HEIG VD HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD

Introduction aux drivers kernel-space

TIC

unité d'enseignement DRV

Auteurs: **Alexandre Iorio**

Professeur: **Alberto Dassatti**

Assistant: **Clément Dieperink**

Salle de laboratoire **A09**

Date: **07.11.2024**

Table des matières

Exercice 1 - mknod

Afin d'utiliser la commande `mknod`, nous avons pu relever les arguments intéressants suivants:

- `mknod /dev/DRV_test c 42 1` : Crée un fichier spécial de caractère nommé `DRV_test` dans le répertoire `/dev` avec les numéros majeur et mineur 42 et 1 respectivement.

En créant ce même fichier avec les arguments identiques à ceux de random, à savoir 1 en numéro mineur et 8 en numero majeur:

```
ø /
dev
x INT at 19:53:08
❯ ls -la | grep random
crw-rw-rw-   1 root   root        1,     8 Nov  7
12:34 random
crw-rw-rw-   1 root   root        1,     9 Nov  7
12:34 urandom
```

Nous avons pu constater que le fichier `DRV_test` a bien été créé:

```
ø /
dev
at 20:03:32
❯ sudo mknod /dev/DRV_test c 1
8


ø /
dev
at 20:04:13
❯ ls -la | grep -E "random|DRV_test"
crw-r--r--   1 root   root        1,     8 Nov  7
20:04 DRV_test
crw-rw-rw-   1 root   root        1,     8 Nov  7
12:34 random
crw-rw-rw-   1 root   root        1,     9 Nov  7
12:34 urandom
```

Son contenu semble identique à celui de `random`.

Exercice 2 - Proc

Le fichier `/proc/devices` contient la liste des périphériques de caractères et de blocs reconnus par le noyau.

Après le branchement de la `De1-SoC` , nous pouvons retrouver l'information de `/ttyUSB0`

```
⌀ /dev
❯ cat /proc/devices | grep -E "tty|:"
Character devices:
  4 tty
  4 ttyS
  5 /dev/tty
  5 ttyprintk
188 ttyUSB
204 ttyMAX
242 ttyDBC
```

On trouve effectivement `ttyUSB` dans la liste des périphériques de caractère et que son numéro majeur est 188.

En recherchant dans le `sysfs`, on trouve `ttyUSB0` dans plusieurs répertoires:

```
⌀ /
at 20:23:01
❯ sudo find ./sys -name 'ttyUSB*'

./sys/class/tty/ttyUSB0
./sys/devices/pci0000:00/0000:00:14.0/
usb3/3-3/3-3:1.0/ttyUSB0
./sys/devices/pci0000:00/0000:00:14.0/
usb3/3-3/3-3:1.0/ttyUSB0/tty/ttyUSB0
./sys/bus/usb-serial/devices/ttyUSB0
./sys/bus/usb-serial/drivers/ftdi_sio/ttyUSB0
```

Exercice 5

Dans un premier temps, nous allons compiler pour la machine hôte.

```
~/heig/drv/labos/DRV/material/lab_03/
parrot_module on lab03 ⇡1
❯ make CC=x86_64-linux-gnu-gcc-13
```

On lance un trackeur de log:

```
~/heig/drv/labos/DRV/material/lab_03/
parrot_module on lab03 ⇡1
❯ sudo dmesg -w | grep ioctl
```

puis inserer le module:

```
~/heig/drv/labos/DRV/material/lab_03/
parrot_module on lab03 ⇡1
❯ insmod parrot.ko
```

On vérifie que le module est bien chargé:

```
⌀ /
dev
❯ lsmod | grep parrot
parrot                 12888  0
```

On analyse la sortie du trackeur de log:

```
~/heig/drv/labos/DRV/material/lab_03/
parrot_module on lab03 ⇡1
❯ sudo dmesg -w | grep ioctl
[ 5001.991867] ioctl PARROT_CMD_TOGGLE: 11008
[ 5001.991869] ioctl PARROT_CMD_ALLCASE:
1074014977
```

On connait maintenant les valeurs des commandes `PARROT_CMD_TOGGLE` et `PARROT_CMD_ALLCASE`.

Après avoir relevé le numéro majeur du module dans le fichier `parrot.c`, nous allons créer un node avec `mknod`:

```
⌀ /
dev
❯ sudo mknod my_node c 97 0
```

On lui donne full access:

```
⌀ /
dev
❯ sudo chmod 777 my_node
```

On ecris dans le fichier:

```
⌀ /
dev
❯ echo "Hello World" > my_node
```

On lit le contenu du fichier:

```
⌀ /dev
❯ cat my_node
Hello World
```

On compile ioctl.c:

```
~/heig/drv/labos/DRV/material/lab_03
❯ gcc -o ioctl ioctl.c
```

On execute le programme avec `PARROT_CMD_TOGGLE`:

```
~/heig/drv/labos/DRV/material/lab_03
❯ ./ioctl /dev/my_node 11008 0
```

On lit le contenu du fichier:

```
⌀ /dev
❯ cat my_node
hELLO wORLD
```

On lance le programme avec `PARROT_CMD_ALLCASE` et l'argument `0`:

```
~/heig/drv/labos/DRV/material/lab_03
❯ ./ioctl /dev/my_node 1074014977 0
```

On lit le contenu du fichier:

```
ø /dev
❯ cat my_node
HELLO WORLD
```

On lance le programme avec `PARROT_CMD_ALLCASE` et l'argument `1`:

```
~/heig/drv/labos/DRV/material/lab_03
❯ ./ioctl /dev/my_node 1074014977 1
```

On lit le contenu du fichier:

```
ø /dev
❯ cat my_node
hello world
```

Nous avons modernisé le driver. Maintenant le node perrot apparait directement dans `/dev/` et nous pouvons faire les mêmes actions qu'avant.

Un Case permettant de restituer la string initial est ajouté. Voici la valeur à passer en argument à `ioctl`:

```
ø /
dev
❯ sudo dmesg | grep ioctl
[ 2896.392884] ioctl PARROT_CMD_TOGGLE: 11008
[ 2896.392888] ioctl PARROT_CMD_ALLCASE:
1074014977
[ 2896.392891] ioctl PARROT_CMD_RESET: 11010
```

Testons le tout:

```
ø /
dev
at 16:14:04
❯ sudo chmod 777 parrot

ø /
dev
at 16:14:21
❯ echo "Hello World" >> parrot
```

```
⌀ /
dev
at 16:14:29
❯ cat parrot
Hello World
```

On execute `ioctl`:

```
~/heig/drv/labos/DRV/material/lab_03 on lab03 !
2
at 16:06:13
❯  ./ioctl /dev/parrot 1074014977 1


⌀ /
dev
at 16:14:37
❯ cat parrot
hello world


~/heig/drv/labos/DRV/material/lab_03 on lab03 !
3
at 16:16:52
❯  ./ioctl /dev/parrot 11010 0


⌀ /
dev
at 16:16:57
❯ cat parrot
Hello World
```

On se rend compte que le driver fonctionne correctement mis a part le problème de permission.

En effet, nous avons testé le uevent et nous avons pu constater que le driver est bien chargé et que le uevent est bien appelé, par contre la permission à du être mise à la main.

Par simplicité, la permission à été mise à 777, mais il serait plus judicieux de mettre une permission plus adéquate.

make CC=arm-linux-gnueabihf-gcc-6.4.1