PNL - Projet 2019/2020

Le système de fichiers le plus classe du monde

Léo Check

Tarik Atlaoui

Max Eliet

05 Juin 2020

1 État d'avancement

1.1 Liste des fonctionnalités implémentées et fonctionnelles

Les fonctionnalitées implémentées sont les suivantes :
-Rotation du système de fichiers
-Politique de suppression de fichiers
-Interaction user / fs

2 Structures utilisées

Afin de réaliser les différentes fonctionalités, les structures utilisées sont les suivantes :

```
// pour les relations entre fichier et dossiers
struct dentry;
// pour obtenir les metadata d'un fichier
struct inode;
// manipuler les fichiers, notamment les supprimer
const struct inode_operations ouichefs_inode_ops;
// notre propre structure, contenant les politiques de suppression
struct ouichefs_politic ouichefs_politic;
```

3 Rotation du système de fichiers

3.1 Implémentation

La rotation de fichiers se réalise dans ces deux cas :

1. Un fichier est créé par l'utilisateur dans un répertoire contenant 128 fichiers et/ou répertoire.
2. Soit X un entier compris entre 0 et 100 : il reste X% blocs restant dans le superblock de la partition.

La résolution du problème passe par l'implémentation des 2 fonctions suivantes (situées dans le fichier `ex1.c`) :

```
int remove_LRU_file_of_dir(struct dentry *dir, int nbFiles);
int remove_lru_file(struct dentry *root);
```

La première fonction permet la rotation de fichiers dans le premier scénario indiqué ci-dessus. En prenant un paramètre le dentry correspondant au répertoire du fichier nouvellement créé et son nombre de fichiers, elle cherche le fichier qui a été utilisé le moins récemment, et le supprime du répertoire en utilisant la fonction unlink de la variable `ouichefs_inode_ops` et retourne sa valeur de retour. Par conséquent, lorsqu'elle ne trouve pas de fichiers à supprimé, une erreur est retournée. Plus particulièrement, lorsqu'un répertoire contient 128 sous répertoires, nous choisissons de ne pas en supprimer pour des raisons de sûreté (répertoires sensibles, répertoire nécessaire au bon fonctionnement du kernel, etc...).

Dans le cas où il reste X% blocs, la deuxième fonction est appelée. Elle prend en entrée le dentry racine de la partition, mais effectue une recherche récursive dans tous les répertoires de la partition pour rechercher le fichier utilisé. La suppression et la valeur retournée est faite de la même manière que dans la première fonction.

Nous créons une structure `ouichefs_politic` dans le fichier `ex2_2.c`, qui contiendra 2 pointeurs vers les 2 fonctions. Cela permettra de modifier les politiques de suppression dans les 2 scénarios par l'intermédiaire d'un module. Elle est par conséquent exportée à l'aide de la macro `EXPORT_SYMBOL` afin qu'elle soit accessible.

4 Politique de suppression de fichiers

4.1 Implémentation

Par défaut la politique d'éviction est donc de supprimer le fichier le moins récemment utilisé, la comparaison se fait au niveau des inodes en comparant leurs attributs `"i_mtime.tv_sec"` .
On souhaite pouvoir modifier cette politique d'éviction par insertion de module. Par conséquent, il était nécessaire que le développeur du module ait accès à la l'attribut `unlink` de la variable `ouichefs_inode_ops`. C'est pourquoi nous utilisons la macro `EXPORT_SYMBOL` sur cette dernière. Un première idée serait de choisir le fichier de plus grande taille lorsque X% sont restants dans une partition, en comparant donc le champ `i_size` de la structure inode.

Pour effectuer cette modification, on insère le module `mod_remove_biger` qui va changer la fonction pointée par le champ `clear_a_file` de la structure `ouichefs_politic` vers une nouvelle fonction `remove_the_bigger_file`, parcourant l'arborescende de fichiers pour en supprimer le plus grand.
Pour retourner à la politique originale, le développeur du module doit s'assurer de garder en mémoire l'ancienne politique dans un pointeur de fonction, et la restaurer plus tard, lorsque le module est retiré par exemple.
On peut aussi utiliser le module `mod_does_nothing` qui remplacera la suppression d'un fichier par un affichage à des fins de debug.

5 Interaction user / fs

5.1 Implémentation

Nos fonctions ont besoin d'un dentry en paramètre pour réaliser les suppressions. C'est pourquoi nous choissisons le sysfs pour permettre à l'utilisateur de les invoquer manuellement, en respectant un certain protocole :

```
echo -n <function> <dir> > /sys/kernel/ouichefs/ouichefs
```

où `<function> = {"clear_a_file", "clear_a_file_in_dir"}` et `<dir>` doit correspondre à un chemin vers un répertoire d'une partition ouichefs. Ainsi, grâce à la fonction `filp_open()`, le sysfs peut récupérer le file, puis le dentry du répertoire cible. Cette implémentation permet donc à l'utilisateur d'avoir plusieurs partition ouichefs, et indiquer dans quel partition il souhaite déclencher une suppression. Cependant, nous devons faire l'hypothèse que l'utilisateur respecte le protocole. Ce sysfs est instancié dans le fichier `fs.c`.