NAME

devfs - DEVFS control

SYNOPSIS

devfs [**-m** *mount-point*] *keyword argument* ...

DESCRIPTION

The **devfs** utility provides an interface to manipulate properties of devfs(5) mounts.

The rules, by default as configured by */etc/rc.conf*, are loaded at boot via the devfs service(8). The rules can be reloaded by running the command:

```
service devfs restart
```

The *keyword* argument determines the context for the rest of the arguments. For example, most of the commands related to the rule subsystem must be preceded by the **rule** keyword. The following flags are common to all keywords:

-m *mount-point* Operate on *mount-point*, which is expected to be a devfs(5) mount. If this option is not specified, **devfs** operates on */dev*.

Rule Subsystem

The devfs(5) rule subsystem provides a way for the administrator of a system to control the attributes of DEVFS nodes. Each DEVFS mount-point has a "ruleset", or a list of rules, associated with it. When a device driver creates a new node, all the rules in the ruleset associated with each mount-point are applied (see below) before the node becomes visible to the userland. This permits the administrator to change the properties, including the visibility, of certain nodes. For example, one might want to hide all disk nodes in a jail(2)'s */dev*.

Rule Manipulation

Rule manipulation commands follow the **rule** keyword. The following flags are common to all of the rule manipulation commands:

-s *ruleset* Operate on the ruleset with the number *ruleset*. If this is not specified, the commands operate on the ruleset currently associated with the specified mount-point.

The following commands are recognized:

rule add [*rulenum*] *rulespec*

Add the rule described by *rulespec* (defined below) to the ruleset. The rule has the number *rulenum* if it is explicitly specified; otherwise, the rule number is automatically determined by the kernel.

rule apply *rulenum* | *rulespec*
Apply rule number *rulenum* or the rule described by *rulespec* to the mount-point. Rules that are "applied" have their conditions checked against all nodes in the mount-point and the actions taken if they match.

rule applyset Apply all the rules in the ruleset to the mount-point (see above for the definition of "apply").

rule del *rulenum* Delete rule number *rulenum* from the ruleset.

rule delset Delete all rules from the ruleset.

rule show [*rulenum*]
Display the rule number *rulenum*, or all the rules in the ruleset. The output lines (one line per rule) are expected to be valid *rulespec*s.

rule showsets Report the numbers of existing rulesets.

ruleset *ruleset* Set ruleset number *ruleset* as the current ruleset for the mount-point.

Rule Specification

Rules have two parts: the conditions and the actions. The conditions determine which DEVFS nodes the rule matches and the actions determine what should be done when a rule matches a node. For example, a rule can be written that sets the GID to "operator" for all devices of type tape. If the first token of a rule specification is a single dash ('-'), rules are read from the standard input and the rest of the specification is ignored.

The following conditions are recognized. Conditions are ANDed together when matching a device; if OR is desired, multiple rules can be written.

path *pattern* Matches any node with a path that matches *pattern*, which is interpreted as a glob(3)-style pattern.

type *devtype* Matches any node that is of type *devtype*. Valid types are **disk**, **mem**, **tape** and **tty**.

The following actions are recognized. Although there is no explicit delimiter between conditions and

actions, they may not be intermixed.

group *gid* Set the GID of the node to *gid*, which may be a group name (looked up in */etc/group*) or number.

hide Hide the node. Nodes may later be revived manually with mknod(8) or with the **unhide** action. Hiding a directory node effectively hides all of its child nodes.

include *ruleset* Apply all the rules in ruleset number *ruleset* to the node. This does not necessarily result in any changes to the node (e.g., if none of the rules in the included ruleset match). Include commands in the referenced *ruleset* are not resolved.

mode *filemode* Set the file mode to *filemode*, which is interpreted as in chmod(1).

user *uid* Set the UID to *uid*, which may be a user name (looked up in */etc/passwd*) or number.

unhide Unhide the node. If the node resides in a subdirectory, all parent directory nodes must be visible to be able to access the node.

