Module d'approfondissement SEEE

Systèmes d'exploitation et environnements d'exécution embarqués

Prof. Daniel Rossier / Magali Fröhlich

Drivers

Labo reptar3 *(v2017.1.4)*

Ce laboratoire se concentre sur le développement de *drivers* sur la plate-forme *Reptar* (émulée et réelle). Il permet de se familiariser avec le modèle orienté plate-forme de GNU/Linux et des structures de données importantes du noyau. La mise au point des *drivers* s'effectuera en alternance sur l'environnement émulé et sur la plate-forme réelle.

Consignes de laboratoire

- Ce laboratoire est à faire en binôme.
- Un rapport détaillé doit être élaboré par groupe et sera évalué.

Prologue **– Mise à jour du *workspace***

a) Assurez-vous que le projet *drivers* est présent dans le *workspace* d'Eclipse.
b) Dans le dossier *drivers*, ouvrez le fichier *reptar_sp6_buttons.c*
c) Désactivez les lignes 20 à 23, en utilisant les directives *#if 0* et *#endif*, comme suit :

```
#if 0

#include <xen-guest/io/kbdif.h>
#include <xen-guest/console.h>

extern int omapfb_xen_switch_domain(domid_t dom);

#endif /* 0 */
```

d) Pour lancer correctement les applications *Qt* dans l'environnement émulé (voir ***Etape no 4***), vous avez besoin d'installer la librairie *sdl* avec la commande suivante :

```
$ sudo apt-get install libsdl1.2-dev
```

❖ Attention, pour réussir l'installation, la VM doit avoir accès à internet.

Etape no. 1 **– Environnement *Qemu* et plate-forme *Reptar***

Dans cette étape, nous allons déployer l'environnement de la cible à partir d'une image de carte MMC (*sdcard*) et nous nous familiariserons avec l'insertion dynamique de module.
Le projet concerné est le projet *drivers* (répertoire du même nom à la racine du *workspace*).

Le répertoire contient les fichiers C suivants :

- *reptar_sp6.c, reptar_sp6.h* Fichier principal contenant le *driver* pour la *FPGA*
- *reptar_sp6_leds.c* Fichier contenant le *driver* pilotant les LEDs
- *reptar_sp6_buttons.c* Fichier contenant le *driver* pilotant les boutons

- *buttons_test.c* Application utilisateur pour tester les boutons
- *ledstest.sh* Script utilisateur pour tester les LEDs
- *usertest.c* Application utilisateur pour tester le *driver* de type caractère

Le *Makefile* produira le fichier ***sp6.ko*** ainsi que les deux applications de test : ***usertest*** et ***buttons_test***

❖ Pour déployer les deux applications sur la cible émulée ou réelle, il est possible d'employer la même méthode suivie dans le premier laboratoire (labo d'introduction *reptar1*).

Vous trouverez, dans *seee_student*, trois scripts utiles :

- *mount-sd.sh* Montage de la *sdcard* dans les répertoires ***filesystem_tmp*** *(rootfs)* et ***boot_tmp***
- *umount-sd.sh* Démontage de la *sdcard* (nécessaire avant toute autre manipulation sur celle-ci)
- *deploy* Montage de la *sdcard*, copie du fichier *driver* (*sp6.ko*) et des applications à la racine du *rootfs*, démontage

a) Lancez *make* dans le répertoire *drivers/*
b) A la racine du *workspace*, lancez les scripts suivants, puis la commande *boot* dans *U-boot* :

```
$ ./deploy
$ ./stf

Reptar # boot
```

c) A la racine du *rootfs* (*cd /*), insérez le module avec la commande suivante :

```
# insmod sp6.ko
```

Vérifiez qu'il n'y ait aucun message d'erreur. La liste des modules chargés dynamiquement est obtenue avec la commande *lsmod* et le retrait du module avec la commande *rmmod*

```
# lsmod
# rmmod sp6
reptar_sp6: bye bye!
#
```

