Oracle Linux 10

Managing System Devices With udev

G14598-01
June 2025

Oracle Linux 10 Managing System Devices With udev,

G14598-01

Contents

Preface

Documentation License iv
Conventions iv
Documentation Accessibility iv
Access to Oracle Support for Accessibility iv
Diversity and Inclusion iv

1 About the udev Device Manager

About Device Files 1-1

2 Querying udev

Prefixes for udevadm Information 2-1
View All Information for a Device 2-1
Limiting Device Information by Query Type 2-2
View Attributes for a Device and Its Parent Devices 2-4

3 Working With udev Rules

Assignment and Comparison Operators 3-2
Pattern-Matching Characters 3-2
Common Match Keys 3-2
Common Assignment Keys 3-3
String Substitutions 3-5

4 Customizing udev Rules

Preface

Oracle Linux 10: Managing Devices with Udev describes how the udev device manager dynamically creates or removes device node files according to rules. Instructions are provided to help you query udev and create changes to udev rules, as required.

Documentation License

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also

mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

About the udev Device Manager

The udev device manager runs as a systemd service to help provide software with predictable and managed access to system devices exposed by the kernel. Typically. udev manages the permissions of device nodes, creates symbolic links in the `/dev/` directory to make device names more predictable and easier to identify, or renames network interfaces.

The udev device manager dynamically creates or removes device node files at boot time. When creating a device node, udev reads the device's `/sys` directory for attributes such as the label, serial number, and bus device number.

udev can use persistent device names to guarantee consistent naming of devices across reboots, regardless of their order of discovery. Persistent device names are especially important when using external storage devices.

udev also handles device driver events that are triggered by the kernel and uses the rules defined in its configuration to trigger particular actions. For example, if a USB storage device is connected to the system, the kernel notifies udev and udev notifies the appropriate handler so that the device can be mounted. Also, if a network cable is connected to a network interface card, the kernel notifies udev of the state change and udev notifies NetworkManager so that the appropriate action can take place to connect to the network.

The configuration file for udev is `/etc/udev/udev.conf`, in which you can define the `udev_log` logging priority, which can be set to `err`, `info` and `debug`. Note that the default value is `err`. Further configuration of rules used by udev are handled in individual rules files in `/etc/udev/rules.d/`.

For more information, see the `udev(7)` manual page.

udev is a component of systemd, also see Oracle Linux 10: System Management with systemd.

For more information about the kernel virtual file systems and device driver modules, see .

About Device Files

The `/dev` directory contains *device files* or *device nodes* that provide access to peripheral devices such as hard disks, to resources on peripheral devices such as disk partitions, and pseudo devices such as a random number generator.

The `/dev` directory has several subdirectory hierarchies, each of which holds device files that relate to a certain type of device. However, the contents of these subdirectories are implemented as symbolic links to corresponding files in `/dev`. Thus, the files can be accessed either through the linked file in `/dev` or the corresponding file in the subdirectory.

