使用 udev 高效、动态地管理 Linux 设备文件

黄 懋, 软件工程师, IBM

简介： 本文以通俗的方法阐述 udev 及相关术语的概念、udev 的配置文件和规则文件，然后以 Red Hat Enterprise Server 为平台演示一些管理设备文件和查询设备信息的实例。本文会使那些需要高效地、方便地管理 Linux 设备的用户受益匪浅，这些用户包括 Linux 最终用户、设备驱动开发人员、设备测试人员和系统管理员等等。

发布日期： 2010 年 3 月 02 日
级别： 初级

平均分 （共 17 个评分 ）

概述：

Linux 用户常常会很难鉴别同一类型的设备名，比如 eth0, eth1, sda, sdb 等等。通过观察这些设备的内核设备名称，用户通常能知道这些是什么类型的设备，但是不知道哪一个设备是他们想要的。例如，在一个充斥着本地磁盘和光纤磁盘的设备名清单 (`/dev/sd*`) 中，用户无法找到一个序列号为“35000c50000a7ef67”的磁盘。在这种情况下，udev 就能动态地在 `/dev`目录里产生自己想要的、标识性强的设备文件或者设备链接，以此帮助用户方便快捷地找到所需的设备文件。

udev 简介

什么是 udev?

udev 是 Linux2.6 内核里的一个功能，它替代了原来的 devfs，成为当前 Linux 默认的设备管理工具。udev 以守护进程的形式运行，通过侦听内核发出来的 uevent 来管理 `/dev`目录下的设备文件。不像之前的设备管理工具，udev 在用户空间 (user space) 运行，而不在内核空间 (kernel space) 运行。

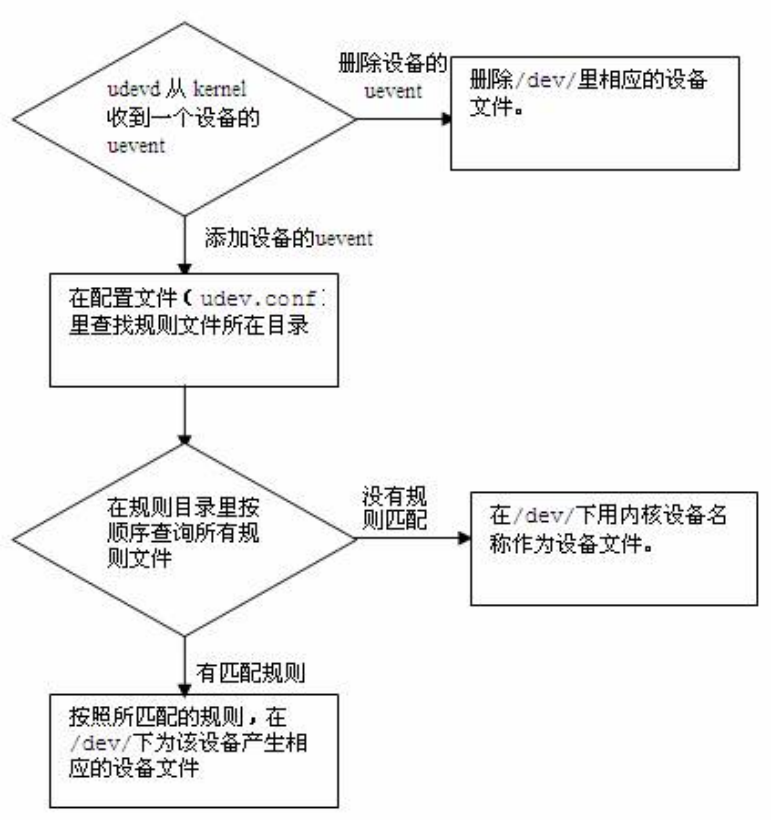
使用 udev 的好处：

我们都知道，所有的设备在 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。很难想象 Linux 用户如何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。

- 动态管理：当设备添加 / 删除时，udev 的守护进程侦听来自内核的 uevent，以此添加或者删除 `/dev`下的设备文件，所以 udev 只为已经连接的设备产生设备文件，而不会在 `/dev`下产生大量虚无的设备文件。
- 自定义命名规则：通过 Linux 默认的规则文件，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 用户可以通过自定义的规则文件，灵活地产生标识性强的设备文件名，比如 `/dev/boot_disk`、`/dev/root_disk`、`/dev/color_printer`等等。
- 设定设备的权限和所有者 / 组：udev 可以按一定的条件来设置设备文件的权限和设备文件所有者 / 组。在不同的 udev 版本中，实现的方法不同，在“如何配置和使用 udev”中会详解。

