TP8 - ASR2 - ENS Lyon - L3IF - 2019-2020
Module noyau et User I/O

Intervenants:

Stella Bitchebe (celestine-stella.ndonga-bitchebe@ens-lyon.fr)
Lavoisier Wapet (patrick-lavoisier.wapet@ens-lyon.fr)
Alain Tchana (alain.tchana@ens-lyon.fr)
(Note: ce sujet a été rédigé par Stella et Lavoisier)

Description

Notion de module noyau: Un module linux est un programme qui étend les fonctionnalités de base du noyau linux. Il peut être intégré au kernel sans effectuer tout le processus de recompilation. Les modules sont d'ailleurs utilisés pour implémenter la plupart des drivers linux.

Pour écrire un module linux, il faut ré-utiliser le template prévu à cet effet dans la documentation linux en y intégrant son propre code[1]. Dans ce TP il sera question d'implémenter un driver d'entrée sortie en user space (UIO).

Notion d'UIO: Un UIO (userspace I/O) permet d'implémenter la logique d'un driver en user space, il est donc constitué de deux parties:

- La première partie est un module linux qui crée et enregistre un nouvel uio device. Il lui réserve aussi de la mémoire kernel.
- La seconde partie est un programme en user space qui mappe la mémoire allouée par le module. Il peut ainsi l'utiliser pour implémenter les fonctionnalités du driver.

Partie 1 : La première partie du TP consiste à écrire, compiler puis à intégrer un module linux au noyau. Le squelette de ce module est mis à votre disposition en annexe de ce TP ("uio_device.c"). Il vous suffira de le completer avec le code source adéquat afin qu'il :

- **Enregistre** un UIO device: pour cela utiliser la fonction ***uio_register_device*** qui prend en paramètre un ***platform_device*** créé par ***platform_device_register_simple*** et une variable de type ***uio_info***. le champ mem[0] de la variable de type ***uio_info*** devra contenir les informations sur la mémoire kernel que vous aurez réservé avec *kzalloc* (son adresse , son type et sa taille). Se référer au lien [2] et aux explications.
- **Définit** un timer par la fonction ***timer_setup*** dont le callback simule l'envoie des interruption à la partie userspace de l'uio driver. Pour envoyer les interruptions dans le callback du timer utiliser la fonction ***uio_event_notify***.

Pour compiler et intégrer le module, suivre les instruction du [1]

- Faire le makefile du module
- Utiliser les commandes insmod et rmmod (pour l'intégration du module au noyau)

Partie 2 : La partie 2 du TP consiste à écrire un programme userspace qui complètera votre module pour en faire un driver d'UIO complet. Le programme à implémenter

- **Utilise *mmap*** pour accéder à la mémoire de l'uio device. Le fichier à ouvrir via ***open*** est sous la forme /dev/uioX, où X représente le nombre d'uio déjà enregistrés dans le noyau.
- **Puis attend** dans une boucle infinie une interruption en effectuant un *read* sur sur le descripteur obtenu . voir le lien [2], pour plus d'amples explications.

Liens utiles:

[1]
https://www.thegeekstuff.com/2013/07/write-linux-kernel-module/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+TheGeekStuff+%28The+Geek+Stuff%29

[2]
https://topic.alibabacloud.com/a/uio-mechanism-of-linux-device-driver_1_16_30147665.html