ISTY-IATIC-3

Projet-Système - Dossier Momonga FS

Mathieu BRETTE Pablo BOURDELAS
Sébastien POIROT Guillaume RYCKAERT

22 juin 2016

Introduction

Le but de ce projet est de créer un Système de fichiers(SGF) en C, en implémentant des primitives, un shell, ainsi que la sauvegarde/recharge du disque.

Nous avons donc choisi de créer un SGF travaillant sur un disque virtuel, stocké sous la forme d'un fichier nommé vdisk. Le SGF devra être capable de créer un disque virtuel, de créer et supprimer des fichiers et des dossiers, ainsi que d'afficher leur contenu.

Structure du disque

Le disque est composé des différentes structures suivantes :

— Le SuperBlock
— Les inodes
— L'Inode Bitmap
— La Block Bitmap
— Les Data

Le SuperBlock

Le SuperBlock contient les informations essentielles du disque, comme par exemple :

— Le nombre total d'inodes sur le disque
— Le nombre total de blocks de données sur le disque
— Le nombre de blocks libres
— Le nombre d'inodes libres
— La taille des blocks (en bytes)
— Ainsi que le numéro du premier block de données

Le superblock est chargé en même temps que le disque, dans le cas de la création d'un disque, il est automatiquement crée. Nous acons choisi de déclarer le SuperBlock en global, car il est utilisé dans de nombreuses fonction du SGF.

Les inodes

Chaque inode représente un fichier/dossier sur le disque. L'inode fournit les informations suivantes :

— La taille des données du fichier. [1]
— Une valeur représentant les droits d'acces, ainsi que la nature(fichier/dossier).
— Un tableau de 15 numéros de blocks, les 12 premiers représentent les blocks contenant les données du dossier/fichier.Le 13ème est le numéro de block indirect : ce block contient des numéros de blocks appartenant au fichier.De même pour le 14ème block qui est double indirect, et pour le 15ème, qui est triple indirect.
— Un nombre représentant la dernière fois que le fichier a été modifié. [2]
— Le nombre de liens symboliques se référrant a cet inode.

1. Pour les dossiers, voir plus bas.
2. Non implémenté

Chaque fichier/dossier a son inode. Dans le cas d'un dossier, les données stockées sont une liste de numéros d'inodes, ainsi que le nom du fichier/dossier qu'ils représentent,chaque couple ecrit sur une nouvelle ligne(spéaré par un \n). Dans ce cas, la taille des données inscrite dans l'inode est la taille des données précedemment décrites.

Les bitmaps

Il ya deux bitmaps dans le disque, une pour marquer les blocks utilisés, l'autre pour marquer les inodes utilisés. Leur taille est égale au nombre d'inode/blocks divisée par 8. [3]

Les données

Les données sont stockeés par blocks de taille fixe, définie a la création du disque virtuel. En écriture, le système remplit toujours les premiers blocs libres.

Fonctions implémentées

Le shell de notre système ne comprend que 4 fonctions : cat,ls, et assert, ainsi que mkdir.

cat

Cette fonction cherche tout simplement l'inode correspondant au chemin passé en paramètre, vérifie qu'il correspond bien a un fichier, puis affiche son contenu jusqu'a ce qu'il aie lu toute la taille du fichier.

ls

ls charge l'inode corresondant, vérifie que c'est un dossier, puis lit les blocks correspondants afin d'afficher le contenu de ce dernier.

assert

assert prend pour arguments un chemin, ainsi qu'une chaîne de caractères, le programme ajoute ensuite la chaine de caractères a la fin du fichier.

3. 0 ou 1 pour utilisé ou pas, 1 inode ou fichier par bit, 8 bits dans un octet.

mkdir

Cette fonction permet de créer un nouveau dossier a l'endroit donné.

Répartiton du travail

La répartition du travail a été la suivante :Pablo a travaillé sur les structures de disque, de superblock et d'inode,ainsi que sur les primitves système, Guillaume a travaillé le shell ainsi que les primitves système, et Mathieu et Sébastien ont travaillé sur les primitves système.

Difficultés Rencontrées

La structure choisie nous semblait complète et plus fidèle a un modèle réel, cependant nous avons sous-estimé le temps requis pour implémenter celle-ci.

Conclusion

Ce projet nous a permis d'apprécier toute la complexité d'un système de fichiers, ainsi que de son implémentation. Pour aller plus loin, nous pourrions developper rm, ainsi que touch, ou encore cp et mv.