下面的流程图显示 udev 添加 / 删除设备文件的过程。

图 1. udev 工作流程图：



相关术语：

- *设备文件*：由于本文以较通俗的方式讲解 udev，所以设备文件是泛指在 /dev/下，可被应用程序用来和设备驱动交互的文件。而不会特别地区分设备文件、设备节点或者设备特殊文件。
- *devfs*：*devfs*是 Linux 早期的设备管理工具，已经被 udev 取代。
- *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。

如何配置和使用 udev

下面会以 RHEL4.8 和 RHEL5.3 为平台，分别描述 udev 的配置和使用：

下载和安装 udev

从 Fedora3 和 Red Hat Enterprise4 开始，udev 就是默认的设备管理工具，无需另外下载安装。

清单 1. 检查 udev 在 RHEL4.8 里的版本和运行情况

```
[root@HOST_RHEL4 dev]# rpm -qa |grep -i udev
udev-039-10.29.el4
[root@HOST_RHEL4 ~]# uname -r
2.6.9-89.ELsmp
[root@HOST_RHEL4 ~]# ps -ef |grep udev
root     21826     1  0 Dec09 ?        00:00:00 udevd
```

清单 2. 检查 udev 在 RHEL5.3 里的版本和运行情况

```
[root@HOST_RHEL5 ~]# rpm -qa |grep -i udev
udev-095-14.19.el5
[root@HOST_RHEL5 sysconfig]# uname -r
2.6.18-128.el5
[root@HOST_RHEL5 sysconfig]# ps -ef|grep udev
root      5466     1  0 18:32 ?      00:00:00 /sbin/udevd -d
```

如果 Linux 用户想更新 udev 包，可以从 http://www.kernel.org/pub/linux/utils/kernel/hotplug/下载并安装。

udev 的配置文件 (/etc/udev/udev.conf)

清单 3. RHEL 4 . 8 下 udev 的配置文件

```
[root@HOST_RHEL4 dev]# cat /etc/udev/udev.conf
# udev.conf
# The main config file for udev
#
# This file can be used to override some of udev's default values
# for where it looks for files, and where it places device nodes.
#
# WARNING: changing any value, can cause serious system breakage!
#

# udev_root - where in the filesystem to place the device nodes
udev_root="/dev/"

# udev_db - The name and location of the udev database.
udev_db="/dev/.udev.tdb"

# udev_rules - The name and location of the udev rules file
udev_rules="/etc/udev/rules.d/"

# udev_permissions - The name and location of the udev permission file
udev_permissions="/etc/udev/permissions.d/"

# default_mode - set the default mode for all nodes that have no
#                explicit match in the permissions file
default_mode="0600"

# default_owner - set the default owner for all nodes that have no
#                 explicit match in the permissions file
default_owner="root"

# default_group - set the default group for all nodes that have no
#                 explicit match in the permissions file
default_group="root"

# udev_log - set to "yes" if you want logging, else "no"
udev_log="no"
```

Linux 用户可以通过该文件设置以下参数：

- *udev_root*：udev 产生的设备所存放的目录，默认值是 `/dev/`。建议不要修改该参数，因为很多应用程序默认会从该目录调用设备文件。
- *udev_db*：udev 信息存放的数据库或者所在目录，默认值是 `/dev/.udev.tdb`。
- *udev_rules*：udev 规则文件的名字或者所在目录，默认值是 `/etc/udev/rules.d/`。
- *udev_permissions*：udev 权限文件的名字或者所在目录，默认值是 `/etc/udev/permissions.d/`。
- *default_mode/ default_owner/ default_group*：如果设备文件的权限没有在权限文件里指定，就使用该参数作为默认权限，默认值分别是：`0600/root/root`。
- *udev_log*：是否需要 *syslog*记录 udev 日志的开关，默认值是 no。

