MA-CSEL1 (construction systèmes embarqués sous Linux) – Exercices

Programmation noyau : Pilotes de périphériques

Pilotes orientés mémoire

1. Réaliser un pilote orienté mémoire permettant de mapper en espace utilisateur les registres du µP en utilisant le fichier virtuel « /dev/mem ». Ce pilote permettra de lire l'identification du µP (Chip-ID aux adresses 0x01c1'4200 à 0x01c1'420c) décrit dans l'exercice n° 5 du cours sur la programmation de modules noyau.

Pilotes orientés caractère

2. Implémenter un pilote de périphérique orienté caractère. Ce pilote sera capable de stocker dans une variable globale au module les données reçues par l'opération write et de les restituer par l'opération read. Pour tester le module, on utilisera les commandes « echo » et « cat ».

3. Etendre la fonctionnalité du pilote de l'exercice #2 afin que l'on puisse à l'aide d'un paramètre module spécifier le nombre d'instances. Pour chaque instance, on créera une variable unique permettant de stocker les données échangées avec l'application en espace utilisateur.

4. Développer une petite application en espace utilisateur permettant d'accéder à ces pilotes orientés caractère. L'application devra écrire un texte dans le pilote et le relire.

sysfs

5. Développer un pilote de périphérique orienté caractère permettant de valider la fonctionnalité du sysfs. Le pilote offrira quelques attributs pouvant être lus et écrites avec les commandes « echo » et « cat ». Ces attributs seront disponibles sous l'arborescence « /sys/class/… ».

6. Adapter l'implémentation de l'exercice #3 ci-dessous afin que celui-ci utilise un « device tree » (DT) pour décrire le nombre de périphériques à mettre en œuvre. Le DT sera externe à l'arborescence des sources du noyaux Linux. La structure « struct miscdevice » peut être utilisée pour instancier les « devices » et les fichiers d'accès (/dev/…).

Opérations bloquantes

7. Développer un pilote et une application utilisant les entrées/sorties bloquantes pour signaler une interruption matérielle provenant de l'un des switches de la carte d'extension du NanoPI. L'application utilisera le service select pour compter le nombre d'interruptions.
   Remarque : les switches n'ont pas d'anti-rebonds, par conséquent il est fort probable que vous comptiez un peu trop d'impulsions ; effet à ignorer.

MA-CSEL1 (construction systèmes embarqués sous Linux) – Exercices

Programmation noyau : Pilotes de périphériques

Pilotes orientés mémoire *(optionel)*

8. Sur la base de l'exercice 1, développer un pilote orienté caractère permettant de mapper en espace utilisateur ces registres (implémentation de l'opération de fichier « mmap »).
   Le driver orienté mémoire sera ensuite adapté à cette nouvelle interface.
   Remarque : à effectuer après les exercices des pilotes orientés caractère.

Ioctl *(optionel)*

9. Implémenter à l'intérieur d'un pilote de périphérique l'opération ioctl afin de pouvoir
   a. Envoyer une commande
   b. Ecrire et lire une valeur entière
   c. Ecrire e lire un bloc de configuration de plus de 50 octets

   Afin de valider le pilote, développer une petite application permettant d'effectuer ces opérations et de les valider.

procfs *(optionel)*

10. Implémenter à l'intérieur d'un pilote de périphérique les opérations nécessaires afin de pouvoir lire un bloc de configuration et de pouvoir modifier le contenu de la valeur entière par procfs. Seules les commandes echo et cat doivent être nécessaires pour manipuler ces attributs.