



udev

An Introduction To Linux's udev.

Adam John Trickett
www.iredale.net

Talk Structure

- History
- Distros
- Kernel
- udev daemon
- udev tools
- udev configuration
- Why?
- Some Links

In The Beginning...

- Everything in UNIX is a file
- All files are the “same”
- Therefore devices appear as files
 - Thus devices appear as files in the /dev directory
- Linux is a copy of UNIX so it has devices in /dev

Static /dev

- Early computers didn't have too many devices, so all the possible devices were created as files at install time in the /dev directory. e.g.
 - `crw-rw-rw- 1 root root 1, 3 2007-07-19 21:03 /dev/null`
 - `brw-rw---- 1 root disk 8, 1 2007-07-19 20:03 /dev/sda1`
 - `crw-rw---- 1 root lp 99, 0 2007-07-19 20:04 /dev/parport0`
- If you needed a new device you had to use the /dev/MAKEDEV tool
- Became unmanageable as the number of devices became huge and hot-swapping became common

Linux 2.4 and devfs

- Kernel Module
- Dynamic and automatic population of /dev
- Similar is used in Solaris and *BSD
- **KNOWN RACE CONDITION IN LINUX**
- Hard to configure from the user-land
- Not maintained anymore
- Not all distos used it with 2.4 kernels

Linux 2.6 and udev

- No code in udev is in the kernel
 - Uses hotplug/sysfs to discover what is going on in the kernel
- Has user-space configuration
- Has user-space tool-kit
- Small and fast
- Is maintained
- Does not introduce “bloat” into the kernel

Distros and Versions

- Debian GNU/Linux
 - Woody (2002) static /dev
 - Sarge (2005) devfs as default, but could use udev
 - Etch (2007) udev as default
- Red Hat Linux & Red Hat Enterprise Linux
 - RHL7 (2001) static /dev
 - RHEL2 ('02) & RHEL3 ('03) static /dev
 - RHEL5 (2007) udev as default

Kernel Components

- Kernel needs to tell user-land what is going on:
 - `sysfs`
 - `/proc`
 - `/sys`
 - Unix domain sockets
 - Networking enabled
- **NONE** of the kernel code is `udev`, but `udev` needs to know what is going on in the kernel to know how to populate `/dev`

udev Daemon

- Started at boot time
- User space tool – you can stop and start it
- Watches the kernel and updates the /dev filesystem in realtime
 - `/etc/rcS.d/S03udev -> /etc/init.d/udev`
 - `/sbin/udev`
- Config files:
 - `/etc/udev`

udev tools

- The udev package comes with a number of binaries that you can use to diagnose what is going on and help with writing tools:
 - `/sbin/udevcontrol`
 - `/sbin/udevsettle`
 - `/sbin/udevtrigger`
 - `/usr/bin/dh_installudev`
 - `/usr/bin/udevinfo`
 - `/usr/bin/udevtest`
 - `/usr/sbin/udevmonitor`

udevcontrol

```
$ sudo udevcontrol --help
```

```
Usage: udevcontrol COMMAND
```

```
log_priority=<level>    set the udev log level for the daemon
stop_exec_queue         keep udevd from executing events, queue only
start_exec_queue        execute events, flush queue
reload_rules            reloads the rules files
max_childds=<N>         maximum number of childds
max_childds_running=<N> maximum number of childds running at the
                        same time
--help                 print this help text
```

udevtrigger

```
$ sudo udevtrigger --help
```

```
Usage: udevtrigger OPTIONS
```

```
--verbose           print the list of devices while
                    running
--dry-run           do not actually trigger the events
--retry-failed      trigger only the events which have been
                    marked as failed during a previous run
--subsystem-match=<subsystem> trigger devices from a matching subsystem
--subsystem-nomatch=<subsystem> exclude devices from a matching subsystem
--attr-match=<file[=<value>]> trigger devices with a matching sysfs
                    attribute
--attr-nomatch=<file[=<value>]> exclude devices with a matching sysfs
                    attribute
--help             print this text
```

udevinfo

```
$ sudo udevinfo --help
```

```
Usage: udevinfo OPTIONS
```

```
--query=<type>      query database for the specified value:
  name              name of device node
  symlink           pointing to node
  path              sysfs device path
  env               the device related imported environment
  all               all values

--path=<devpath>    sysfs device path used for query or chain
--name=<name>       node or symlink name used for query

--root              prepend to query result or print udev_root
--attribute-walk    print all SYSFS_attributes along the device chain
--export-db         export the content of the udev database
--version           print udev version
--help             print this text
```

udevinfo example

```
$ sudo udevinfo --query=all --name=/dev/sdb
P: /block/sdb
N: sdb
S: cruzer
S: disk/by-id/usb-SanDisk_U3_Cruzer_Micro_000018742C6378EB
S: disk/by-path/pci-0000:00:10.4-usb-0:6:1.0-scsi-0:0:0:0
E: ID_VENDOR=SanDisk
E: ID_MODEL=U3_Cruzer_Micro
E: ID_REVISION=3.21
E: ID_SERIAL=SanDisk_U3_Cruzer_Micro_000018742C6378EB
E: ID_TYPE=disk
E: ID_BUS=usb
E: ID_PATH=pci-0000:00:10.4-usb-0:6:1.0-scsi-0:0:0:0
```