清单 4. RHEL5.3 下 udev 的配置文件

```
[root@HOST_RHEL5 ~]# cat /etc/udev/udev.conf
# udev.conf

# The initial syslog(3) priority: "err", "info", "debug" or its
# numerical equivalent. For runtime debugging, the daemons internal
# state can be changed with: "udevcontrol log_priority=<value>".
udev_log="err"
```

udev_log：*syslog*记录日志的级别，默认值是 err。如果改为 info 或者 debug 的话，会有冗长的 udev 日志被记录下来。

实际上在 RHEL5.3 里，除了配置文件里列出的参数 *udev_log*外，Linux 用户还可以修改参数 *udev_root*和 *udev_rules*(请参考上面的“RHEL4.8 的 udev 配置文件”)，只不过这 2 个参数是不建议修改的，所以没显示在 udev.conf 里。

可见该版本的 udev.conf 改动不小：*syslog*默认会记录 udev 的日志，Linux 用户只能修改日志的级别 (err、info、degub 等)；设备的权限不能在 udev.conf 里设定，而是要在规则文件 (*.rules) 里设定。

通过 udev 设定设备文件的权限

在 RHEL4.8 的 udev，设备的权限是通过权限文件来设置。

清单 5. RHEL4.8 下 udev 的权限文件

```
[root@HOST_RHEL4 ~]# cat /etc/udev/permissions.d/50-udev.permissions
……
# disk devices
hd*:root:disk:0660
sd*:root:disk:0660
dasd*:root:disk:0660
ataraid*:root:disk:0660
loop*:root:disk:0660
md*:root:disk:0660
ide/*/*/*/*/*:root:disk:0660
discs/*/*:root:disk:0660
loop/*:root:disk:0660
md/*:root:disk:0660

# tape devices
ht*:root:disk:0660
nht*:root:disk:0660
pt[0-9]*:root:disk:0660
npt*:root:disk:0660
st*:root:disk:0660
nst*:root:disk:0660
……
```

RHEL4.8 里 udev 的权限文件会为所有常用的设备设定权限和 ownership，如果有设备没有被权限文件设置权限，udev 就按照 udev.conf 里的默认权限值为这些设备设置权限。由于篇幅的限制，上图只显示了 udev 权限文件的一部分，该部分设 置了所有可能连接上的磁盘设备和磁带设备的权限和 ownership。

而在 RHEL5.3 的 udev，已经没有权限文件，所有的权限都是通过规则文件 (*.rules)来设置，在下面的规则文件配置过程会介绍到。

udev 的规则和规则文件

规则文件是 udev 里最重要的部分，默认是存放在 /etc/udev/rules.d/下。所有的规则文件必须以“.rules”为后缀名。RHEL 有默认的规则文件，这些默认规则文件不仅为设备产生内核设备名称，还会产生标识性强的符号链接。例如：

```
[root@HOST_RHEL5 ~]# ls /dev/disk/by-uuid/
16afe28a-9da0-482d-93e8-1a9474e7245c
```

但这些链接名较长，不易调用，所以通常需要自定义规则文件，以此产生易用且标识性强的设备文件或符号链接。

此外，一些应用程序也会在 /dev/下产生一些方便调用的符号链接。例如规则 40-multipath.rules 为磁盘产生下面的符号链接：

```
[root@ HOST_RHEL5 ~]# ls /dev/mpath/*
/dev/mpath/mpath0  /dev/mpath/mpath0p1  /dev/mpath/mpath0p2
```

udev 按照规则文件名的字母顺序来查询全部规则文件，然后为匹配规则的设备管理其设备文件或文件链接。虽然 udev 不会因为一个设备匹配了一条规则而停止解析后面的规则文件，但是解析的顺序仍然很重要。通常情况下，建议让自己想要的规则文件最先被解析。比如，创建一个名为 /etc/udev/rules.d/10-myrule.rules的文件，并把你的规则写入该文件，这样 udev 就会在解析系统默认的规则文件之前解析到你的文件。

RHEL5.3 的 udev 规则文件比 RHEL4.8 里的更完善。受篇幅的限制，同时也为了不让大家混淆，本文将不对 RHEL4.8 里的规则文件进行详解，下面关于规则文件的配置和实例都是在 RHEL5.3 上进行的。如果大家需要配置 RHEL4 的 udev 规则文件，可以先参照下面 RHEL5.3 的配置过程，然后查询 RHEL4 里的用户手册 (man udev) 后进行配置。