Using the `ls -l /dev` command lists files, some of which are flagged as being either type `b` (for *block*) or type `c` (for *character*). These devices have an associated pair of numbers that identify the device to the system.

```
ls -l /dev
```

```
total 0
crw-r--r--. 1 root root     10, 235 Aug 20 08:36 autofs
drwxr-xr-x. 2 root root         240 Sep 20 07:37 block
drwxr-xr-x. 2 root root         100 Aug 20 08:36 bsg
drwxr-xr-x. 3 root root          60 Nov  4  2019 bus
lrwxrwxrwx. 1 root root           3 Aug 20 08:36 cdrom -> sr0
drwxr-xr-x. 2 root root        2720 Sep 20 07:37 char
crw-------. 1 root root      5,   1 Aug 20 08:36 console
lrwxrwxrwx. 1 root root          11 Aug 20 08:36 core -> /proc/kcore
drwxr-xr-x. 3 root root          60 Nov  4  2019 cpu
crw-------. 1 root root     10,  62 Aug 20 08:36 cpu_dma_latency
drwxr-xr-x. 7 root root         140 Aug 20 08:36 disk
brw-rw----. 1 root disk    253,   0 Aug 20 08:36 dm-0
brw-rw----. 1 root disk    253,   1 Aug 20 08:36 dm-1
brw-rw----. 1 root disk    253,   2 Aug 20 08:36 dm-2
lrwxrwxrwx. 1 root root          13 Aug 20 08:36 fd -> /proc/self/fd
crw-rw-rw-. 1 root root      1,   7 Aug 20 08:36 full
crw-rw-rw-. 1 root root     10, 229 Aug 20 08:36 fuse
crw-------. 1 root root     10, 228 Aug 20 08:36 hpet
drwxr-xr-x. 2 root root           0 Aug 20 08:36 hugepages
crw-------. 1 root root     10, 183 Aug 20 08:36 hwrng
lrwxrwxrwx. 1 root root          12 Aug 20 08:36 initctl -> /run/initctl
drwxr-xr-x. 3 root root         220 Aug 20 08:36 input
crw-r--r--. 1 root root      1,  11 Aug 20 08:36 kmsg
lrwxrwxrwx. 1 root root          28 Aug 20 08:36 log -> /run/systemd/journal/
dev-log
brw-rw----. 1 root disk      7,   0 Sep 23 01:28 loop0
crw-rw----. 1 root disk     10, 237 Sep 20 07:37 loop-control
drwxr-xr-x. 2 root root         120 Aug 20 08:36 mapper
crw-------. 1 root root     10, 227 Aug 20 08:36 mcelog
crw-r-----. 1 root kmem      1,   1 Aug 20 08:36 mem
crw-------. 1 root root     10,  59 Aug 20 08:36 memory_bandwidth
drwxrwxrwt. 2 root root          40 Nov  4  2019 mqueue
drwxr-xr-x. 2 root root          60 Aug 20 08:36 net
crw-------. 1 root root     10,  61 Aug 20 08:36 network_latency
crw-------. 1 root root     10,  60 Aug 20 08:36 network_throughput
crw-rw-rw-. 1 root root      1,   3 Aug 20 08:36 null
crw-------. 1 root root     10, 144 Aug 20 08:36 nvram
drwxr-xr-x. 2 root root         100 Aug 20 08:36 ol_ca-virtdoc-oltest1
crw-r-----. 1 root kmem      1,   4 Aug 20 08:36 port
crw-------. 1 root root    108,   0 Aug 20 08:36 ppp
crw-rw-rw-. 1 root tty       5,   2 Oct  7 08:10 ptmx
drwxr-xr-x. 2 root root           0 Aug 20 08:36 pts
crw-rw-rw-. 1 root root      1,   8 Aug 20 08:36 random
drwxr-xr-x. 2 root root          60 Nov  4  2019 raw
lrwxrwxrwx. 1 root root           4 Aug 20 08:36 rtc -> rtc0
crw-------. 1 root root    251,   0 Aug 20 08:36 rtc0
brw-rw----. 1 root disk      8,   0 Aug 20 08:36 sda
brw-rw----. 1 root disk      8,   1 Aug 20 08:36 sda1
```

```
brw-rw----. 1 root disk      8,   2 Aug 20 08:36 sda2
brw-rw----. 1 root disk      8,  16 Aug 20 08:36 sdb
brw-rw----. 1 root disk      8,  17 Aug 20 08:36 sdb1
crw-rw----. 1 root cdrom    21,   0 Aug 20 08:36 sg0
```

Block Devices

Block devices enable random access to data, seeking media for data, and typically buffers data while data is being written or read. Examples of block devices include hard disks, CD-ROM drives, flash memory, and other addressable memory devices.

Character Devices

Character devices enable the streaming of data to or from a device. The data isn't typically buffered nor is random access granted to data on a device. The kernel writes data to or reads data from a character device 1 byte at a time. Examples of character devices include keyboards, mice, terminals, pseudo terminals, and tape drives. `tty0` and `tty1` are character device files that correspond to terminal devices so users can log in from serial terminals or terminal emulators.

