Université de Rouen Normandie

Master 1 Informatique
Parcours SSI

Programmation Noyau Linux : TP n°5

Périphériques en mode bloc

Richard DUFOUR
Numéro étudiant : 22108382

Professeur : Y. Guesnet

2022 — 2023

1/ On doit implémenter un module permettant de gérer un périphérique en mode bloc qui transfert toutes les opérations sur les blocs dans un fichier situé dans le répertoire /tmp. Pour faire ceci nous allons nous aider de la base du code vu en cours dans blk_dev_exemple, l'exemple my_ramdisk_mq_wq utilisant un système multi queues et un worker permettant de gèrer le périphérique.

Les sources du TP sont soit sur universitice soit sur le repertoire github suivant : https://github.com/RichardDufour/ProgNoyau

En testant de compiler l'exemple de base nous avons une erreur au niveau de la fonction mrd_do_bvec, il faut modifier le type de req_op est mettre un unsigned int à la place de enum, cela est du à la version du noyau, les sources données fonctionnent bien du noyau 5.15 au noyau 5.19 étant donné que j'ai testé sur les deux.
Toutes les opérations sur les blocs sont transférées dans un fichier situé dans le répertoire /tmp et dont le nom est fixé par un paramètre du module. On ajoute dans un paramètre comme ceci :

```
static char *filename = "/tmp/blk_periph";
module_param(filename, charp, 0444);
MODULE_PARM_DESC(filename, "Name of the tmp file use");
```

Puis dans les fonctions copy_to_blk et copy_from_blk nous gérons le fichier qui est /tmp/periph_blk avec les fonctions kernel_write dans copy_to_blk et kernel_read dans copy_from_blk.

2/ Nous pouvons tester un peu notre module (script install.sh dans les sources) :
Pour suivre l'avancement du périphique et du fichier nous pouvons suivre le fichier de syslog grâce à la commande :

```
sudo tail -f /var/log/syslog
```

Nous testons les choses suivantes :

- Nous insérons le module pour créer le périphérique et le fichier :

```
sudo insmod periph_blk.ko
```

- Nous créons une nouvelle partition sur le périphérique :

```
sudo fdisk /dev/periph_blk
```

- Nous formatons la partition :

```
sudo mkfs.ext4 /dev/periph_blk1
```

- Nous montons la partition sur le dossier dans le projet :

```
sudo mount /dev/periph_blk1 periph_blk/
```

Nous pouvons voir que toutes ces opérations fonctionnent bien et nous avons bien le système de fichier dans le dossier periph_blk étant donné que nous avons le dossier lost+found.

3/ Nous avons les deux fonctions encrypt et decrypt pour un chiffrement à décalage classique comme ceci :

```
// Encryption function
void encrypt_buffer(char *buf, int key)
{
    int i;
        size_t len = strlen(buf);
    for (i = 0; i < len; i++) {
        if (buf[i] >= 'a' && buf[i] <= 'z') {
            buf[i] = 'a' + ((buf[i] - 'a' + key) % 26);
        } else if (buf[i] >= 'A' && buf[i] <= 'Z') {
            buf[i] = 'A' + ((buf[i] - 'A' + key) % 26);
        }
    }
}

// Decryption function
void decrypt_buffer(char *buf, int key)
{
    int i;
        size_t len = strlen(buf);
    for (i = 0; i < len; i++) {
      if (buf[i] >= 'a' && buf[i] <= 'z') {
          buf[i] = 'a' + ((buf[i] - 'a' - key + 26) % 26);
      } else if (buf[i] >= 'A' && buf[i] <= 'Z') {
          buf[i] = 'A' + ((buf[i] - 'A' - key + 26) % 26);
      }
    }
}
```

Nous pouvons donc utiliser ces fonctions pour chiffrer et déchiffrer les informations contenus dans le fichier.

On ajoute un paramètre pour la clé de chiffrement comme ceci :

```
static int private_key = 3;
module_param(private_key, int, 0444);
MODULE_PARM_DESC(private_key, "Private key for encryption")
```

Installation

1) Cloner le dépôt avec :

```
git clone https://github.com/RichardDufour/ProgNoyau.git.
```

2) Se rendre dans le dossier du projet avec : cd ProgNoyau.
3) Compiler le projet avec : make.

Vous pouvez supprimer les fichiers créés avec : make clean

Utilisation

Installation du périphérique et test de toutes les fonctionnalitées avec le module :

```
./install.sh
```

Reset de tout (sauf le fichier /tmp/periph_blk) :

```
./reset.sh
```

Pour insérer le module créé dans le kernel, utilisez la commande suivante :

```
sudo insmod periph_blk.ko
```

Pour supprimer le module dans le kernel, utilisez la commande suivante :

```
sudo rmmod periph_blk.ko
```