UDEV
Gérerer les device node

Mickaël Tansorier

Présentation du fonctionnement d'UDEV.

Plan

UDEV est un démon chargé de gérer les device-nodes.

Principe d'unix

Tout est fichier

Donc les périphériques matériels sont des pseudo-fichier.
Ils peuvent être manipulés par : cat, grep, open(), read(), write() etc...

Ils sont stockés dans : /dev
On les appelle device-nodes.

Les device nodes sont :

- des points d'entrée vers le noyau
- de type bloc ou char
- identifiés par un majeur et un mineur

Le majeur permet au noyau de savoir quel driver doit gérer le périphérique.
Le mineur permet au driver de savoir quel périphérique parmi ceux qu'il gère est utilisé.

- Le script `MAKEDEV`
  - Les majeur et mineur était codés en dur dans un script appelé par la commande mknode.
  - Pour créer les périphériques dans `/dev`.
  - Tout ceux possible existaient sur le système (~ 18 000)
  - Ne gère pas le hotplug
- Le `devfs`
  - Les noeuds ne sont plus crées par le système mais par les drivers eux-mêmes à la détection d'un périphérique.
  - Mais les majeurs mineurs étaient toujours codé en dur
  - devfs est difficile à paramétrer
- Le `udev`
  - udev est un programme purement user-space
  - udev évolue en s'appuyant sur les évolutions du noyau
  - Il s'appuie sur les informations laissées dans `/sys` (majeur et mineur)
  - Le noyau fournit un lien netlink qui permet à udev d'être prévenu lors d'un changement.

udev réagit donc à des événements envoyés par le noyau via `inotify`.

```
udevadm monitor -k
```

Lorsque l'on branche un USB :

```
KERNEL[805.692293] add      /devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1 (usb)
KERNEL[805.694999] add      /devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1/1-1:1.0 (
    usb)
KERNEL[806.006810] add      /devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1/1-1:1.0/
    net/eth1 (net)
KERNEL[806.006948] add      /devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1/1-1:1.0/
    net/eth1/queues/rx-0 (queues)
KERNEL[806.007010] add      /devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1/1-1:1.0/
    net/eth1/queues/tx-0 (queues)
```

Et si l'on retire :

```
KERNEL[800.026043] remove   /devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1/1-1:1.0/
    net/eth1/queues/rx-0 (queues)
KERNEL[800.026228] remove   /devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1/1-1:1.0/
    net/eth1/queues/tx-0 (queues)
KERNEL[800.026366] remove   /devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1/1-1:1.0/
    net/eth1 (net)
KERNEL[800.050534] remove   /devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1/1-1:1.0 (
    usb)
KERNEL[800.080535] remove   /devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1 (usb)
```

On peut demander à udev de nous fournir les propriétés du périphérique :

```
udevadm monitor -k -p
```

Pour une seule sortie :

```
KERNEL[5763.052898] add      /devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1 (usb)
ACTION=add
BUSNUM=001
DEVNAME=/dev/bus/usb/001/003
DEVNUM=003
DEVPATH=/devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1
DEVTYPE=usb_device
MAJOR=189
MINOR=2
PRODUCT=bda/8153/3000
SEQNUM=1544
SUBSYSTEM=usb
TYPE=0/0/0
```

Les propriétés dépendent du type exact de périphérique mais certaines propriétés sont toujours présentes :

- `ACTION` : Le type d'événement à traiter,
- `MAJOR`, `MINOR` : Les numéro majeur et mineur du périphérique concerné,
- `SEQNUM` : un numéro croissant pour ordonner les événements,
- `SUBSYSTEM` : Le sous-système noyau ayant causé l'événement,
- `DEVPATH` : Le fichier dans `/sys` correspondant au périphérique.

L'écriture de règles

Les règles udev peuvent venir de plusieurs endroits différents :

```
/etc/udev/rules.d/      # regles locales ajoutees par l'administrateur
/run/udev/rules.d/      # regles volatiles, generalement creees par d'autres regles
/usr/lib/udev/rules.d/  # regles fourni par la distribution
/lib/udev/rules.d/      # regles fourni par la distribution
```

Les fichiers sont préfixé par un numéro (comme init), et postfixé par `.rules`.
Exemple : `/usr/lib/udev/rules.d/82-net-usb-up.rules`

```
ACTION=="add", RUN+="/bin/sh -c '/bin/echo %k : %p >> /root/events'"
```

Pour filtrer les évènements on utilise la syntaxe : `PROPRIETE==valeur`
`"=="`, `"!="`, `"="`, `"+="`, `"-="`, `":="`

Le champs `PROPRIETE` peut prendre les valeurs suivantes :

- `ATTR{fichier}` — permet de filtrer sur le contenu d'un fichier dans le répertoire sysfs correspondant à l'événement en cours.
- `ATTRS{fichier}` — idem + également sur ceux des périphériques parents (bus, plateforme etc. . .).
- `ENV{clé}` — permet de filtrer sur une propriété ajoutée par une autre règle udev.
- `PROGRAM` — permet de filtrer sur la valeur de retour d'un programme.