Pseudo-Terminal Character Devices

Pseudo terminals secondary devices emulate real terminal devices to interact with software. For example, a user might log in to a terminal device such as `/dev/tty1`, which then uses the pseudo terminal primary device, `/dev/pts/ptmx`, to interact with an underlying pseudo terminal device. The character device files for pseudo terminal secondary and primary devices are in the `/dev/pts` directory, as shown in the following example:

```
ls -l /dev/pts
```

```
total 0
crw--w----. 1 guest tty  136, 0 Mar 17 10:11 0
crw--w----. 1 guest tty  136, 1 Mar 17 10:53 1
crw--w----. 1 guest tty  136, 2 Mar 17 10:11 2
c---------. 1 root  root   5, 2 Mar 17 08:16 ptmx
```

Some device entries, such as `stdin` for the standard input, are symbolically linked through the `self` subdirectory of the `proc` file system. The pseudo-terminal device file to which they point depends on the context of the process.

```
ls -l /proc/self/fd/[012]
```

```
lrwx------. 1 root root 64 Oct  7 08:23 /proc/self/fd/0 -> /dev/pts/0
lrwx------. 1 root root 64 Oct  7 08:23 /proc/self/fd/1 -> /dev/pts/0
lrwx------. 1 root root 64 Oct  7 08:23 /proc/self/fd/2 -> /dev/pts/0
```

`null`, `random`, `urandom`, and `zero` Character Devices

Character devices, such as `null`, `random`, `urandom`, and `zero` are examples of pseudo devices that provide access to virtual functionality implemented in software rather than to physical hardware.

`/dev/null` is a data sink. Data that you write to `/dev/null` effectively disappears but the write operation succeeds. Reading from `/dev/null` returns `EOF` (end-of-file).

`/dev/zero` is a data source of an unlimited number of 0-value bytes.

`/dev/random` and `/dev/urandom` are data sources of streams of pseudo random bytes. To maintain high-entropy output, `/dev/random` blocks if its entropy pool doesn't contain sufficient bits of noise. `/dev/urandom` doesn't block and, therefore, the entropy of its output might not be as consistently high as that of `/dev/random`. However, neither `/dev/random` nor `/dev/urandom` are considered to be random enough for the purposes of secure cryptography such as military-grade encryption.

You can find out the size of the entropy pool and the entropy value for `/dev/random` from virtual files under `/proc/sys/kernel/random`:

```
cat /proc/sys/kernel/random/poolsize


4096


cat /proc/sys/kernel/random/entropy_avail


3467
```

For more information, see the `null(4)`, `pts(4)`, and `random(4)` manual pages.

2
Querying udev

You can use the `udevadm` command to query the udev database. For more information, see the `udevadm(8)` manual page.

Prefixes for `udevadm` Information

Outputs from the `udevadm info --query=all` command are prefixed to indicate what the value relates to. Some prefixes are likely to only have a single value, while others might have several values. For example, the device path is singular in value, but there might be several symbolic links and device properties. Outputs for the `udevadm info` command can be restricted to a specific types of information by changing the `--query=<type>`. When the query type is specific, prefixes aren't returned in output.

Table 2-1 udevadm info output prefixes

Prefix	Meaning
P:	Device path in `/sys/`
M:	Device name in `/sys/` (the last component of "P:")
R:	Device number in `/sys/` (the numeric suffix of the last component of "P:")
U:	Kernel subsystem
T:	Kernel device type within subsystem
D:	Kernel device node major/minor
I:	Network interface index
N:	Kernel device node name
L:	Device node symbolic link priority
S:	Device node symbolic link
Q:	Block device sequence number (DISKSEQ)
V:	Attached driver
E:	Device property

