第 12 章使用 udev 进行动态内核设备管理

第 12 章使用 udev 进行动态内核设备管理¶

内核几乎可以添加或删除运行系统中的任何设备。设备状态的更改（无论插入还是删除设备）需要传播给用户空间。插入或者识别设备后需要进行配置。某个设备已识别状态的任何更改都需要通知给此设备的用户。udev 可提供所需的基础结构来动态维护 /dev 目录中的设备节点文件和符号链接。udev 规则提供了将外部工具插入内核设备事件处理的方式。这使您能够自定义 udev 设备处理，例如通过添加特定脚本来作为内核设备处理的一部分执行，或者请求并导入额外数据以在设备处理期间评估。

/dev 目录中的设备节点提供对相应的内核设备的访问。使用 udev 时，/dev 目录反映内核的当前状态。每个内核设备都有相应的设备文件。如果设备从系统断开，则删除此设备节点。

/dev 目录的内容保存在临时文件系统中，所有文件都是在每个系统启动时提供的。手动创建或修改的文件在重引导时是有意不保存的。无论相应内核设备的状态如何都出现在 /dev 目录中的静态文件和目录，可以放置在 /lib/udev/devices 目录中。系统启动时，此目录的内容复制到 /dev 目录，它们与 /lib/udev/devices 中的文件具有相同的所有权和许可权限。

必需的设备信息由 sysfs 文件系统导出。对于内核检测到并已初始化的设备，将创建一个带有该设备名称的目录。它包含带有特定于设备属性的属性文件。

每次添加或删除设备时，内核都会发送 uevent 来向 udev 通知更改。一旦启动，udev 守护程序便会读取和分析 /etc/udev/rules.d/*.rules 文件中提供的所有规则，并将它们保留在内存中。如果更改、添加或删除了规则文件，则守护程序可以使用命令 **udevadm control reload_rules** 重新装载所有规则在内存中的表示形式。运行 **/etc/init.d/boot.udev reload** 时也会执行此操作。有关 udev 规则及其语法的更多细节，请参见第 12.6 节 “使用 udev 规则影响内核设备事件处理”。

每个接收到的事件都根据所提供的规则集进行匹配。这些规则可以增加或更改事件环境键、为要创建的设备节点请求特定名称、添加指向该节点的符号链接或者添加设备节点创建后运行的程序。从内核 netlink 套接字接收驱动程序核心 uevent。

设备的内核总线驱动程序探测。对于每个检测到的设备，内核都会在驱动程序核心将 uevent 发送到 udev 守护程序时创建内部设备结构。总线设备通过特殊格式的 ID 来标识自己，这可以识别设备的类型。通常，这些 ID 由供应商和产品 ID 以及其他特定于子系统的值组成。每个总线都有自己对于这些 ID 的方案，称为 MODALIAS。内核获取设备信息，由此组成一个 MODALIAS ID 字符串，并将该字符串与事件一起发送。对于 USB 鼠标，如下所示：

```
MODALIAS=usb:v046DpC03Ed2000dc00dsc00dp00ic03isc01ip02
```

每个设备驱动程序都带有它可以处理的设备的已知别名列表。这个列表包含在内核模块文件中。程序 depmod 读取 ID 列表并在内核的 /lib/modules 目录中为所有当前可用的模块创建文件 modules.alias。使用这种基础结构，模块的装载就如为每个带有 **MODALIAS** 关键字的事件调用 modprobe 一样简单。如果调用 **modprobe $MODALIAS**，它将组成该设备的设备别名与模块提供的别名相匹配。如果找到匹配的项，则装载该模块。所有这些操作均由 udev 自动触发。

在 udev 守护程序运行之前的引导进程中发生的所有设备事件都会丢失，因为处理这些事件的基础结构保存在 root 文件系统中，并且此时不可用。为了弥补此损失，内核提供了一个 uevent 文件，该文件位于 sysfs 文件系统每个设备的设备目录中。通过将 add 写入到该文件，内核将再次发送引导时丢失的相同事件。/sys 触发器中所有 uevent 文件的简单循环将再次触发所有事件来创建设备节点并执行设备设置。

