Rocky Enterprise Linux 9.2 Manual Pages on command 'udev.7'

$ man udev.7

UDEV(7) udev UDEV(7)

NAME

udev - Dynamic device management

DESCRIPTION

udev supplies the system software with device events, manages permissions of device nodes and may create additional symlinks in the /dev/ directory, or renames network interfaces. The kernel usually just assigns unpredictable device names based on the order of discovery. Meaningful symlinks or network device names provide a way to reliably identify devices based on their properties or current configuration.

The udev daemon, systemd-udevd.service(8), receives device uevents directly from the kernel whenever a device is added or removed from the system, or it changes its state. When udev receives a device event, it matches its configured set of rules against various device attributes to identify the device. Rules that match may provide additional device information to be stored in the udev database or to be used to create meaningful symlink names.

All device information udev processes is stored in the udev database

and sent out to possible event subscribers. Access to all stored data and the event sources is provided by the library libudev.

RULES FILES

The udev rules are read from the files located in the system rules directories /usr/lib/udev/rules.d and /usr/local/lib/udev/rules.d, the volatile runtime directory /run/udev/rules.d and the local administration directory /etc/udev/rules.d. All rules files are collectively sorted and processed in lexical order, regardless of the directories in which they live. However, files with identical filenames replace each other. Files in /etc/ have the highest priority, files in /run/ take precedence over files with the same name under /usr/. This can be used to override a system-supplied rules file with a local file if needed; a symlink in /etc/ with the same name as a rules file in /usr/lib/, pointing to /dev/null, disables the rules file entirely.

Rule files must have the extension .rules; other extensions are ignored.

Every line in the rules file contains at least one key-value pair. Except for empty lines or lines beginning with "#", which are ignored.

There are two kinds of keys: match and assignment. If all match keys match against their values, the rule gets applied and the assignment keys get the specified values assigned.

A matching rule may rename a network interface, add symlinks pointing to the device node, or run a specified program as part of the event handling.

A rule consists of a comma-separated list of one or more key-operator-value expressions. Each expression has a distinct effect, depending on the key and operator used.

Operators

"=="

Compare for equality. (The specified key has the specified value.)

"!="

Compare for inequality. (The specified key doesn't have the specified value, or the specified key is not present at all.)

"="

Assign a value to a key. Keys that represent a list are reset and only this single value is assigned.

"+="

Add the value to a key that holds a list of entries.

"-="

Remove the value from a key that holds a list of entries.

":="

Assign a value to a key finally; disallow any later changes.

Values

Values are written as double quoted strings, such as ("string"). To include a quotation mark (") in the value, precede it by a backslash (\"). Any other occurrences of a backslash followed by a character are not unescaped. That is, "\t\n" is treated as four characters: backslash, lowercase t, backslash, lowercase n.

The string can be prefixed with a lowercase e (e"string\n") to mark the string as C-style escaped[1]. For example, e"string\n" is parsed as 7 characters: 6 lowercase letters and a newline. This can be useful for writing special characters when a kernel driver requires them.

Please note that NUL is not allowed in either string variant.

Keys

The following key names can be used to match against device properties. Some of the keys also match against properties of the parent devices in sysfs, not only the device that has generated the event. If multiple keys that match a parent device are specified in a single rule, all these keys must match at one and the same parent device.

ACTION

Match the name of the event action.

DEVPATH

Match the devpath of the event device.

KERNEL

Match the name of the event device.

KERNELS

Search the devpath upwards for a matching device name.

NAME

Match the name of a network interface. It can be used once the NAME key has been set in one of the preceding rules.

SYMLINK

Match the name of a symlink targeting the node. It can be used once a SYMLINK key has been set in one of the preceding rules. There may be multiple symlinks; only one needs to match.

SUBSYSTEM

Match the subsystem of the event device.

SUBSYSTEMS

Search the devpath upwards for a matching device subsystem name.

DRIVER

Match the driver name of the event device. Only set this key for devices which are bound to a driver at the time the event is generated.

DRIVERS

Search the devpath upwards for a matching device driver name.

ATTR{filename}

Match sysfs attribute value of the event device.

Trailing whitespace in the attribute values is ignored unless the specified match value itself contains trailing whitespace.

ATTRS{filename}

Search the devpath upwards for a device with matching sysfs attribute values. If multiple ATTRS matches are specified, all of them must match on the same device.

Trailing whitespace in the attribute values is ignored unless the specified match value itself contains trailing whitespace.

SYSCTL{kernel parameter}

Match a kernel parameter value.

ENV{key}

Match against a device property value.

CONST{key}

Match against a system-wide constant. Supported keys are:

"arch"

System's architecture. See ConditionArchitecture= in systemd.unit(5) for possible values.

"virt"

System's virtualization environment. See systemd-detect-virt(1) for possible values.