View All Information for a Device

- To query the entire information for `/dev/sda`, use the `udevadm info --query=all` command.

```
udevadm info --query=all --name=/dev/sda


P: /devices/pci0000:00/0000:00:04.0/virtio1/host2/target2:0:0/2:0:0:1/
block/sda
M: sda
U: block
```

```
T: disk
D: b 8:0
N: sda
L: 0
S: disk/by-path/pci-0000:00:04.0-scsi-0:0:0:1
S: oracleoci/oraclevda
S: disk/by-id/wwn-0x601666418e094990a94f6e388025315b
S: disk/by-diskseq/1
S: disk/by-id/scsi-3601666418e094990a94f6e388025315b
Q: 1
E: DEVPATH=/devices/pci0000:00/0000:00:04.0/virtio1/host2/
target2:0:0/2:0:0:1/block/sda
E: DEVNAME=/dev/sda
E: DEVTYPE=disk
E: DISKSEQ=1
E: MAJOR=8
E: MINOR=0
E: SUBSYSTEM=block
E: USEC_INITIALIZED=21284275
E: ID_SCSI=1
E: ID_VENDOR=ORACLE
E: ID_VENDOR_ENC=ORACLE\x20\x20
E: ID_MODEL=BlockVolume
E: ID_MODEL_ENC=BlockVolume\x20\x20\x20\x20\x20
E: ID_REVISION=1.0
E: ID_TYPE=disk
E: ID_SERIAL=3601666418e094990a94f6e388025315b
E: ID_SERIAL_SHORT=601666418e094990a94f6e388025315b
E: ID_WWN=0x601666418e094990
E: ID_WWN_VENDOR_EXTENSION=0xa94f6e388025315b
E: ID_WWN_WITH_EXTENSION=0x601666418e094990a94f6e388025315b
E: ID_BUS=scsi
E: ID_PATH=pci-0000:00:04.0-scsi-0:0:0:1
E: ID_PATH_TAG=pci-0000_00_04_0-scsi-0_0_0_1
E: ID_PART_TABLE_UUID=e361a5bb-fab3-4d47-bacd-05f1689be8f0
E: ID_PART_TABLE_TYPE=gpt
E: SCSI_TPGS=0
E: SCSI_TYPE=disk
E: SCSI_VENDOR=ORACLE
E: SCSI_VENDOR_ENC=ORACLE\x20\x20
E: SCSI_MODEL=BlockVolume
E: SCSI_MODEL_ENC=BlockVolume\x20\x20\x20\x20\x20
E: SCSI_REVISION=1.0
E: ID_SCSI_INQUIRY=1
E: SCSI_IDENT_LUN_NAA_REGEXT=601666418e094990a94f6e388025315b
E: DEVLINKS=/dev/disk/by-path/pci-0000:00:04.0-scsi-0:0:0:1 /dev/oracleoci/
oraclevda /dev/disk/by-id/w>
E: TAGS=:systemd:
E: CURRENT_TAGS=:systemd:
```

Limiting Device Information by Query Type

The following examples show how to limit device information by query type.

- Get the relative `sysfs` device path for a device.

To query the `sysfs` device path relative to `/sys` that corresponds to the device file `/dev/sda`:

```
udevadm info --query=path --name=/dev/sda


/devices/pci0000:00/0000:00:0d.0/host0/target0:0:0/0:0:0:0/block/sda
```

- Get all symbolic links for the device.

  To query the symbolic links that point to `/dev/sda`, use the following command:

```
udevadm info --query=symlink --name=/dev/sda


block/8:0
disk/by-id/ata-VBOX_HARDDISK_VB6ad0115d-356e4c09
disk/by-id/scsi-SATA_VBOX_HARDDISK_VB6ad0115d-356e4c09
disk/by-path/pci-0000:00:0d.0-scsi-0:0:0:0
```

- Get the properties of a device.

  To query the properties of `/dev/sda`, use the following command:

```
udevadm info --query=property --name=/dev/sda


DEVPATH=/devices/pci0000:00/0000:00:04.0/virtio1/host2/target2:0:0/2:0:0:1/
block/sda
DEVNAME=/dev/sda
DEVTYPE=disk
DISKSEQ=1
MAJOR=8
MINOR=0
SUBSYSTEM=block
USEC_INITIALIZED=21284275
ID_SCSI=1
ID_VENDOR=ORACLE
ID_VENDOR_ENC=ORACLE\x20\x20
ID_MODEL=BlockVolume
ID_MODEL_ENC=BlockVolume\x20\x20\x20\x20\x20
ID_REVISION=1.0
ID_TYPE=disk
ID_SERIAL=3601666418e094990a94f6e388025315b
ID_SERIAL_SHORT=601666418e094990a94f6e388025315b
ID_WWN=0x601666418e094990
ID_WWN_VENDOR_EXTENSION=0xa94f6e388025315b
ID_WWN_WITH_EXTENSION=0x601666418e094990a94f6e388025315b
ID_BUS=scsi
ID_PATH=pci-0000:00:04.0-scsi-0:0:0:1
ID_PATH_TAG=pci-0000_00_04_0-scsi-0_0_0_1
ID_PART_TABLE_UUID=e361a5bb-fab3-4d47-bacd-05f1689be8f0
ID_PART_TABLE_TYPE=gpt
SCSI_TPGS=0
SCSI_TYPE=disk
SCSI_VENDOR=ORACLE
```

```
SCSI_VENDOR_ENC=ORACLE\x20\x20
SCSI_MODEL=BlockVolume
SCSI_MODEL_ENC=BlockVolume\x20\x20\x20\x20\x20
SCSI_REVISION=1.0
ID_SCSI_INQUIRY=1
SCSI_IDENT_LUN_NAA_REGEXT=601666418e094990a94f6e388025315b
DEVLINKS=/dev/disk/by-id/wwn-0x601666418e094990a94f6e388025315b /dev/
disk/by-diskseq/1 /dev/disk/by-pa>
TAGS=:systemd:
CURRENT_TAGS=:systemd:
```

