Programmation système – TP1

Objectifs du TP :

1. Charger et décharger un module noyau ;
2. Allouer de la mémoire dans le noyau.

1 Les modules noyaux

1.1 Code source

Le code source d'un module est divisé en une fonction d'initialisation et une fonction de terminaison.

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>

int mymodule_init(void)
{
  ...
  if (error)
    return -ENOMEM; /* erreur, pas assez de mémoire */

  printk("J'affiche que je suis chargé\n");

  return 0; /* succès */
}

void mymodule_exit(void)
{
  ...

  printk("J'affiche que je suis déchargé\n");
}

module_init(mymodule_init);
module_exit(mymodule_exit);
```

La fonction `printk` permet d'afficher un message. Elle fonctionne de la même façon que `printf` en espace utilisateur. Cependant, les messages ne sont pas affichés sur la console normale. Ils sont regroupés dans les messages du noyau, qu'on peut voir par la commande `dmesg` ou à la fin de `/var/log/messages`.

1.2 Chargement et déchargement

Après avoir compilé le module (en utilisant le polycopié correspondant), il faut l'insérer dans le noyau. Pour les noyaux 2.4, il faut insérer `mymodule.o`. Pour les 2.6, `mymodule.ko`.

```
insmod mymodule.ko
rmmod mymodule
```

La commande `modprobe` permet de charger/décharger un module qui a été installé (dans `/lib/modules`) en tenant compte de ses dépendances de symboles.

2 Allocation mémoire dans le noyau

L'allocation de *petites* zones de mémoire utilise les routines `kmalloc` et `kfree` qui fonctionnent comme `malloc` et `free` en espace utilisateur.

```
#include <linux/slab.h>
void * kmalloc(size_t size, int flags);
void kfree(void * ptr);
```

Le second argument de `kmalloc` définit ce que l'allocateur du noyau peut faire pour allouer cette mémoire. En général, on utilise `GFP_KERNEL` pour lui signifier qu'il peut tout faire pour y arriver (bloquer, faire des entrées-sorties, swapper d'autres pages, ...).

La mémoire allouée par `kmalloc` dans un module doit absolument être libérée. Il n'y a pas de libération automatiquement en cas de terminaison du processus comme en espace utilisateur puisqu'un module n'est pas un processus !