服务)

GOTO 跳到匹配的规则（通过LABEL来标识），有点类似程序语言中的GOTO

IMPORT{ type} 导入一个文件或者一个程序执行后而生成的规则集到当前文件

WAIT\_FOR\_SYSFS 等待一个特定的设备文件的创建。主要是用作时序和依赖问题。

PTIONS 特定的选项：

last\_rule 对这类设备终端规则执行；

ignore\_device 忽略当前规则；

ignore\_remove 忽略接下来的并移走请求。

all\_partitions 为所有的磁盘分区创建设备文件。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

udev 的重要赋值键

引用

NAME：在 /dev下产生的设备文件名。只有第一次对某个设备的 NAME 的赋值行为生效，之后匹配的规则再对该设备的 NAME 赋值行为将被忽略。如果没有任何规则对设备的 NAME 赋值，udev 将使用内核设备名称来产生设备文件。

SYMLINK：为 /dev/下的设备文件产生符号链接。由于 udev 只能为某个设备产生一个设备文件，所以为了不覆盖系统默认的 udev 规则所产生的文件，推荐使用符号链接。

OWNER, GROUP, MODE：为设备设定权限。

ENV{key}：导入一个环境变量。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

我们给出一个列子来解释如何使用这些键。下面的例子来自Fedora Core 5系统的标准配置文件。

KERNEL==”\*”, OWNER=”root” GROUP=”root”, MODE=”0600″

KERNEL==”tty”, NAME=”%k”, GROUP=”tty”, MODE=”0666″, OPTIONS=”last\_rule”

KERNEL==”scd[0-9]\*”, SYMLINK+=”cdrom cdrom-%k”

KERNEL==”hd[a-z]”, BUS==”ide”, SYSFS{removable}==”1″, SYSFS{device/media}==”cdrom”, SYMLINK+=”cdrom cdrom-%k”

ACTION==”add”, SUBSYSTEM==”scsi\_device”, RUN+=”/sbin/modprobe sg”

上面的例子给出了5个规则，每一个都是KERNEL或者ACTION键开头：

\*第一个规则是缺省的，他匹配任意被内核识别到的设备，然后设定这些设备的属组是root，组是root，访问权限模式是0600(-rw——-)。这也是一个安全的缺省设置保证所有的设备在默认情况下只有root可以读写。

\*第二个规则也是比较典型的规则了。它匹配终端设备(tty)，然后设置新的权限为0600，所在的组是tty。它也设置了一个特别的设备文件名:%K。在这里例子里，%k代表设备的内核名字。那也就意味着内核识别出这些设备是什么名字，就创建什么样的设备文件名。

\*第三行开始的KERNEL==”scd[0-9]\*”,表示 SCSI CD-ROM 驱动. 它创建一对设备符号连接：cdrom和cdrom-%k。

\*第四行，开始的 KERNEL==”hd[a-z]“, 表示ATA CDROM驱动器。这个规则创建和上面的规则相同的符号连接。ATA CDROM驱动器需要sysfs值以来区别别的ATA设备，因为SCSI CDROM可以被内核唯一识别。.

\*第五行以 ACTION==”add”开始，它告诉udev增加 /sbin/modprobe sg 到命令列表，当任意SCSI设备增加到系统后，这些命令将执行。其效果就是计算机应该会增加sg内核模块来侦测新的SCSI设备。

当然，上面仅仅是一小部分例子，如果你的系统采用了udev方式，那你应该可以看到更多的规则。如果你想修改设备的权限或者创建信的符号连接，那么你需要熟读这些规则，特别是要仔细注意你修改的那些与之相关的设备。

udev 的值和可调用的替换操作符

在键值对中的键和操作符都介绍完了，最后是值 (value)。Linux 用户可以随意地定制 udev 规则文件的值。

例如：my\_root\_disk, my\_printer。同时也可以引用下面的替换操作符：

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

$kernel, %k：设备的内核设备名称，例如：sda、cdrom。

$number, %n：设备的内核号码，例如：sda3 的内核号码是 3。

$devpath, %p：设备的 devpath路径。

$id, %b：设备在 devpath里的 ID 号。

$sysfs{file}, %s{file}：设备的 sysfs里 file 的内容。其实就是设备的属性值。

例如：$sysfs{size} 表示该设备 ( 磁盘 ) 的大小。

$env{key}, %E{key}：一个环境变量的值。

$major, %M：设备的 major 号。

$minor %m：设备的 minor 号。

$result, %c：PROGRAM 返回的结果

$parent, %P：父设备的设备文件名。

$root, %r：udev\_root的值，默认是 /dev/。

$tempnode, %N：临时设备名。

%%：符号 % 本身。

$$：符号 $ 本身。

udev 规则所需要信息的查询

常用的查上面匹配键信息的命令

udevinfo -a -p $(udevinfo -q path -n /dev/sda1 )

上面的命令两次使用udevinfo：

第一次是返回sysfs设备路径(他通常和我们看到的Linux设备文件名所在路径－－/dev/hda－－不同)；

第二次才是查询这个设备路径，结果将是非常常的syfs信息汇总

udevinfo -a -p /sys/class/net/eth0

scsi\_id -g -s /block/sda

scsi\_id -g -x -s /block/sda/sda3

ata\_id /dev/hda

udev 信息的测试和生效

查出来后，根据上面文件中的内容写规则后，怎么测试

udevtest /block/sda

start\_dev #命令重启 udev守护进程

本操作会对所有的设备重新查询规则目录下所有的规则文件，然后执行所匹配的规则里的行为。

通常使用该命令让新的规则文件立即生效。

------------------------------------ 资料整理网络 ---------------------------------------------