/var/log/kern.log

```
kernel: usb 5-6: new high speed USB device using ehci_hcd and address 2
kernel: usb 5-6: configuration #1 chosen from 1 choice
kernel: Initializing USB Mass Storage driver...
kernel: scsi2 : SCSI emulation for USB Mass Storage devices
kernel: usbcore: registered new driver usb-storage
kernel: USB Mass Storage support registered.
kernel: usb-storage: device found at 2
kernel: usb-storage: waiting for device to settle before scanning
kernel:   Vendor: SanDisk Model: U3 Cruzer Micro Rev: 3.21
kernel:   Type:   Direct-Access           ANSI SCSI revision: 02
kernel: SCSI device sdb: 1994385 512-byte hdwr sectors (1021 MB)
kernel: sdb: Write Protect is off
kernel: sdb: Mode Sense: 03 00 00 00
kernel: sdb: assuming drive cache: write through
kernel: SCSI device sdb: 1994385 512-byte hdwr sectors (1021 MB)
kernel: sdb: Write Protect is off
kernel: sdb: Mode Sense: 03 00 00 00
kernel: sdb: assuming drive cache: write through
kernel:   sdb: sdb1
kernel: sd 2:0:0:0: Attached scsi removable disk sdb
kernel:   Vendor: SanDisk Model: U3 Cruzer Micro Rev: 3.21
kernel:   Type:   CD-ROM           ANSI SCSI revision: 02
kernel: usb-storage: device scan complete
kernel: sr0: scsi3-mmc drive: 8x/40x writer xa/form2 cdda tray
kernel: sr 2:0:0:1: Attached scsi CD-ROM sr0
kernel: sd 0:0:0:0: Attached scsi generic sg0 type 0
kernel: sd 2:0:0:0: Attached scsi generic sg1 type 0
kernel: sr 2:0:0:1: Attached scsi generic sg2 type 5
```

udevtest

```

$ sudo udevtest /block/sdb
parse_file: reading '/etc/udev/rules.d/020_permissions.rules' as rules file
...many rules...
parse_file: reading '/etc/udev/rules.d/z99_hal.rules' as rules file
main: looking at device '/block/sdb' from subsystem 'block'
udev_rules_get_name: reset symlink list
udev_rules_get_name: add symlink 'cruzer'
run_program: 'usb_id -x'
run_program: '/lib/udev/usb_id' (stdout) 'ID_VENDOR=SanDisk'
run_program: '/lib/udev/usb_id' (stdout) 'ID_MODEL=U3_Cruzer_Micro'
run_program: '/lib/udev/usb_id' (stdout) 'ID_TYPE=disk'
run_program: '/lib/udev/usb_id' (stdout) 'ID_BUS=usb'
run_program: 'edd_id --export /dev/.tmp-8-16'
run_program: 'path_id /block/sdb'
run_program: '/lib/udev/path_id' (stdout) 'ID_PATH=pci-0000:00:10.4-usb-0:6:1.0-scsi-0:0:0:0'
run_program: '/lib/udev/path_id' returned with status 0
udev_rules_get_name: add symlink 'disk/by-id/usb-SanDisk_U3_Cruzer_Micro_000018742C6378EB'
udev_rules_get_name: add symlink 'disk/by-path/pci-0000:00:10.4-usb-0:6:1.0-scsi-0:0:0:0'
udev_rules_get_name: no node name set, will use kernel name 'sdb'
udev_device_event: device '/block/sdb' already in database, validate currently present symlinks
udev_node_add: creating device node '/dev/sdb', major='8', minor='16', mode='0660', uid='0', gid= '25'
udev_node_add: creating symlink '/dev/cruzer' to 'sdb'
main: run: 'socket:/org/kernel/udev/monitor'
main: run: 'socket:/org/freedesktop/hal/udev_event'