例如，在引导期间出现的 USB 鼠标可能不会由早期引导逻辑初始化，因为驱动程序在那时不可用。此设备发现的事件丢失并且不能为该设备查找内核模块。不用手动搜索可能连接的设备，udev 会在 root 文件系统可用后直接从内核请求所有设备事件，以便 USB 鼠标设备的事件可以再次运行。现在它在装入的 root 文件系统上找到内核模块，因此可以初始化 USB 鼠标。

在用户空间，设备冷插入序列和运行时期间发现的设备之间没有明显的区别。在这两种情况下，使用相同的规则来匹配并且运行相同的配置程序。

程序 **udevadm monitor** 可以用于将驱动程序核心事件和 udev 事件处理的计时可视化。

```
UEVENT[1185238505.276660] add   /devices/pci0000:00/0000:00:1d.2/usb3/3-1 (usb)
UDEV  [1185238505.279198] add   /devices/pci0000:00/0000:00:1d.2/usb3/3-1 (usb)
```

```
UEVENT[1185238505.279527] add   /devices/pci0000:00/0000:00:1d.2/usb3/3-1/3-1:1.0 (usb)
UDEV  [1185238505.285573] add   /devices/pci0000:00/0000:00:1d.2/usb3/3-1/3-1:1.0 (usb)
UEVENT[1185238505.298878] add   /devices/pci0000:00/0000:00:1d.2/usb3/3-1/3-1:1.0/input/input10 (input)
UDEV  [1185238505.305026] add   /devices/pci0000:00/0000:00:1d.2/usb3/3-1/3-1:1.0/input/input10 (input)
UEVENT[1185238505.305442] add   /devices/pci0000:00/0000:00:1d.2/usb3/3-1/3-1:1.0/input/input10/mouse2 (input)
UEVENT[1185238505.306440] add   /devices/pci0000:00/0000:00:1d.2/usb3/3-1/3-1:1.0/input/input10/event4 (input)
UDEV  [1185238505.325384] add   /devices/pci0000:00/0000:00:1d.2/usb3/3-1/3-1:1.0/input/input10/event4 (input)
UDEV  [1185238505.342257] add   /devices/pci0000:00/0000:00:1d.2/usb3/3-1/3-1:1.0/input/input10/mouse2 (input)
```

UEVENT 行显示内核已经通过 netlink 发送的事件。UDEV 行显示已经完成的 udev 事件处理程序。计时以微秒为单位显示。UEVENT 和 UDEV 之间的时间是 udev 用于处理此事件或者 udev 守护程序延迟执行从而同步此事件与相关以及已运行的事件的时间。例如，硬盘分区的事件总是等待主磁盘设备事件完成，因为分区事件可能依赖于主磁盘事件从硬件查询的数据。

udevadm monitor --env 显示完整的事件环境：

```
ACTION=add
DEVPATH=/devices/pci0000:00/0000:00:1d.2/usb3/3-1/3-1:1.0/input/input10
SUBSYSTEM=input
SEQNUM=1181
NAME="Logitech USB-PS/2 Optical Mouse"
PHYS="usb-0000:00:1d.2-1/input0"
UNIQ=""
EV=7
KEY=70000 0 0 0 0
REL=103
MODALIAS=input:b0003v046DpC03Ee0110-e0,1,2,k110,111,112,r0,1,8,amlsfw
```

udev 也将消息发送给 syslog。用于控制将哪些消息发送到系统日志的默认系统日志优先级在 udev 配置文件 /etc/udev/udev.conf 中指定。可以使用 **udevadm control log_priority=*level/number*** 更改正在运行的守护程序的日志优先权。

12.6. 使用 udev 规则影响内核设备事件处理¶

udev 规则可以与内核添加到事件本身的属性或者内核导出到 sysfs 的任何信息匹配。规则还可以从外部程序请求其他信息。根据提供的规则匹配每个事件。所有规则都位于 /etc/udev/rules.d 目录下。