Etape no. 2 – Driver de type caractère

Cette étape consiste à travailler sur un *driver* de type caractère au niveau FPGA. Le code de cette partie se trouve dans les fichiers *reptar_sp6.h* et *reptar_sp6.c*.
Sur la base du code existant, on souhaite pouvoir écrire et lire une chaîne de caractères contenant la version du *bitstream* (hypothétique) dans la FPGA, stockée dans la variable globale *bitstream_version*.

a) Complétez les *callbacks read()* et *write()* afin qu'une application utilisateur puisse lire et écrire une chaîne de (80 max.) caractères.

b) Pour identifier le nom de l'entrée dans */dev/* qui sera créée automatiquement, examinez la fonction ***probe( )*** du *driver*.

⇒ Comment l'entrée dans */dev* est-elle générée ?
⇒ Quel sera le nom de l'entrée dans */dev* ?

c) Afin de tester votre *driver,* écrivez une application *usertest* (fichier *usertest.c*) qui écrira puis relira la chaîne de version en utilisant l'entrée dans */dev* évoquée ci-dessus.

⇒ Recherchez les valeurs du *major* et *minor* attribuées à ce *driver*. Expliquez votre démarche.

Etape no. 3 – Pilotage des LEDs

Le code de pilotage des LEDs se trouve dans le fichier *reptar_sp6_leds.c*. La réalisation du *driver* des LEDs.

- L'application graphique *qtemu* sera utilisée pour l'environnement émulé.
- Le *driver* devra être également testé sur la plate-forme réelle.

Pour le pilotage des LEDs, on souhaite utiliser le sous-système *leds* présent dans le noyau *Linux*.

Effectuez un mappage du registre des LEDs à l'aide de la fonction ***ioremap( )***, en vous servant de la structure *fpga_resource*.

a) Enregistrez le *device* comme un *device* de type *leds* à l'aide de la fonction ***led_classdev_register( )***.
b) Cherchez et implémentez le(s) *callback(s)* gérant l'enclenchement/déclenchement des *LEDs*.

⇒ Combien y a-t-il de *devices* de type LED gérés par notre *driver* ?

c) Testez le *driver* LED dans l'environnement *qtemu* (application graphique). Lancez le script *ledstest.sh*.
d) Testez votre *driver* sur la plate-forme (réelle) *Reptar*.

Etape no. 4 – Pilotage des boutons

Lors de cette étape, nous travaillerons sur le *driver* gérant la pression des boutons. L'objectif est de contrôler une application dans l'espace utilisateur à l'aide des boutons.

a) Complétez la fonction ***probe( )*** pour l'enregistrement des deux *callbacks* d'interruption (traitement immédiat + traitement différé) à l'aide de la fonction ***request_threaded_irq( )***.
b) Implémentez le traitement immédiat lié à l'interruption. Celui-ci devra :

   1) **stocker** la valeur du registre bouton dans le champ ***current_button*** de la structure privée (l'adresse du registre contenant cette information est également disponible dans la structure privée),
   2) **acquitter** l'interruption.

c) Testez votre *driver* des boutons à l'aide de l'application *buttons_test*. Dans l'environnement émulé, l'application devra ouvrir le fichier */dev/input/event1*. Il faudra taper la commande suivante:

```
# ./buttons_test -e1
```

d) Testez les boutons un par un.
e) Testez votre *driver* avec une application *Qt* disponible dans */usr/share/qt/demos*. Essayez *textedit* par exemple :

```
# export QWS_KEYBOARD="LinuxInput:/dev/input/event1"
# cd /usr/share/qt/demos/textedit
# ./textedit -qws
```

f) Effectuez un déploiement et un test sur la plate-forme réelle *Reptar*. Sur celle-ci, vous devrez utiliser l'entrée */dev/input/event2*

- A la fin de cette étape, vous devrez être en mesure de **décrire le cheminement d'une interruption en provenance d'un bouton** jusqu'à l'effet perçu au niveau de l’application utilisateur.