View Attributes for a Device and Its Parent Devices

To display all the properties of a device and all parent devices that udev finds in `/sys`, use the `--attribute-walk` option.

- Run `udevadm info --attribute-walk` against the device.

  To view information about `/dev/sda` and all parent devices, run:

  ```
  udevadm info --attribute-walk --name=/dev/sda
  ```

  ```
  ...
    looking at device '/devices/pci0000:00/0000:00:04.0/virtio1/host2/
  target2:0:0/2:0:0:1/block/sda':
      KERNEL=="sda"
      SUBSYSTEM=="block"
      DRIVER==""
      ATTR{alignment_offset}=="0"
      ATTR{capability}=="40"
      ATTR{discard_alignment}=="0"
      ATTR{diskseq}=="1"
      ATTR{events}==""
      ATTR{events_async}==""
      ATTR{events_poll_msecs}=="-1"
      ATTR{ext_range}=="256"
      ATTR{hidden}=="0"
      ATTR{inflight}=="       0        0"
      ATTR{integrity/device_is_integrity_capable}=="0"
      ATTR{integrity/format}=="none"
      ATTR{integrity/protection_interval_bytes}=="0"
      ATTR{integrity/read_verify}=="0"
      ATTR{integrity/tag_size}=="0"
      ATTR{integrity/write_generate}=="0"
  ...
      ATTR{trace/enable}=="0"
      ATTR{trace/end_lba}=="disabled"
      ATTR{trace/pid}=="disabled"
      ATTR{trace/start_lba}=="disabled"

    looking at parent device '/devices/pci0000:00/0000:00:04.0/virtio1/host2/
  target2:0:0/2:0:0:1':
      KERNELS=="2:0:0:1"
      SUBSYSTEMS=="scsi"
  ```

```
    DRIVERS=="sd"
    ATTRS{blacklist}=="TRY_VPD_PAGES"
    ATTRS{delete}=="(not readable)"
    ATTRS{device_blocked}=="0"
    ATTRS{device_busy}=="0"
    ATTRS{dh_state}=="detached"
    ATTRS{eh_timeout}=="10"
    ATTRS{evt_capacity_change_reported}=="0"
    ATTRS{evt_inquiry_change_reported}=="0"
    ATTRS{evt_lun_change_reported}=="0"
    ATTRS{evt_media_change}=="0"
    ATTRS{evt_mode_parameter_change_reported}=="0"
    ATTRS{evt_soft_threshold_reached}=="0"
    ATTRS{inquiry}==""
    ATTRS{iocounterbits}=="32"
    ATTRS{iodone_cnt}=="0x34ad3b"
    ATTRS{ioerr_cnt}=="0x2"
    ATTRS{iorequest_cnt}=="0x34ad3b"
    ATTRS{model}=="BlockVolume     "
...
    ATTRS{vpd_pg83}==""
    ATTRS{wwid}=="naa.601666418e094990a94f6e388025315b"

  looking at parent device '/devices/pci0000:00/0000:00:04.0/virtio1/host2/
target2:0:0':
    KERNELS=="target2:0:0"
    SUBSYSTEMS=="scsi"
    DRIVERS==""
    ATTRS{power/control}=="auto"
    ATTRS{power/runtime_active_time}=="2172559026"
    ATTRS{power/runtime_status}=="active"
    ATTRS{power/runtime_suspended_time}=="0"

  looking at parent device '/devices/pci0000:00/0000:00:04.0/virtio1/
host2':
    KERNELS=="host2"
    SUBSYSTEMS=="scsi"
    DRIVERS==""
    ATTRS{power/control}=="auto"
    ATTRS{power/runtime_active_time}=="2172559046"
    ATTRS{power/runtime_status}=="active"
    ATTRS{power/runtime_suspended_time}=="0"

  looking at parent device '/devices/pci0000:00/0000:00:04.0/virtio1':
    KERNELS=="virtio1"
    SUBSYSTEMS=="virtio"
    DRIVERS=="virtio_scsi"
    ATTRS{device}=="0x0008"

ATTRS{features}=="01100000000000000000000000001100100000000000000000000000000
0000000"
    ATTRS{power/control}=="auto"
    ATTRS{power/runtime_active_time}=="0"
    ATTRS{power/runtime_status}=="unsupported"
    ATTRS{power/runtime_suspended_time}=="0"
    ATTRS{status}=="0x0000000f"
```

```
    ATTRS{vendor}=="0x108e"

  looking at parent device '/devices/pci0000:00/0000:00:04.0':
    KERNELS=="0000:00:04.0"
    SUBSYSTEMS=="pci"
    DRIVERS=="virtio-pci"
    ATTRS{ari_enabled}=="0"
...
    ATTRS{revision}=="0x00"
    ATTRS{subsystem_device}=="0x0008"
    ATTRS{subsystem_vendor}=="0x108e"
    ATTRS{vendor}=="0x1af4"

  looking at parent device '/devices/pci0000:00':
    KERNELS=="pci0000:00"
    SUBSYSTEMS==""
    DRIVERS==""
    ATTRS{power/control}=="auto"
    ATTRS{power/runtime_active_time}=="0"
    ATTRS{power/runtime_status}=="unsupported"
    ATTRS{power/runtime_suspended_time}=="0"
    ATTRS{waiting_for_supplier}=="0"
```