规则文件中的每一行至少包含一个关键字值对。有两种类型的关键字，匹配关键字和指派关键字。如果所有匹配关键字与它们的值匹配，则应用此规则并将指派关键字指派给特定的值。匹配规则可以指定设备节点的名称、添加指向该节点的符号链接或者运行作为事件处理一部分的特定程序。如果找不到匹配的规则，则使用默认设备节点名来创建设备节点。udev 手册页中描述了有关规则语法和提供用来与数据匹配或导入数据的关键字的详细信息。以下示例规则提供了 udev 规则语法的基本介绍。这些示例规则全部取自 /etc/udev/rules.d/50-udev-default.rules 下的 udev 默认规则集。

例 **12.1.** 示例 **udev** 规则¶

```
# console
KERNEL=="console", MODE="0600", OPTIONS="last_rule"

# serial devices
KERNEL=="ttyUSB*", ATTRS{product}=="[Pp]alm*Handheld*", SYMLINK+="pilot"

# printer
SUBSYSTEM=="usb", KERNEL=="lp*", NAME="usb/%k", SYMLINK+="usb%k", GROUP="lp"

# kernel firmware loader
SUBSYSTEM=="firmware", ACTION=="add", RUN+="firmware.sh"
```

console 规则由三个键构成：一个匹配键 (KERNEL) 和两个赋值键（MODE、OPTIONS）。KERNEL 匹配规则搜索设备列表以查找类型为 console 的所有项。只有完全匹配才有效，才能触发执行此规则。在这种情况下，MODE 关键字为设备节点指派特殊权限，仅为此设备的拥有者指派读写权限。OPTIONS 关键字将该规则标记为此类型的所有设备最后采用的规则。匹配此特殊设备类型的任何后续规则都不产生任何影响。

50-udev-default.rules 中不再提供 serial devices 规则，但该规则仍然值得考虑。该规则由两个匹配关键字（KERNEL 和 ATTRS）和一个赋值关键字 (SYMLINK) 构成。KERNEL 关键字搜索类型为 ttyUSB 的所有设备。该关键字使用 * 通配符匹配这些设备中的几个。第二个匹配关键字 ATTRS 检查任何 ttyUSB 设备的 sysfs 中的 product 属性文件是否包含特定字符串。赋值关键字 (SYMLINK) 将符号链接添加至该设备的 /dev/pilot 下。此关键字中使用的运算符 (+=) 告知 udev 进一步执行此操作，即使前面或后面的规则添加其他符号链接。由于此规则包含两个匹配关键字，因此仅当两个条件都满足时，才应用。

printer 规则处理 USB 打印机，其中包含两个匹配关键字（SUBSYSTEM 和 KERNEL），并且必须同时应用这两个关键字，才能应用整个规则。三个赋值键处理该设备类型的命名 (NAME)、符号设备链接 (SYMLINK) 的创建，以及此设备类型的组成员资格 (GROUP)。在 KERNEL 关键字中使用通配符 * 将使其匹配若干 lp 打印机设备。NAME 和 SYMLINK 关键字中都使用了替换项，以便按内部设备名称扩展这些字符串。例如，指向第一个 lp USB 打印机的符号链接为 /dev/usblp0。

kernel firmware loader 规则用于使 udev 在运行时期间通过外部助手脚本装载其他固件。SUBSYSTEM 匹配关键字搜索 firmware 子系统。ACTION 关键字检查是否添加了属于 firmware 子系统的任何设备。RUN+= 关键字触发执行 firmware.sh 脚本，以便找到应装载的固件。

所有规则具有一些共同的特征：

- 每个规则由一个或多个以逗号分隔的关键字值对构成。
- 关键字的运算由运算符确定。udev 规则支持多个不同的运算符。
- 每个给定值必须用引号引起来。
- 规则文件的每一行代表一个规则。如果一个规则超过一行，请使用 \ 合并不同行，就像在壳层语法中一样。
- udev 规则支持与 *、? 和 [] 模式匹配的外壳式模式。
- udev 规则支持替换。

