Système d'exploitation – TD5
Module noyau et fichiers spéciaux

Les sources pour les parties 2 et 3 sont disponibles sur la page du cours.

Précautions

Tout ce TD nécessite des droits administrateur (root) pour modifier le noyau Linux, il doit donc être effectué dans une machine (virtuelle) adaptée. Des images de machine virtuelle pour KVM et VirtualBox ont été préconfigurées [1].

Vous pouvez éventuellement utiliser directement votre machine personnelle. Mais vérifiez bien votre code avant de charger vos modules car un crash noyau peut théoriquement avoir des conséquences graves sur votre installation (corruption disque dans le pire des cas).

Dans tous les cas, il est préférable de rebooter la machine crashée à chaque fois qu'un problème se produit dans votre module noyau. Si vous utilisez une machine virtuelle, il est inutile de rebooter la machine hôte.

1 Modules noyau

1.a) Listez la liste des modules avec la commande `lsmod`.

Chargez le module `uio` (ou un autre quelconque pas encore chargé) avec
`sudo /sbin/insmod /lib/modules/3.16.0-4-amd64/kernel/drivers/uio/uio.ko`
Observez son apparition dans `lsmod`. Déchargez-le et observez sa disparition dans `lsmod`.

1.b) Observez le compteur de références et les dépendances des différents modules dans les colonnes de droite de la sortie de `lsmod`.

Trouvez dans la sortie de `lsmod` un module A qui n'a aucune référence ni dépendance, et un autre module B qui a une seule référence et qui a A dans sa liste de dépendance.

Essayez de décharger B avec `rmmod <nom>`. Déchargez ensuite A et observez l'évolution de la sortie de `lsmod`. Essayez à nouveau de décharger B.

Dans quel sens sont en fait indiquées les dépendances et quelles sont leur lien avec le compteur de références ?

1.c) Dans la vraie vie, on a jamais besoin de préciser le chemin complet, ni même les dépendances, car on utilise `modprobe` (qui appelle `insmod` ou `rmmod` en interne).

Lancez `/sbin/modinfo <name>` sur des modules avec et sans dépendances (par exemple les modules A et B de la question précédente) pour observer leur chemin d'installation et leurs dépendances.

Utilisez `modprobe <name>` pour charger un module avec ses dépendances d'un seul coup (par exemple le module A).

1. Voir la page `goglin/teaching/Systeme/VM.xhtml` à coté de la page du cours.

2 Création d'un module noyau

2.a) Compilez les modules fournis avec `make` puis chargez le module `hello.ko` et vérifiez avec `lsmod` que cela a fonctionné.
Si `sudo insmod hello.ko` échoue avec l'erreur *Permission denied*, passez root avec `sudo su` avant de faire `insmod hello.ko`.

2.b) Observez le code de `hello.c` et notez qu'il demande au noyau d'afficher un message au chargement et au déchargement. Observez ces messages (ainsi que tous les autres messages du noyau) en lançant `dmesg | less`.

2.c) Ajoutez d'autres messages dans `hello.c`, recompilez et rechargez-le pour confirmer que vos changements apparaissent dans `dmesg`.

3 Fichiers spéciaux

3.a) Chargez le module `special.ko` (compilé en même temps que `hello.ko`). Il va créer un fichier spécial `/dev/special` Comparez ce fichier avec les informations données par `modinfo`.

Faites un `cat /dev/special` et expliquez ce que vous observez.

Faites ensuite un `echo -n toto > /dev/special` (l'option `-n` évite le retour à la ligne à la fin de `toto`) suivi d'un `cat /dev/special` et expliquez à nouveau.

Dans tous ces tests, on pourra suivre l'évolutions des opérations dans `dmesg`.

3.b) Modifiez le module `special.c` pour qu'il gère réellement la lecture et l'écriture dans le tampon `buffer`. On veillera bien à ne pas accéder trop loin (respecter `LENGTH`), à retourner le bon nombre de caractères lus ou écrits, et à mettre à jour la valeur du pointeur d'avancement dans le fichier `*offp`.

Créez un petit fichier texte, copiez le dans `/dev/special` puis vérifiez le contenu de ce dernier.

Essayez à nouveau avec un fichier texte long (plus long que `LENGTH`) et vérifiez qu'il est correctement tronqué.

On pourra utiliser `dd if=/dev/special of=monfichier bs=100 count=20` pour vérifier que la lecture par morceaux fonctionne correctement. Et `dd if=monfichier of=/dev/special bs=100 count=20` pour l'écriture par morceaux.

3.c) Modifiez le module pour que chaque caractère lu soit incrémenté de 1. Le buffer noyau reste inchangé, mais on lit `bcd` s'il contient réellement `abc`. Quelle opération plus utile pourrait-on imaginer d'appliquer à la place de cette incrémentation ?

3.d) Modifiez le module `special.c` pour que chaque caractère dans le tampon se comporte individuellement de la façon suivante : initialisation au caractère `'0'`, une écriture incrémente le caractère correspondant dans le tableau (et ignore la valeur écrite), une lecture retourne le caractère stocké dans le tableau puis le remet à `'0'`.

Exemple avec un tableau de taille 4 :

```
Initialisation              Tampon = 0000
Ecriture de abc au début             1110
Ecriture de ab à l'offset 2          1121
Ecriture de abcd au début            2232
Lecture de 1 caractère au début      0232      retourne 2
Lecture de 4 caractères au début     0000      retourne 0232
```

Vérifiez que cela permet effectivement de créer un fichier *spécial* dont le comportement n'a rien à voir avec un fichier normal. Quel utilisation pourrait-on avoir de ce fichier ?

3.e) Modifiez le module `special.c` pour pouvoir ajuster dynamiquement la longueur du buffer par un `ioctl`. On veillera à bien se protéger des accès concurrents pendant une modification de la taille en utilisant le mutex.