The command starts at the device that's specified by the device path and walks the chain of parent devices. For every device that the command finds, the command displays the possible attributes for the device and its parent devices by using the match key format for udev rules.

3
Working With udev Rules

udev uses rules files to identify devices and create device names. The `udev` service (`systemd-udevd`) reads the rules files at system start-up and stores the rules in memory. If the kernel discovers a new device or an existing device goes offline, the kernel sends an event action (*uevent*) notification to `udev`, which matches the in-memory rules against the device attributes in the `/sys` directory to identify the device.

Rules files exist in several different directories. However, you only need to know about `/etc/udev/rules.d/*.rules` files because these are the only rules files that you can edit. See Customizing udev Rules.

udev processes the rules files in lexical order, regardless of the directory of the rule files. Rules files in `/etc/udev/rules.d` override rules files of the same name in other locations.

The following rules are extracted from the file `/lib/udev/rules.d/50-udev-default.rules` and illustrate the syntax of udev rules:

```
# do not edit this file, it will be overwritten on update

SUBSYSTEM=="block", SYMLINK{unique}+="block/%M:%m"
SUBSYSTEM!="block", SYMLINK{unique}+="char/%M:%m"

KERNEL=="pty[pqrstuvwxyzabcdef][0123456789abcdef]", GROUP="tty", MODE="0660"
KERNEL=="tty[pqrstuvwxyzabcdef][0123456789abcdef]", GROUP="tty", MODE="0660"
...
# mem
KERNEL=="null|zero|full|random|urandom", MODE="0666"
KERNEL=="mem|kmem|port|nvram",  GROUP="kmem", MODE="0640"
...
# block
SUBSYSTEM=="block", GROUP="disk"
...
# network
KERNEL=="tun",                  MODE="0666"
KERNEL=="rfkill",               MODE="0644"

# CPU
KERNEL=="cpu[0-9]*",            MODE="0444"
...
# do not delete static device nodes
ACTION=="remove", NAME=="", TEST=="/lib/udev/devices/%k", \
    OPTIONS+="ignore_remove"
ACTION=="remove", NAME=="?*", TEST=="/lib/udev/devices/$name", \
    OPTIONS+="ignore_remove"
```

For more information, see the `udev(7)` manual page.