创建可以从若干不同运算符选择的关键字，具体取决于希望创建的关键字类型。匹配关键字通常仅用于查找匹配或明显不匹配搜索值的值。匹配关键字包含以下运算符之一：

==

比较等于性。如果关键字包含搜索模式，则匹配该模式的所有结果均有效。

!=

比较不等于性。如果关键字包含搜索模式，则匹配该模式的所有结果均有效。

赋值关键字可以使用下面的任何运算符：

=

为关键字指派值。如果关键字以前由一列值构成，关键字将重置，并且仅指派一个值。

+=

为包含一列项的关键字添加一个值。

:=

指派最终值。不允许后面的规则进行任何后续更改。

udev 规则支持使用占位符和替换项。请按照在其他任何脚本中的相同方式使用。在 udev 规则中可使用以下替换项：

%r、$root

设备目录 /dev（默认）。

%p、$devpath

DEVPATH 的值。

%k、$kernel

KERNEL 的值或内部设备名称。

%n、$number

设备号。

%N、$tempnode

设备文件的临时名称。

%M、$major

设备的主编号。

%m、$minor

设备的次编号。

%s{*attribute*}/$attr{*attribute*}

sysfs 属性的值(由 *attribute* 指定)。

%E{*variable*}、$attr{*variable*}

环境变量的值(由 *variable* 指定)。

%c、$result

PROGRAM 的输出。

%%

% 字符。

$$

$ 字符。

匹配关键字描述应用 udev 规则之前必须满足的条件。以下匹配关键字可用:

ACTION

事件操作的名称,如 add 或 remove(添加或删除设备时)。

DEVPATH

事件设备的设备路径,如 DEVPATH=/bus/pci/drivers/ipw3945,用于搜索与 ipw3945 驱动程序有关的所有事件。

KERNEL

事件设备的内部(内核)名称。

SUBSYSTEM

事件设备的子系统,如 SUBSYSTEM=usb(用于与 USB 设备有关的所有事件)。

ATTR{*filename*}

事件设备的 sysfs 属性。例如,要匹配 vendor 属性文件名中包含的字符串,可以使用 ATTR{vendor}=="On[sS]tream"。

KERNELS

让 udev 向上搜索设备路径以查找匹配的设备名称。

SUBSYSTEMS

让 udev 向上搜索设备路径以查找匹配的设备子系统名称。

DRIVERS

让 udev 向上搜索设备路径以查找匹配的设备驱动程序名称。

ATTRS{*filename*}

让 udev 向上搜索设备路径以查找具有匹配的 sysfs 属性值的设备。

ENV{*key*}

环境变量的值，如 ENV{ID_BUS}="ieee1394"，用于搜索与该 FireWire 总线 ID 有关的所有事件。

PROGRAM

让 udev 执行外部程序。程序必须返回退出码零，才能成功。程序的输出（打印到 stdout）可用于 RESULT 关键字。

RESULT

匹配上次 PROGRAM 调用的输出字符串。在与 PROGRAM 关键字相同的规则中包含该关键字，或在后面的一个中。

与上述匹配键相比，赋值键未描述必须满足的条件。它们将值、名称和操作指派给由 udev 维护的设备节点。

NAME

将创建的设备节点的名称。在一个规则设置节点名称之后，将对该节点忽略带有 NAME 关键字的其他所有规则。

SYMLINK

与要创建的节点有关的符号链接名称。多个匹配的规则可添加要使用设备节点创建的符号链接。也可以通过使用空格字符分隔符号链接名称，在一个规则中为一个节点指定多个符号链接。

OWNER, GROUP, MODE

新设备节点的权限。此处指定的值重写已编译的任何值。

ATTR{*key*}

指定要写入事件设备的 sysfs 属性的值。如果使用 == 运算符，也将使用该关键字匹配 sysfs 属性的值。

ENV{*key*}

告知 udev 将变量导出到环境。如果使用 == 运算符，也将使用该关键字匹配环境变量。

RUN

告知 udev 向程序列表添加要为该设备执行的程序。请记住，将此程序限制于很短的任务，以免妨碍此设备的后续事件。

LABEL

添加 GOTO 可跳至的标签。

GOTO

告知 udev 跳过一些规则，继续执行具有按 GOTO 关键字引用的标签的规则。

IMPORT{*type*}

将变量装载入外部程序输出之类的事件环境中。udev 导入不同类型的若干变量。如果未指定任何类型，udev 将尝试根据文件许可权限的可执行位来自行确定类型。

- program 告知 udev 执行外部程序并导入其输出。
- file 告知 udev 导入文本文件。
- parent 告知 udev 从父设备导入储存的关键字。

