SHAFIE Ichem
BADUEL Thomas

FILESYSTEM

I - Installation

Possibles librairies à installer : (compile sans besoin d'installer quoi que ce soit avec les ordinateurs de l'université)

```
$> sudo apt-get install libncurses5-dev
```

- Extraire le projet.
- Une fois le projet extrait, entrez dans le dossier extrait et tapez :

```
$> make
```

Pour éxécuter le programme :

```
$> ./filesystem
```
La commande vous crée par défaut un filesystem sous le nom de mem.img d'une taille de 400Mo

```
$> ./filesystem create <nom du fs> <taille en Mo>
```
La commande vous crée un filesystem sous le nom <nom du fs> de taille <taille en Mo>

```
$> ./filesystem read <nom du fs>
```
Reprends un filesystem crée au préalable sous le nom <nom du fs> (ex: Vous créez un filesystem et ajoutez 3 dossiers dedans, vous quittez le programme, la commande read vous permettra de reprendre ou vous en étiez dans ce filesystem.)

II – Commandes disponibles

blocks
```
$> blocks
```
Permet de recenser les blocks utilisés par le filesystem.

add
```
$> add <nom du fichier>
```
Permet d'ajouter un fichier depuis le filesystem extérieur dans le filesystem intérieur (fichier placé directement dans le dossier courant).

SHAFIE Ichem
BADUEL Thomas

ls

```
$> ls <chemin>
$> ls -l <chemin>
```

Permet de lister les fichiers du dossier <chemin>, sans argument ls vous affiche le dossier courant.
L'option -l vous permet d'afficher plus de détails comme la date de création, le type du fichier (fichier ou dossier), la taille…
D'autres options étaient prévues mais nous n'avons pas eu le temps d'implémenter tout ce qui était voulu au départ. La date de modification bien qu'implémentée n'est pas affichée.

rm

```
$> rm <nom du fichier>
$> rm <nom du dossier>
```

Permet de supprimer un fichier ou un dossier. A noter que contrairement à son homologue de linux, il ne nécessite pas de flag particulier pour supprimer un dossier récursivement. (Possible implémentation future).

extend

```
$> extend <nom du fichier> <contenu>
```

Permet d'étendre un fichier <nom du fichier> avec <contenu> . A noter que vous disposez d'une édition de ligne complète identique à celle de bash (faite avec amour par nos soins grâce aux termcaps) donc vous pouvez écrire des commandes telles que :

```
$> extend toto.txt "je
dquote> m'appelle
dquote> toto"
```

Et vous aurez effectivement dans le fichier toto.txt les 3 lignes correspondantes.

cat

```
$> cat <nom du fichier>
```

Permet de lire le contenu d'un fichier du filesystem et de l'afficher sur la sortie standard.

exit

```
$> exit
```

Permet de quitter le programme.

rename

```
$> rename <nom du fichier> <nouveau nom du fichier>
```

Permet de renommer un fichier.

mkdir

```
$> mkdir <nom du dossier>
```

Permet de créer un dossier.

cd

```
$> cd <nom du dossier>
$> cd ..
```

Permet de se déplacer du dossier courant vers <nom du dossier>

SHAFIE Ichem
BADUEL Thomas

cp

```
$> cp <nom du fichier> <nom du dossier>
```

Permet de copier un fichier vers un dossier.
Malheureusement, nous n'avons pas eu le temps de gérer la copie de dossier entiers récursivement à la même manière que le rm. Donc il n'est possible que de copier un fichier vers un dossier.

```
$> cp toto.txt "dossier1/dossier2/../dossier3"
```

(la gestion des chemins n'existe pas avec toutes les commandes).

L'édition de ligne

L'édition de ligne permet plusieurs choses, notamment :
- Pouvoir se déplacer sur une ligne de commande.
- Effacer/Ajouter des caractères en milieu de ligne.
- Accéder à l'historique des commandes du filesystem. (flèche du haut)
- L'appairage des quotes et double quotes.

Il nous manque l'implémentation de la gestion des tabulations, celle-ci est réalisée (via arbre lexicographique vu en L2) et il ne reste plus qu'à lui donner son espace de recherche. (nos commandes et les fichiers du dossier au moment de l'appui sur tab)

III – Descriptif des choix

Notre structure filesystem se compose de blocks et d'inodes.
Le filesystem est divisé en x blocks de taille égale et un block ne peut pas contenir plus d'un inode.
Ainsi nos blocks sont au départ vide et nous pouvons les remplir de fichiers.

Chaque fichier possède un inode, cet inode contient :
- Le nom du fichier
- La date de création
- La date de modification
- La date de dernier accès
- Le type du fichier
- L'inode du dossier parent
- La taille du fichier

Pour accéder à cet inode, il nous faut seulement son index, ainsi on se réfère à un inode comme étant un index et aussi comme étant la structure liée à l'index (ce qui peut parfois sembler confus).

Ces informations sont stockées et écrites dans le filesystem dans ce que nous appellons un header d'une taille fixe. Cet header nous permettra ainsi de restituer les fichiers et les rendre à leur état d'origine à chaque lecture du filesystem.
Ce header est donc stocké au début du premier block d'un inode.
Pour se situer dans le filesystem, notre block contient :

SHAFIE Ichem
BADUEL Thomas

- La position du block en octet dans le filesystem.
- L’inode correspondant à ce block.
- Sa disponibilité.

Donc dans une situation avec un filesystem constitué de 100 blocks qui disposent eux d’une taille de 500 octets chacun.
Un fichier qui prend 1100 octets sera dispatché sur trois blocks, le premier contiendra le header et une partie du contenu du fichier, le deuxième contiendra une autre partie du contenu et le troisème contiendra la fin du contenu du fichier.
Ainsi, nos 3 blocks indiqueront que l’inode correspondant est l’inode du fichier ajouté, permettant ainsi de retrouver facilement des fichiers/inode et leur block correspondant.

Notre structure filesystem contient une variable nommée “i_currentfolder”, celle-ci crée tout le système de dossier.

Ces structures nous permettent de créer toutes les commandes bien plus facilement. Effectivement, créer un dossier va lui attribuer un inode, entrer dans le dossier va juste remplacer la variable i_currentfolder de notre filesystem et à chaque ls, on ne listera que les fichiers ayant pour parent notre i_currentfolder.
Il suffit de chercher l’index de l’inode que l’on désire à chaque commande, puis à traiter les deux. Une fois que l’on en vient au moment de l’écriture dans le fichier filesystem, il nous suffit de récupérer la position en octet du block et écrire le header et le contenu à partir de cette position là.
La disposition en block nous permet aussi une copie plus facile grâce à une structure bien plus carrée.