Unknown keys will never match.

TAG

Match against a device tag.

TAGS

Search the devpath upwards for a device with matching tag.

TEST{octal mode mask}

Test the existence of a file. An octal mode mask can be specified if needed.

PROGRAM

Execute a program to determine whether there is a match; the key is true if the program returns successfully. The device properties are made available to the executed program in the environment. The program's standard output is available in the RESULT key.

This can only be used for very short-running foreground tasks. For details, see RUN.

Note that multiple PROGRAM keys may be specified in one rule, and "=", ":=", and "+=" have the same effect as "==".

RESULT

Match the returned string of the last PROGRAM call. This key can be used in the same or in any later rule after a PROGRAM call.

Most of the fields support shell glob pattern matching and alternate patterns. The following special characters are supported:

"*"

Matches zero or more characters.

"?"

Matches any single character.

"[]"

Matches any single character specified within the brackets. For example, the pattern string "tty[SR]" would match either "ttyS" or "ttyR". Ranges are also supported via the "-" character. For example, to match on the range of all digits, the pattern "[0-9]" could be used. If the first character following the "[" is a "!", any characters not enclosed are matched.

"|"

Separates alternative patterns. For example, the pattern string "abc|x*" would match either "abc" or "x*".

The following keys can get values assigned:

NAME

The name to use for a network interface. See systemd.link(5) for a higher-level mechanism for setting the interface name. The name of a device node cannot be changed by udev, only additional symlinks can be created.

SYMLINK

The name of a symlink targeting the node. Every matching rule adds this value to the list of symlinks to be created.

The set of characters to name a symlink is limited. Allowed characters are "0-9A-Za-z#+-.:=@_/", valid UTF-8 character sequences, and "\x00" hex encoding. All other characters are replaced by a "_" character.

Multiple symlinks may be specified by separating the names by the space character. In case multiple devices claim the same name, the link always points to the device with the highest link_priority. If the current device goes away, the links are re-evaluated and the device with the next highest link_priority becomes the owner of the link. If no link_priority is specified, the order of the devices (and which one of them owns the link) is undefined.

Symlink names must never conflict with the kernel's default device node names, as that would result in unpredictable behavior.

OWNER, GROUP, MODE

The permissions for the device node. Every specified value overrides the compiled-in default value.

SECLABEL{module}

Applies the specified Linux Security Module label to the device node.

ATTR{key}

The value that should be written to a sysfs attribute of the event device.

SYSCTL{kernel parameter}

The value that should be written to kernel parameter.

ENV{key}

Set a device property value. Property names with a leading "." are neither stored in the database nor exported to events or external tools (run by, for example, the PROGRAM match key).

TAG

Attach a tag to a device. This is used to filter events for users of libudev's monitor functionality, or to enumerate a group of tagged devices. The implementation can only work efficiently if only a few tags are attached to a device. It is only meant to be used in contexts with specific device filter requirements, and not as a general-purpose flag. Excessive use might result in inefficient event handling.

RUN{type}

Specify a program to be executed after processing of all the rules for the event. With "+=", this invocation is added to the list, and with "=" or ":=", it replaces any previous contents of the list.
Please note that both "program" and "builtin" types described below share a common list, so clearing the list with ":=" and "=" affects both types.
type may be:
"program"

Execute an external program specified as the assigned value. If no absolute path is given, the program is expected to live in

/usr/lib/udev; otherwise, the absolute path must be specified.

This is the default if no type is specified.

"builtin"

As program, but use one of the built-in programs rather than an external one.

The program name and following arguments are separated by spaces. Single quotes can be used to specify arguments with spaces.

This can only be used for very short-running foreground tasks. Running an event process for a long period of time may block all further events for this or a dependent device.

Note that running programs that access the network or mount/unmount filesystems is not allowed inside of udev rules, due to the default sandbox that is enforced on systemd-udevd.service.

Starting daemons or other long-running processes is not allowed; the forked processes, detached or not, will be unconditionally killed after the event handling has finished. In order to activate long-running processes from udev rules, provide a service unit and pull it in from a udev device using the SYSTEMD_WANTS device property. See systemd.device(5) for details.

LABEL

A named label to which a GOTO may jump.

GOTO

Jumps to the next LABEL with a matching name.

IMPORT{type}

Import a set of variables as device properties, depending on type:

"program"

Execute an external program specified as the assigned value and, if it returns successfully, import its output, which must be in environment key format. Path specification, command/argument separation, and quoting work like in RUN.

"builtin"

Similar to "program", but use one of the built-in programs rather than an external one.

"file"

Import a text file specified as the assigned value, the content of which must be in environment key format.

"db"

Import a single property specified as the assigned value from the current device database. This works only if the database is already populated by an earlier event.

