NAME

udev − dynamic device management

DESCRIPTION

udev provides a dynamic device directory containing only the files for actually present devices. It creates or removes device node files in the /dev directory, or it renames network interfaces.

Usually udev runs as **udevd**(8) and receives uevents directly from the kernel if a device is added or removed from the system.

If udev receives a device event, it matches its configured rules against the available device attributes provided in sysfs to identify the device. Rules that match may provide additional device information or specify a device node name and multiple symlink names and instruct udev to run additional programs as part of the device event handling.

CONFIGURATION

udev configuration files are placed in /etc/udev/ and /lib/udev/. All empty lines, or lines beginning with ´#´ will be ignored.

Configuration file

udev expects its main configuration file at /etc/udev/udev.conf. It consists of a set of variables allowing the user to override default udev values. The following variables can be set:

udev_root
Specifies where to place the device nodes in the filesystem. The default value is /dev.

udev_log
The logging priority. Valid values are the numerical syslog priorities or their textual representations: **err**, **info** and **debug**.

Rules files

The udev rules are read from the files located in the default rules directory /lib/udev/rules.d/, the custom rules directory /etc/udev/rules.d/ and the temporary rules directory /dev/.udev/rules.d/. All rule files are sorted and processed in lexical order, regardless in which of these directories they live.

Rule files are required to have a unique name, duplicate file names are ignored. Files in /etc/udev/rules.d/ have precedence over files with the same name in /lib/udev/rules.d/. This can be used to ignore a default rules file if needed.

Every line in the rules file contains at least one key value pair. There are two kind of keys, match and assignment keys. If all match keys are matching against its value, the rule gets applied and the assign keys get the specified value assigned.

A matching rule may specify the name of the device node, add a symlink pointing to the node, or run a specified program as part of the event handling. If no matching rule is found, the default device node name is used.

A rule consists of a list of one or more key value pairs separated by a comma. Each key has a distinct operation, depending on the used operator. Valid operators are:

==
Compare for equality.

!=
Compare for inequality.

=
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.

:=
Assign a value to a key finally; disallow any later changes, which may be used to prevent changes by any later rules.

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.

NAME
Match the name of the node or 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.

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

ATTR{*filename***}**
Match sysfs attribute values of the event device. Trailing whitespace in the attribute values is ignored, if the specified match value does not contain trailing whitespace itself.

KERNELS
Search the devpath upwards for a matching device name.

SUBSYSTEMS
Search the devpath upwards for a matching device subsystem name.

DRIVERS
Search the devpath upwards for a matching device driver name.

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, if the specified match value does not contain trailing whitespace itself.

ENV{*key***}**
Match against a device property value.

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

PROGRAM
Execute a program. 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 output printed to stdout, is available in the RESULT key.

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 a shell style pattern matching. The following pattern characters are supported:

*****

Matches zero, or any number of 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 within this match with the ´−´ character. For example, to match on the range of all digits, the pattern [0−9] would be used. If the first character following the ´[´ is a ´!´, any characters not enclosed are matched.

The following keys can get values assigned:

NAME
The name of the node to be created, or the name the network interface should be renamed to.

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 utf8 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 kernels default device node names, as that would result in unpredictable behavior.

OWNER, GROUP, MODE
The permissions for the device node. Every specified value overwrites the compiled−in default value.

ATTR{*key***}**
The value that should be written to a sysfs attribute of the event device.

ENV{*key***}**
Set a device property value. Property names with a leading ´.´ are not stored in the database or exported to external tool or events.

RUN
Add a program to the list of programs to be executed for a specific device. This can only be used for very short running tasks. Running an event process for a long period of time may block all further events for this or a dependent device. Long running tasks need to be immediately detached from the event process itself. If the option **RUN{***fail_event_on_error***}** is specified, and the executed program returns non−zero, the event will be marked as failed for a possible later handling.

If the specified string starts with **socket:***path*, all current event values will be passed to the specified socket, as a message in the same format the kernel sends an uevent. If the first character of the specified path is an @ character, an abstract namespace socket is used, instead of an existing socket file.

LABEL
Named label where a GOTO can jump to.

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 import its output, which must be in environment key format.

file
Import a text file specified as the assigned value, 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 commandline. For simple flags the value of the property will be 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−style pattern matching used for comparisons).

If no option is given, udev will choose between **program** and **file** based on the executable bit of the file permissions.

WAIT_FOR
Wait for a file to become available.

OPTIONS
Rule and device options:

ignore_device
Ignore this event completely.

ignore_remove
Do not remove the device node when the device goes away. This may be useful as a workaround for broken device drivers.

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

all_partitions
Create the device nodes for all available partitions of a block device. This may be useful for removable media devices where media changes are not detected.

event_timeout=
Number of seconds an event will wait for operations to finish, before it will terminate itself.

string_escape=*none*|*replace*
Usually control and other possibly unsafe characters are replaced in strings used for device naming. The mode of replacement can be specified with this option.

watch
Watch the device node with inotify, when closed after being opened for writing, a change uevent will be synthesised.

nowatch
Disable the watching of a device node with inotify.

The **NAME**, **SYMLINK**, **PROGRAM**, **OWNER**, **GROUP**, **MODE** and **RUN** fields support simple printf−like string substitutions. The **RUN** format chars gets applied after all rules have been processed, right before the program is executed. It allows the use of device properties set by earlier matching rules. For all other fields, substitutions are applied 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 of ´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, follow the chain of parent devices and use the value of the first attribute that matches. 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 ´+´ char 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 node. If not changed by a rule, it is the name of the kernel device.

$links
The current list of symlinks, separated by a space character. The value is only set if an earlier rule assigned a value, or during a remove events.

$root, **%r**
The udev_root value.

$sys, **%S**
The sysfs mount point.

$tempnode, **%N**
The name of a created temporary device node to provide access to the device from a external program before the real node is created.

%%
The ´%´ character itself.

$$
The ´$´ character itself.

AUTHOR

Written by Greg Kroah−Hartman greg@kroah.com and Kay Sievers kay.sievers@vrfy.org. With much help from Dan Stekloff and many others.

SEE ALSO

udevd(8), **udevadm**(8)