EPITA – SYS2 – Projet

WLKOM : Wild Linux Kernel Object Module

Rendu

Date de rendu : 15/06/2022 – 14h42

Mail : jules1.aubert@epita.fr avec la tarball

Tag en sujet du mail : [WLKOM] ...

Si vous êtes en solo, le nom de l'équipe est votre login, si vous êtes dans une équipe de 2 à 4, vous pouvez choisir un nom d'équipe.

A rendre : une tarball NOMDELEQUIPE.tar.gz qui contient le répertoire du projet

L'énoncé du projet se trouve plus bas. Si vous souhaitez un autre sujet, vous pouvez m'envoyer un mail avec le nom de l'équipe, le nom du projet, les élèves et l'énoncé de votre projet. Je vous répondrai pour vous dire si je valide votre idée.

Vous avez jusqu'au 25/04/2022 – 14h42 pour me propoer d'autres sujets en solo ou en équipe par mail. Après cette deadline, j'assumerai que tout le monde qui ne m'a rien proposé fait le projet proposé par défaut en solo.

Kernel utilisé pour la correction : 5.13.0-39-generic (Ubuntu 20.04.4 LTS)

Projet

Fichiers obligatoires dans la tarball :

- Makefile
- README (texte, Markdown, pdf, html, …)
- AUTHORS (le ou les logins, un par ligne)
- Les sources

Le fichier README doit présenter votre projet, ce qu'il fait, comment il fonctionne (succintement, pas besoin d'écrire du code dedans). Il doit aussi documenter l'utilisation de votre projet, comment je dois le compiler, le tester.

Enoncé : epirandom

Avec ce qui a été vu en cours, vous devez créer un kernel object qui utilise un fichier device (man mknod(1) sur la partie /dev)

Niveau 1

Le premier niveau (pour avoir entre 7/20 et 10/20 selon l'optimisation et la clarté du code et du projet) est de créer un kernel object qui, par un fichier device, sort une suite de chiffres de façon aléatoire sans fin. Je dois arrêter le flux par exmeple en faisant ctrl+c avec cat, ou en limitant la donnée à sortir avec dd.

Exemple 1 :

42sh$: cat /dev/epirandom
235711131719232931374 1(…)

Exemple 2 :

42sh$: dd if=/dev/epirandom of=/tmp/out bs=512 count=1
(…)
42sh$: 31415926535897932384626(…)

Attention, si je dump deux fichiers avec dd, de la même taille, je dois bien avoir des fichiers différents, et pas le même contenu.

Niveau 2

Vous prenez d'autres caractères, à savoir les caractères alphanumériques majuscules et minuscules, ainsi que les caractères de la table ASCII. Toujours sans fin.

Niveau 3

Vous sortez des ocets aléatoires comme le fait /dev/urandom (man urandom(4)). Toujours sans fin.

Niveau 4

Vous utilisez un paramètre pour définir votre alphabet pour sortir de l'aléatoire.

Exemple 1 :
42sh#: insmod ./epidriver.ko alphabet=a,b,c,X,Y,Z,0,1,2,3,4,5,6,7,8,9,\,,\?,\\

Exemple 2 :
42sh#: insmod ./epidriver.ko alphabet='abcXYZ0123456789,?\'

Ce niveau peut ne pas être compatible avec le niveau 3. Ca ne sera pas vu comme une faute s'il n'y a pas d'octets aléatoires dans cette partie. Ce qui m'intéresse, c'est la gestion des paramètres.

Documentations

La documentation officielle du développpement de kernel et kernel object pour Linux : https://www.kernel.org/doc/html/latest

Exemple pour commencer : https://www.kernel.org/doc/html/latest/core-api/printk-basics.html

Le livre Professional Linux Kernel Architecure qui contient deux chapitres entiers sur les périphériques (p 391) et les modules (p. 473).

Quelques liens qui peuvent vous intéresser : https://linux-kernel-labs.github.io/refs/heads/master/labs/kernel_modules.html#exercises

https://linux-kernel-labs.github.io/refs/heads/master/labs/kernel_modules.html#extra-exercises

https://tldp.org/LDP/lkmpg/2.6/html/index.html

https://github.com/dwmkerr/linux-kernel-module#sample-2-babel