```

udevmonitor (edited)

```
$ sudo udevmonitor
```

```
Password:
```

```
udevmonitor prints the received event from the kernel [UEVENT]  
and the event which udev sends out after rule processing [UDEV]
```

```
UEVENT[1185798928.759331] add@/devices/pci0000:00/0000:00:10.4/usb5/5-6  
UEVENT[1185798928.759359] add@/devices/pci0000:00/0000:00:10.4/usb5/5-6/usbdev5.4_ep00  
UEVENT[1185798928.760788] add@/devices/pci0000:00/0000:00:10.4/usb5/5-6/5-6:1.0  
UEVENT[1185798928.760807] add@/class/scsi_host/host4  
UEVENT[1185798928.760818] add@/devices/pci0000:00/0000:00:10.4/usb5/5-6/5-6:1.0/usbdev5.4_ep01  
UEVENT[1185798928.760824] add@/class/usb_device/usbdev5.4  
UDEV [1185798928.761702] add@/devices/pci0000:00/0000:00:10.4/usb5/5-6  
UDEV [1185798928.889463] add@/class/scsi_host/host4  
UDEV [1185798928.911609] add@/devices/pci0000:00/0000:00:10.4/usb5/5-6/5-6:1.0/usbdev5.4_ep01  
UDEV [1185798928.977445] add@/class/usb_device/usbdev5.4  
UEVENT[1185798933.759002] add@/class/scsi_disk/4:0:0:0  
UEVENT[1185798933.768135] add@/block/sdb  
UEVENT[1185798933.768158] add@/block/sdb/sdb1  
UEVENT[1185798933.768163] add@/class/scsi_device/4:0:0:0  
UEVENT[1185798933.768168] add@/class/scsi_generic/sg1  
UEVENT[1185798933.774025] add@/block/sr0  
UEVENT[1185798933.774050] add@/class/scsi_device/4:0:0:1  
UEVENT[1185798933.774055] add@/class/scsi_generic/sg2  
UDEV [1185798933.789034] add@/class/scsi_disk/4:0:0:0  
UDEV [1185798933.801308] add@/block/sdb  
UDEV [1185798934.086578] add@/block/sr0  
UDEV [1185798934.143578] add@/block/sdb/sdb1
```

Configuration and Rules

- udev is designed to be configured by configuration files outside the kernel
- udev is designed to be configured in real-time
- udev does not care about device order on the bus or device initiation order
- udev allows the user to set arbitrary rules and create symlinks as the user sees fit
- Everything lives in
 - /etc/udev

/etc/udev

- `udev.conf`
- `links.conf`
- `*.rules` (Debian)
- `rules.d/`
 - symlinks back to `../`
 - Think `/etc/init.d/` and `/etc/rc*.d/`

Rules...

- By default your distro will come with rules to create all the things you expect
- Most things you could plug-in should automatically be detected and created in the /dev directory
- udev will also notify higher tools via d-bus/HAL, so KDE and GNOME should auto-load the hardware
- **BUT** sometimes it's fun or useful to create your own rules...

/etc/udev/local.rules

- You can put your own rules in this file
- Rules follow the following format, multiple pairs per line:
 - key operator “value”,
 - assignment operator “value”
- Keys:
 - BUS, SYSFS or KERNEL
- Operator:
 - == equals, != not-equal, = assign, += append assign

local.rules example

```
# my cheap Konica digital camera
BUS=="scsi", SYSFS{model}=="Camera KD-25", SYMLINK="konica_camera"

# A Western Digital hard disk in an external USB ICY container
BUS=="scsi", SYSFS{vendor}=="WDC AC34", SYSFS{model}=="000L", \
SYMLINK="icy_box"

# A Dabs "Easy" brand USB key
BUS=="scsi", SYSFS{vendor}=="Easy", SYSFS{model}=="Disk", SYMLINK="easy_disk"

# A SanDisk U3 Cruzer Micro USB key
BUS=="scsi", SYSFS{vendor}=="SanDisk", SYSFS{model}=="U3 Cruzer Micro", \
SYMLINK="cruzer"
```

Why Bother?

- Most of the time there is no need to bother BUT if you want to ensure a device ALWAYS appears where you want it you can
- Why do you want a device or symlink to appear in the same place?
 - So you can put a static line in your `/etc/fstab`
 - `/dev/cruzer /media/cruzer vfat user,noauto,noatime 0 0`
 - Now you can just plug the device in and use a script to mount/umount it without worrying where it appears in `/dev` as you always have a symlink

More Reasons

- Specific permissions
- Specific ownership
- Even works on network interfaces
- Execute external programs on insertion/removal of device

Resources

- <http://www.debian-administration.org/articles/126>
- <http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html>
- http://www.reactivated.net/writing_udev_rules.html

Thank You.

Any
Questions?