Assignment and Comparison Operators

A rule either assigns a value to a key or it tries to find a match for a key by comparing its current value with the specified value. The following table shows the assignment and comparison operators that you can use.

Operator	Description
`=`	Assign a value to a key, overwriting any previous value.
`+=`	Assign a value by appending it to the key's current list of values.
`:=`	Assign a value to a key. This value cannot be changed by any further rules.
`==`	Match the key's current value against the specified value for equality.
`!=`	Match the key's current value against the specified value for inequality.

Pattern-Matching Characters

You can use the following shell-style pattern-matching characters in values.

Character	Description
`?`	Matches a single character.
`*`	Matches any number of characters, including zero.
`[]`	Matches any single character or character from a range of characters specified within the brackets. For example, `tty[sS][0-9]` would match `ttys7` or `ttyS7`.

Common Match Keys

The following table describes commonly used match keys in rules.

Match Key	Description
`ACTION`	Matches the name of the action that led to an event. For example, `ACTION=="add"` or `ACTION=="remove"`.
`ENV{`*key*`}`	Matches a value for the device property *key*. For example, `ENV{DEVTYPE}=="disk"`.
`KERNEL`	Matches the name of the device that's affected by an event. For example, `KERNEL=="dm-*"` for disk media.
`NAME`	Matches the name of a device file or network interface. For example, `NAME=="?*"` for any name that consists of one or more characters.

Match Key	Description
SUBSYSTEM	Matches the subsystem of the device that's affected by an event. For example, SUBSYSTEM=="tty".
TEST	Tests wheter the specified file or path exists; for example, TEST=="/lib/udev/devices/$name", where $name is the name of the matched device file.

Other match keys include ATTR{*filename*}, ATTRS{*filename*}, DEVPATH, DRIVER, DRIVERS, KERNELS, PROGRAM, RESULT, SUBSYSTEMS, and SYMLINK.

Common Assignment Keys

The following table describes commonly used assignment keys in rules.

Assignment Key	Description
ENV{*key*}	Specifies a value for the device property *key*, such as GROUP="disk".
GROUP	Specifies the group for a device file, such as GROUP="disk".

Assignment Key	Description
IMPORT{*type*}	Specifies a set of variables for the device property, depending on *type*: **cmdline** Import a single property from the boot `kernel` command line. For simple flags, `udev` sets the value of the property to 1. For example, `IMPORT{cmdline}="nodmraid"`. **db** Interpret the specified value as an index into the device database and import a single property, which must have already been set by an earlier event. For example, `IMPORT{db}="DM_UDEV_LOW_PRIORITY_FLAG"`. **file** Interpret the specified value as the name of a text file and import its contents, which must be in environmental key format. For example, `IMPORT{file}="keyfile"`. **parent** Interpret the specified value as a key-name filter and import the stored keys from the database entry for the parent device. For example `IMPORT{parent}="ID_*"`. **program** Run the specified value as an external program and imports its result, which must be in environmental key format. For example `IMPORT{program}="usb_id --export %p"`.
MODE	Specifies the permissions for a device file, such as `MODE="0640"`.
NAME	Specifies the name of a device file, such as `NAME="em1"`.
OPTIONS	Specifies rule and device options, such as `OPTIONS+="ignore_remove"`, which means that the device file isn't removed if the device is removed.
OWNER	Specifies the owner for a device file, such as `GROUP="root"`.
RUN	Specifies a command to be run after the device file has been created, such as `RUN+="/usr/bin/eject $kernel"`, where `$kernel` is the kernel name of the device.
SYMLINK	Specifies the name of a symbolic link to a device file, such as `SYMLINK+="disk/by-uuid/$env{ID_FS_UUID_ENC}"`, where `$env{}` is substituted with the specified device property.

Other assignment keys include `ATTR{key}`, `GOTO`, `LABEL`, `RUN`, and `WAIT_FOR`.

String Substitutions

The following table describes the string substitutions that are commonly used with the `GROUP`, `MODE`, `NAME`, `OWNER`, `PROGRAM`, `RUN`, and `SYMLINK` keys.

