S. Lavirotte
Polytech'Nice – Sophia
SI5 - Master 2 IFI
2016-2017

TD n° 5
Systèmes de Fichiers: /proc et VFS

1 But du TD

Le but de l'exercice est de réaliser un nouveau service de `/proc`. Ce service est un service global, donc attaché directement à la racine de `/proc`. Nous le nommerons `statfs` car son rôle est de donner accès à des informations sur les systèmes de fichiers montés. Plutôt qu'un long discours, en voici un exemple d'utilisation:

```
emu# cd /proc
emu# cat statfs
[00:15] name: tmpfs mounted on: */rw (rw)
        File system type: tmpfs
        Block size: 4096
[00:14] name: usbfs mounted on: */usb (rw)
        File system type: usbfs
        Block size: 4096
[00:18] name: //tweedy/Xchange mounted on: */win (rw)
        File system type: cifs
        Block size: 16384
[03:65] name: /dev/hdb1 mounted on: */work (rw)
        File system type: ext3
        Block size: 4096
[00:01] name: rootfs mounted on: / (rw)
        File system type: rootfs
        Block size: 4096
[00:00] name: none mounted on: */sys (rw)
        File system type: sysfs
        Block size: 4096
emu#
```

2 Création d'un module gérant une entrée de /proc

Pour résoudre ce problème, on commencera par créer un module qui ajoutera/enlèvera une entrée dans `/proc`. Cette entrée `/proc/statfs` sera mise en place de la manière suivante:

- Créer un module "coquille vide": commencez par créer la structure de votre module et le `Makefile` qui permettra de le compiler.
- Créer l'entrée dans `/proc` au démarrage du module:
  - Trouvez les méthodes qui sont utilisées pour créer les entrées dans `/proc` suivant leur type (au moins 6 méthodes sont utilisées dans les sources du noyau). Consultez le répertoire `/proc` pour trouver les fichiers déclarés. Puis trouverez les informations pertinentes pour la création de ces entrées dans les fichiers `fs/proc/root.c` et `fs/proc/generic.c`. Vous pouvez aussi plusieurs exemples d'utilisation des méthodes dans `fs/proc/*`.
    1. `create_proc_read_entry`: utilisé dans le code du noyau, mais pas exporté
    2. `proc_create`: utilisé dans le code du noyau, mais pas exporté
    3. `proc_symlink`: utilisé pour créer un lien symbolique, exporté
    4. `proc_mkdir`: utilisé pour créer un dossier, exporté
    5. `proc_create_data`: utilisé pour passer des arguments, exporté
    6. `create_proc_entry`: exporté mais plus utilisé dans les sources du noyau
  - Déterminez celle qui est adaptée à votre cas.
- Supprimer l'entrée dans `/proc` au déchargement du module: trouvez la méthode permettant de supprimer une entrée dans `/proc`.

TD n° 5
Systèmes de Fichiers: /proc et VFS

3 Accès aux informations du Virtual File System VFS

- Récupérer un point d'accès à la racine:
  - Pour cela vous commencerez par créer un pointeur sur une structure de type `vfsmount`.

```
static struct vfsmount * root_mnt;
```

  - Pour l'initialiser, vous utiliserez l'interface `linux/namei.h` ainsi que la fonction `kern_path`. Vous veillerez à bien inclure les `.h` nécessaires et vous vérifierez la valeur de retour de cet appel.

```
struct path p;
kern_path("/", LOOKUP_FOLLOW | LOOKUP_DIRECTORY, &p);
```

- Connecter la fonction d'accès en lecture sur le fichier créé dans `/proc`: Ajoutez la fonction `statfs_read` comme callback sur l'accès au fichier (voir dans `linux/proc_fs.h`)
- Ajouter le code de la fonction `statfs_read`: il ne vous reste qu'à ajouter le code nécessaire pour afficher l'ensemble des informations telles qu'affichée dans l'exemple au début de ce TD. Vous aurez besoin pour cela d'étudier les structures `vfsmount` (`linux/mount.h`) et `super_block` (`linux/fs.h`) pour récupérer les bonnes données.

Il ne vous reste plus qu'à utiliser votre module !