WAIT_FOR_SYSFS

告知 udev 等待要为某个设备创建的指定 sysfs 文件。例如，WAIT_FOR_SYSFS="ioerr_cnt" 通知 udev 等待 ioerr_cnt 文件创建完成。

OPTIONS

OPTION 关键字可能有若干值：

- last_rule 告知 udev 忽略后面的所有规则。
- ignore_device 告知 udev 完全忽略此事件。
- ignore_remove 告知 udev 忽略后面针对设备的所有删除事件。
- all_partitions 告知 udev 为块设备上的所有可用分区创建设备节点。

动态设备目录和 udev 规则基础结构可以为所有磁盘设备提供固定名称，而不考虑它们的识别顺序或设备使用的连接。内核创建的每个相应的块设备由工具根据有关特定总线、驱动器类型或者文件系统的特殊知识进行检查。除了动态内核提供的设备节点名，udev 还维护各种指向该设备的永久符号链接：

```
/dev/disk
|-- by-id
|   |-- scsi-SATA_HTS726060M9AT00_MRH453M4HWHG7B -> ../../sda
|   |-- scsi-SATA_HTS726060M9AT00_MRH453M4HWHG7B-part1 -> ../../sda1
|   |-- scsi-SATA_HTS726060M9AT00_MRH453M4HWHG7B-part6 -> ../../sda6
|   |-- scsi-SATA_HTS726060M9AT00_MRH453M4HWHG7B-part7 -> ../../sda7
|   |-- usb-Generic_STORAGE_DEVICE_02773 -> ../../sdd
|   `-- usb-Generic_STORAGE_DEVICE_02773-part1 -> ../../sdd1
|-- by-label
|   |-- Photos -> ../../sdd1
|   |-- SUSE10 -> ../../sda7
|   `-- devel -> ../../sda6
|-- by-path
|   |-- pci-0000:00:1f.2-scsi-0:0:0:0 -> ../../sda
|   |-- pci-0000:00:1f.2-scsi-0:0:0:0-part1 -> ../../sda1
|   |-- pci-0000:00:1f.2-scsi-0:0:0:0-part6 -> ../../sda6
|   |-- pci-0000:00:1f.2-scsi-0:0:0:0-part7 -> ../../sda7
|   |-- pci-0000:00:1f.2-scsi-1:0:0:0 -> ../../sr0
|   |-- usb-02773:0:0:2 -> ../../sdd
|   |-- usb-02773:0:0:2-part1 -> ../../sdd1
`-- by-uuid
    |-- 159a47a4-e6e6-40be-a757-a629991479ae -> ../../sda7
    |-- 3e999973-00c9-4917-9442-b7633bd95b9e -> ../../sda6
    `-- 4210-8F8C -> ../../sdd1
```

`/sys/*`

Linux 内核提供的虚拟文件系统，用于导出所有当前已知设备。此信息由 udev 用于在 /dev 中创建设备节点

`/dev/*`

动态创建的设备节点和引导时从 /lib/udev/devices/* 复制的静态内容

以下文件和目录包含 udev 基础结构的关键元素：

`/etc/udev/udev.conf`

主 udev 配置文件。

`/etc/udev/rules.d/*`

udev 事件匹配规则.

`/lib/udev/devices/*`

静态 /dev 内容。

`/lib/udev/*`

从 udev 规则调用的帮助程序。

有关 udev 基础结构的更多信息，请参见以下手册页：

`udev`

有关 udev、关键字、规则和其他重要配置问题的常规信息。

udevadm

udevadm 可用于控制 udev 的运行时行为、请求内核事件、管理事件队列，以及提供简单的调试机制。

udevd

有关 udev 事件管理守护程序的信息。