String Substitution	Description
`$attr{file}` or `%s{file}`	Specifies the value of a device attribute from a file under `/sys`, such as `ENV{MATCHADDR}="$attr{address}"`.
`$devpath` or `%p`	The device path of the device in the `sysfs` file system under `/sys`, such as `RUN+="keyboard-force-release.sh $devpath common-volume-keys"`.
`$env{key}` or `%E{key}`	Specifies the value of a device property, such as `SYMLINK+="disk/by-id/md-name-$env{MD_NAME}-part%n"`.
`$kernel` or `%k`	Specifies the kernel name for the device.
`$major` or `%M`	Specifies the major number of a device, such as `IMPORT{program}="udisks-dm-export %M %m"`.
`$minor` or `%m`	Specifies the minor number of a device, such as `RUN+="$env{LVM_SBIN_PATH}/lvm pvscan --cache --major $major --minor $minor"`.
`$name`	Specifies the device file of the current device, such as `TEST=="/lib/udev/devices/$name"`.

udev expands the strings specified for `RUN` immediately before its program is run, which is after udev has finished processing all other rules for the device. For the other keys, `udev` expands the strings while it's processing the rules.

4
Customizing udev Rules

The order in which rules are evaluated is important. udev processes rules in lexical order. To add custom rules, you need `udev` to find and evaluate these rules before the default rules.

The following example procedure shows how to implement a `udev` rules file that adds a symbolic link to the disk device `/dev/sdb`.

1. Create the rule file in `/etc/udev/rules.d`.

   Create a rule file under `/etc/udev/rules.d` with a file name such as `10-local.rules` that udev reads before any other rules file.

   The following rule in `10-local.rules` creates the symbolic link `/dev/my_disk`, which points to `/dev/sdb`:

   ```
   SUBSYSTEM=="block", KERNEL=="sdb", SYMLINK+="my_disk"
   ```

   Listing the device files in `/dev` shows that `udev` hasn't yet applied the rule:

   ```
   ls /dev/sd* /dev/my_disk
   ```

   ```
   ls: cannot access /dev/my_disk: No such file or directory
   /dev/sda  /dev/sda1  /dev/sda2  /dev/sdb
   ```

2. Test the new rule by using the `udevadm test` command.

   To simulate how `udev` applies its rules to create a device, you can use the `udevadm test` command with the device path of `sdb` listed under the `/sys/class/block` hierarchy, for example:

   ```
   udevadm test /sys/class/block/sdb
   ```

   ```
   calling: test
   version ...
   This program is for debugging only, it does not run any program
   specified by a RUN key. It may show incorrect results, because
   some values may be different, or not available at a simulation run.
   ...
   sdb: /etc/udev/rules.d/10-local.rules:1 SYMLINK+="my_disk": Added device
   node symlink "my_disk".
   ...
   creating link '/dev/my_disk' to '/dev/sdb'
   creating symlink '/dev/my_disk' to 'sdb
   ...
   ACTION=add
   SUBSYSTEM=block
   DEVLINKS=/dev/disk/by-id/ata-VBOX_HARDDISK_VB186e4ce2-f80f170d
     /dev/disk/by-uuid/a7dc508d-5bcc-4112-b96e-f40b19e369fe
   ```

```
  /dev/my_disk
...
```

3. Restart the `systemd-udevd` service.

   ```
   sudo systemctl restart systemd-udevd
   ```

4. Trigger a block event for the device that's affected by the new rule:

   ```
   sudo udevadm trigger /sys/class/block/sdb
   ```

5. Verify that the rule is active.

   After `udev` processes the rules files, the symbolic link `/dev/my_disk` is added:

   ```
   ls -F /dev/sd* /dev/my_disk


   /dev/my_disk@  /dev/sda  /dev/sda1  /dev/sda2  /dev/sdb
   ```

6. (Optional) Undo the changes so that the rule and the symbolic link are removed.

   To undo the changes, remove `/etc/udev/rules.d/10-local.rules` and `/dev/my_disk`, then run `systemctl restart systemd-udevd` again.

   ```
   sudo rm /etc/udev/rules.d/10-local.rules
   sudo rm /dev/my_disk
   sudo systemctl restart systemd-udevd
   ```