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。下面是一个简单的规则：

清单 6. 简单说明键值对的例子

```
KERNEL=="sda", NAME="my_root_disk", MODE="0660"
```

KERNEL 是匹配键，NAME 和 MODE 是赋值键。这条规则的意思是：如果有一个设备的内核设备名称为 sda，则该条件生效，执行后面的赋值：在 /dev下产生一个名为 my_root_disk的设备文件，并把设备文件的权限设为 0660。

通过这条简单的规则，大家应该对 udev 规则有直观的了解。但可能会产生疑惑，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。

- RHEL5.3 里 udev 规则的所有操作符：

  “==”：比较键、值，若等于，则该条件满足；

  “!=”： 比较键、值，若不等于，则该条件满足；

  “=”： 对一个键赋值；

  “+=”：为一个表示多个条目的键赋值。

  “:=”：对一个键赋值，并拒绝之后所有对该键的改动。目的是防止后面的规则文件对该键赋值。

- RHEL5.3 里 udev 规则的匹配键

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

KERNEL： 内核设备名称，例如：sda, cdrom。

DEVPATH：设备的 devpath 路径。

SUBSYSTEM： 设备的子系统名称，例如：sda 的子系统为 block。

BUS： 设备在 devpath 里的总线名称，例如：usb。

DRIVER： 设备在 devpath 里的设备驱动名称，例如：ide-cdrom。

ID： 设备在 devpath 里的识别号。

SYSFS{filename}： 设备的 devpath 路径下，设备的属性文件“filename”里的内容。

例如：SYSFS{model}==“ST936701SS”表示：如果设备的型号为 ST936701SS，则该设备匹配该 *匹配键*。

在一条规则中，可以设定最多五条 SYSFS 的 *匹配键*。

ENV{key}： 环境变量。在一条规则中，可以设定最多五条环境变量的 *匹配键*。

PROGRAM：调用外部命令。

RESULT： 外部命令 PROGRAM 的返回结果。例如：

```
PROGRAM=="/lib/udev/scsi_id -g -s $devpath", RESULT=="35000c50000a7ef67"
```

调用外部命令 `/lib/udev/scsi_id`查询设备的 SCSI ID，如果返回结果为 35000c50000a7ef67，则该设备匹配该 *匹配键*。

- RHEL5.3 里 udev 的重要赋值键

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。

- RHEL5.3 里 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。

清单 7. 说明替换操作符的规则例子

```
KERNEL=="sd*", PROGRAM="/lib/udev/scsi_id -g -s %p", \
RESULT=="35000c50000a7ef67", 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 `35000c50000a7ef67`，则为设备文件产生一个符号链接“sda_35000c50000a7ef67”.

制定 udev 规则和查询设备信息的实例：

如何查找设备的信息（属性）来制定 udev 规则：

当我们为指定的设备设定规则时，首先需要知道该设备的属性，比如设备的序列号、磁盘大小、厂商 ID、设备路径等等。通常我们可以通过以下的方法获得：

- **查询sysfs文件系统：**

  前面介绍过，sysfs 里包含了很多设备和驱动的信息。

  例如：设备 sda 的 SYSFS{size} 可以通过 `cat /sys/block/sda/size`得到；SYSFS{model} 信息可以通过 `cat /sys/block/sda/device/model`得到。

- **udevinfo命令：**

  udevinfo 可以查询 udev 数据库里的设备信息。例如：用 udevinfo 查询设备 sda 的 model 和 size 信息：

清单 8. 通过 udevinfo 查询设备属性的例子

```
[root@HOST_RHEL5 rules.d]# udevinfo -a -p /block/sda | egrep "model|size"
   SYSFS{size}=="71096640"
   SYSFS{model}=="ST936701SS      "
```

- **其他外部命令：**

  清单 9. 通过 scsi_id 查询磁盘的 SCSI_ID 的例子

```
[root@HOST_RHEL5 ~]# scsi_id -g -s /block/sda
35000c50000a7ef67
```

udev 的简单规则：

清单 10. 产生网卡设备文件的规则

```
SUBSYSTEM=="net", SYSFS{address}=="AA:BB:CC:DD:EE:FF", NAME="public_NIC"
```

该规则表示：如果存在设备的子系统为 net，并且地址 (MAC address) 为“AA:BB:CC:DD:EE:FF”，为该设备产生一个名为 public_NIC 的设备文件。

清单 11. 为指定大小的磁盘产生符号链接的规则