Lorsque l'on a le filtrage que l'on souhaite, on peut effectuer des actions avec les options suivantes :

- `SYMLINK` — crée un lien symbolique vers le fichier `/dev` du pérphérique. Le noyau fournit un premier fichier `/dev` dont le nom ne peut pas être changé.
- `OWNER`, `GROUP`, `MODE`, `SECLABEL{module}` — permet de préciser les droits d'accès et les propriétés SELinux du périphérique.
- `ATTR{fichier}` — permet d'écrire une valeur dans un fichier du répertoire sysfs correspondant au périphérique.
- `ENV{clé}` — permet de positionner un attribut interne à udev pour le périphérique en cours,
- `RUN` — permet d'exécuter un programme

Certain mots clés peuvent être utilisés pour ces actions, comme :

- `$kernel`, `%k` — The kernel name for this device.
- `$devpath`, `%p` — The devpath of the device.

Exemple :

```
ACTION=="add", SUBSYSTEM=="net", RUN+="/bin/sh -c '/bin/echo %k : %p >> /root/events'"
```

Donne

```
$ cat /tmp/event
usb0 : /devices/soc0/soc/2100000.aips-bus/2184000.usb/ci_hdrc.0/gadget/net/usb0
eth0 : /devices/soc0/soc/2100000.aips-bus/2188000.ethernet/net/eth0
eth1 : /devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1/1-1:1.0/net/eth1
sit0 : /devices/virtual/net/sit0
lo : /devices/virtual/net/lo
```

retrouver le répertoire sysfs d'un device :

```
$ udevadm info -x -q path -n /dev/hidraw0
/devices/pci0000:00/0000:00:14.0/usb1/1-1/1-1.3/1-1.3:1.0/0003:046D:C01A.0015/hidraw/hidraw0
```

Obtenir des info sur le device :

```
$ udevadm info /sys/devices/pci0000:00/0000:00:14.0/usb1/1-1/1-1.3/1-1.3:1.0/0003:046D:C01A.0015/hidraw/
    hidraw0
P: //devices/pci0000:00/0000:00:14.0/usb1/1-1/1-1.3/1-1.3:1.0/0003:046D:C01A.0015/hidraw/hidraw0
N: hidraw0
E: DEVNAME=/dev/hidraw0
E: DEVPATH=//devices/pci0000:00/0000:00:14.0/usb1/1-1/1-1.3/1-1.3:1.0/0003:046D:C01A.0015/hidraw/hidraw0
E: MAJOR=247
E: MINOR=0
E: SUBSYSTEM=hidraw
```

- E : une propriété purement udev. elles sont accessibles grâce à `ENV{}`
- N : le nom du périphérique suggéré par le noyau
- S : les liens symboliques créés vers le périphérique
- P : le chemin vers l'entrée dans `sysfs`

Monitorer les actions en direct :

```
$ udevadm monitor -p
```

L'option -p permet d'avoir les propriétés des évènements.

Tester un évènement :

```
$ udevadm test -a add /sys/devices/pci0000:00/0000:00:14.0/usb1
   /1-1/1-1.3/1-1.3:1.0/0003:046D:C01A.0018/hidraw/hidraw0
```

```
$ cat board/eolane/modx6/rootfs_pendant/usr/lib/udev/rules.d/82-net-
    usb-up.rules
# do not edit this file, it will be overwritten on update

ACTION!="add", GOTO="net_usb_up_end"
SUBSYSTEM!="net", GOTO="net_usb_up_end"

ENV{ID_BUS}=="usb", RUN+="/bin/sh -c 'ifup %k'"

LABEL="net_usb_up_end"
```

`mdev` est un utilitaire similaire à `udev` présent dans `busybox`. Très utilisé dans l'embarqué.

Mais `udev` est déjà très léger (le binaire fait 520K).

Question ?

Enfin je vais essayer de répondre...

Document basé sur :
`http://www.linuxembedded.fr/2015/05/une-introduction-a-udev/`