"cmdline"

Import a single property from the kernel command line. For simple flags the value of the property is set to "1".

"parent"

Import the stored keys from the parent device by reading the database entry of the parent device. The value assigned to IMPORT{parent} is used as a filter of key names to import (with the same shell glob pattern matching used for comparisons).

This can only be used for very short-running foreground tasks. For details see RUN.

Note that multiple IMPORT{} keys may be specified in one rule, and "=", ":=", and "+=" have the same effect as "==". The key is true if the import is successful, unless "!=" is used as the operator which causes the key to be true if the import failed.

WAIT_FOR

Wait for a file to become available or until a timeout of 10 seconds expires. The path is relative to the sysfs device; if no path is specified, this waits for an attribute to appear.

OPTIONS

Rule and device options:

link_priority=value

Specify the priority of the created symlinks. Devices with higher priorities overwrite existing symlinks of other devices. The default is 0.

string_escape=none|replace

When "replace", possibly unsafe characters in strings assigned

to NAME, SYMLINK, and ENV{key} are replaced. When "none", no replacement is performed. When unset, the replacement is performed for NAME, SYMLINK, but not for ENV{key}. Defaults to unset.

static_node=

Apply the permissions specified in this rule to the static device node with the specified name. Also, for every tag specified in this rule, create a symlink in the directory /run/udev/static_node-tags/tag pointing at the static device node with the specified name. Static device node creation is performed by systemd-tmpfiles before systemd-udevd is started. The static nodes might not have a corresponding kernel device; they are used to trigger automatic kernel module loading when they are accessed.

watch

Watch the device node with inotify; when the node is closed after being opened for writing, a change uevent is synthesized.

nowatch

Disable the watching of a device node with inotify.

db_persist

Set the flag (sticky bit) on the udev database entry of the event device. Device properties are then kept in the database even when udevadm info --cleanup-db is called. This option can be useful in certain cases (e.g. Device Mapper devices) for persisting device state on the transition from initrd.

log_level=level

Takes a log level name like "debug" or "info", or a special value "reset". When a log level name is specified, the maximum log level is changed to that level. When "reset" is set, then the previously specified log level is revoked. Defaults to the log level of the main process of systemd-udevd.

This may be useful when debugging events for certain devices.

Note that the log level is applied when the line including this

rule is processed. So, for debugging, it is recommended that this is specified at earlier place, e.g., the first line of 00-debug.rules.

Example for debugging uevent processing for network interfaces:

```
# /etc/udev/rules.d/00-debug-net.rules
SUBSYSTEM=="net", OPTIONS="log_level=debug"
```

The NAME, SYMLINK, PROGRAM, OWNER, GROUP, MODE, SECLABEL, and RUN fields support simple string substitutions. The RUN substitutions are performed after all rules have been processed, right before the program is executed, allowing for the use of device properties set by earlier matching rules. For all other fields, substitutions are performed while the individual rule is being processed. The available substitutions are:

$kernel, %k

The kernel name for this device.

$number, %n

The kernel number for this device. For example, "sda3" has kernel number 3.

$devpath, %p

The devpath of the device.

$id, %b

The name of the device matched while searching the devpath upwards for SUBSYSTEMS, KERNELS, DRIVERS, and ATTRS.

$driver

The driver name of the device matched while searching the devpath upwards for SUBSYSTEMS, KERNELS, DRIVERS, and ATTRS.

$attr{file}, %s{file}

The value of a sysfs attribute found at the device where all keys of the rule have matched. If the matching device does not have such an attribute, and a previous KERNELS, SUBSYSTEMS, DRIVERS, or ATTRS test selected a parent device, then the attribute from that parent device is used.

If the attribute is a symlink, the last element of the symlink

target is returned as the value.

$env{key}, %E{key}

A device property value.

$major, %M

The kernel major number for the device.

$minor, %m

The kernel minor number for the device.

$result, %c

The string returned by the external program requested with PROGRAM.

A single part of the string, separated by a space character, may be

selected by specifying the part number as an attribute: "%c{N}". If

the number is followed by the "+" character, this part plus all

remaining parts of the result string are substituted: "%c{N+}".

$parent, %P

The node name of the parent device.

$name

The current name of the device. If not changed by a rule, it is the

name of the kernel device.

$links

A space-separated list of the current symlinks. The value is only

set during a remove event or if an earlier rule assigned a value.

$root, %r

The udev_root value.

$sys, %S

The sysfs mount point.

$devnode, %N

The name of the device node.

%%

The "%" character itself.

$$

The "$" character itself.

SEE ALSO

systemd-udevd.service(8), udevadm(8), systemd.link(5)

NOTES

1. C-style escaped

   https://en.wikipedia.org/wiki/Escape_sequences_in_C#Table_of_escape_sequences