```
SUBSYSTEM=="block", SYSFS{size}=="71096640", SYMLINK ="my_disk"
```

该规则表示：如果存在设备的子系统为 block，并且大小为 71096640(block)，则为该设备的设备文件名产生一个名为 my_disk 的符号链接。

清单 12. 通过外部命令为指定序列号的磁盘产生设备文件的规则

```
KERNEL=="sd*[0-9]", PROGRAM=="/lib/udev/scsi_id -g -s %p", \
RESULT=="35000c50000a7ef67", NAME +="root_disk%n"
```

该规则表示：如果存在设备的内核设备名称是以 sd 开头 (磁盘设备)，以数字结尾 (磁盘分区)，并且通过外部命令查询该设备的 SCSI_ID 号为“35000c50000a7ef67”，则产生一个以 root_disk 开头，内核号码结尾的设备文件，并替换原来的设备文件（如果存在的话）。例如：产生设备名 /dev/root_disk2，替换原来的设备名 /dev/sda2。

运用这条规则，可以在 /etc/fstab里保持系统分区名称的一致性，而不会受驱动加载顺序或者磁盘标签被破坏的影响，导致操作系统启动时找不到系统分区。

其他常用的 udev 命令：

- **udevtest:**

  udevtest会针对一个设备，在不需要 uevent 触发的情况下模拟一次 udev的运行，并输出查询规则文件的过程、所执行的行为、规则文件的执行结果。通常使用 udevtest来调试规则文件。以下是一个针对设备 sda 的 udevtest例子。由于 udevtest是扫描所有的规则文件 (包括系统自带的规则文件)，所以会产生冗长的输出。为了让读者清楚地了解 udevtest，本例只在规则目录里保留一条规则：

  清单 13. 为 udevtest 保留的规则

```
KERNEL=="sd*", PROGRAM="/lib/udev/scsi_id -g -s %p", RESULT=="35000c50000a7ef67", \
NAME="root_disk%n", SYMLINK="symlink_root_disk%n"
```

  清单 14. udevtest 的执行过程

```
[root@HOST_RHEL5 rules.d]# udevtest /block/sda
main: looking at device '/block/sda' from subsystem 'block'
run_program: '/lib/udev/scsi_id -g -s /block/sda'
run_program: '/lib/udev/scsi_id' (stdout) '35000c50000a7ef67'
run_program: '/lib/udev/scsi_id' returned with status 0
udev_rules_get_name: reset symlink list
udev_rules_get_name: add symlink 'symlink_root_disk'
udev_rules_get_name: rule applied, 'sda' becomes 'root_disk'
udev_device_event: device '/block/sda' already in database, \
                   validate currently present symlinks
udev_node_add: creating device node '/dev/root_disk', major = '8', \
           minor = '0', mode = '0660', uid = '0', gid = '0'
udev_node_add: creating symlink '/dev/symlink_root_disk' to 'root_disk'
```

可以看出，udevtest对 sda 执行了外部命令 scsi_id, 得到的 stdout 和规则文件里的 RESULT 匹配，所以该规则匹配。然后 (模拟) 产生设备文件 /dev/root_disk和符号链接 /dev/symlink_root_disk，并为其设定权限。

- **start_udev:**

  start_dev命令重启 udev守护进程，并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：

  清单 15. start_udev 的执行过程

```
[root@HOST_RHEL5 rules.d]# start_udev
Starting udev:                                             [  OK  ]
```

  start_udev一般没有标准输出，所有的 udev 相关信息都按照配置文件 (udev.conf)的参数设置，由 *syslog*记录。

小结：

udev 是高效的设备管理工具，其最大的优势是动态管理设备和自定义设备的命名规则，因此替代 devfs 成为 Linux 默认的设备管理工具。通过阅读本文，Linux 用户能够了解到 udev 的工作原理和流程，灵活地运用 udev 规则文件，从而方便地管理 Linux 设备文件。

参考资料

- 有关 Udev 更多信息，请参考：http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html。
- 在 developerWorks Linux 专区 寻找为 Linux 开发人员（包括 Linux 新手入门）准备的更多参考资料，查阅我们 最受欢迎的文章和教程。
- 在 developerWorks 上查阅所有 Linux 技巧 和 Linux 教程。

关于作者

黄懋，IBM 中国系统和技术实验室，Open Systems Interoperability Validation Lab 存储测试团队的成员。3-4 年 Linux/Unix 平台和 SAN/NAS 存储产品的测试经验。