IMPLEMENTATION NOTES

Rulesets are created by the kernel at the first reference and destroyed when the last reference disappears. E.g., a ruleset is created when a rule is added to it or when it is set as the current ruleset for a mount-point, and a ruleset is destroyed when the last rule in it is deleted and no other references to it exist (i.e., it is not included by any rules and it is not the current ruleset for any mount-point).

Ruleset number 0 is the default ruleset for all new mount-points. It is always empty, cannot be modified or deleted, and does not show up in the output of **showsets**.

Rules and rulesets are unique to the entire system, not a particular mount-point. I.e., a **showsets** will return the same information regardless of the mount-point specified with **-m**. The mount-point is only relevant when changing what its current ruleset is or when using one of the apply commands.

FILES

/etc/defaults/devfs.rules	Default **devfs** configuration file.
/etc/devfs.rules	Local **devfs** configuration file. Rulesets in here override those in */etc/defaults/devfs.rules* with the same ruleset number, otherwise the two files are effectively merged.
/etc/devfs.conf	Boot-time **devfs** configuration file.
/usr/share/examples/etc/devfs.conf	Example boot-time **devfs** configuration file.

EXAMPLES

When the system boots, the only ruleset that exists is ruleset number 0; since the latter may not be modified, we have to create another ruleset before adding rules. Note that since most of the following examples do not specify **-m**, the operations are performed on */dev* (this only matters for things that might change the properties of nodes).

Specify that ruleset 10 should be the current ruleset for */dev* (if it does not already exist, it is created):

```
devfs ruleset 10
```

Add a rule that causes all nodes that have a path that matches "speaker" (this is only */dev/speaker*) to have the file mode 666 (read and write for all). Note that if any such nodes already exist, their mode will not be changed unless this rule (or ruleset) is explicitly applied (see below). The mode *will* be changed if the node is created *after* the rule is added (e.g., the *atspeaker* module is loaded after the above rule is added):

```
devfs rule add path speaker mode 666
```

Apply all the rules in the current ruleset to all the existing nodes. E.g., if the below rule was added after */dev/speaker* was created, this command will cause its file mode to be changed to 666 as prescribed by the rule:

```
devfs rule applyset
```

For all devices with a path that matches "snp*", set the file mode to 660 and the GID to "snoopers". This permits users in the "snoopers" group to use the snp(4) devices (quoting the argument to **path** is often necessary to disable the shell's globbing features):

```
devfs rule add path snp* mode 660 group snoopers
```

Add a rule to ruleset number 20. Since this ruleset is not the current ruleset for any mount-points, this rule is never applied automatically (unless ruleset 20 becomes a current ruleset for some mount-point at a later time):

```
devfs rule -s 20 add type disk group wheel
```

Explicitly apply all rules in ruleset number 20 to the DEVFS mount on */my/jail/dev*. It does not matter that ruleset 20 is not the current ruleset for that mount-point; the rules are still applied:

```
devfs -m /my/jail/dev rule -s 20 applyset
```

Since the following rule has no conditions, the action (**hide**) will be applied to all nodes:

```
devfs rule apply hide
```

Since hiding all nodes is not very useful, we can undo it. The following applies **unhide** to all the nodes, causing them to reappear:

```
devfs rule apply unhide
```

Add all the rules from the file *my_rules* to ruleset 10:

```
devfs rule -s 10 add - < my_rules
```

The below copies all the rules from ruleset 20 into ruleset 10. The rule numbers are preserved, but ruleset 10 may already have rules with non-conflicting numbers (these will be preserved). Since **show** outputs valid rules, this feature can be used to copy rulesets:

```
devfs rule -s 20 show | devfs rule -s 10 add -
```

SEE ALSO

chmod(1), jail(2), glob(3), devfs(5), devfs.conf(5), devfs.rules(5), chown(8), jail(8), mknod(8), service(8)

HISTORY

The **devfs** utility first appeared in FreeBSD 5.0.

AUTHORS

Dima Dorfman