HEIG-VD | REDS Institut Reconfigurable and Embedded Digital Systems

Systèmes d'exploitation (SYE)

Profs Daniel Rossier, Fiorenzo Gamba, Marina Zapater
Assistants : David Truan, Lucas Elisei, Mattia Gallacchi

Systèmes de fichiers

lab08 (24.01.2022)

Objectifs de laboratoire

Ce laboratoire porte sur les systèmes de fichiers et les liens, et plus particulièrement sur le système de fichiers FAT qui est celui implémenté dans SO3.

Le but de la première partie est d'améliorer l'application « ls » afin qu'elle affiche plus d'informations pour chaque fichier trouvé. Puis la deuxième partie portera sur le développement d'un programme qui crée des liens symboliques.

Echéance

- Le laboratoire sera rendu au plus tard le **dimanche 30 janvier 2022, 23h59.**

Validation du laboratoire

Selon la méthode énoncée en début de semestre, **chaque fin de session** doit aboutir à l'exécution d'un script qui transmettra le travail fourni. Chaque laboratoire s'accompagne d'un script de rendu appelé *rendu.sh* qu'il est **nécessaire** d'appeler à la fin de chaque session :

```
reds@reds2021:~/sye/sye21_lab08$ ./rendu.sh session
```

Cette commande va générer un dossier *rendu* qui contiendra les différences avec le dépôt de base. Lorsque le laboratoire est terminé, il suffit **d'exécuter le script** en lui spécifiant **fin** comme paramètre :

```
reds@reds2021:~/sye/sye21_lab08$ ./rendu.sh fin
```

Cette commande va générer un dernier fichier de différences, puis archiver le dossier *rendu*. Il est demandé de **déposer cette archive** (*rendu.tar.gz*) **dans Moodle à la fin du laboratoire**.

Etape 1 - Mise à jour du dépôt (environnement)

La commande suivante permet de récupérer une mise à jour du dépôt pour la réalisation de ce laboratoire.

```
reds@reds2021:~/sye$ retrieve_lab sye21 lab08
```

Ceci va créer un dossier « **sye_lab08** » qui contiendra les fichiers nécessaires au laboratoire.

Etape 2 – Amélioration de l'application « ls »

Dans le système actuel, une première version de la commande « ***ls*** » est implémentée. Cette version affiche uniquement les fichiers présents (un par ligne) dans le dossier courant. De plus, dans le système de fichiers, chaque fichier est accompagné de métadonnées (*metadata*).

Le but de cette étape est de récupérer ces métadonnées, et d'afficher certaines informations lors de l'appel à la commande « *ls* ». Les informations à afficher sont les suivantes :

```
<type><attributs> | <date> | <taille> | <nom>
```

Voici un exemple d'utilisation :

```
so3% ls -l
-rwx | 2022-1-18 8:17 | 116.3K | cat.elf
drwx | 2022-1-18 8:17 | 0 | dev/
...
```

Le fichier peut être de type « régulier » (**-**) ou de type répertoire (**d**). Les attributs supportés pour un fichier sont *read* (**r**), *write* (**w**) et *execute* (**x**).

a) Ajouter à la commande « *ls* », le support de l'option « ***-l*** » qui permettra l'affichage des métadonnées de chaque fichier listé.

⇒ Afin de récupérer les métadonnées d'un fichier, utiliser la fonction « ***stat()*** ». Celle-ci prend comme arguments le nom de fichier ainsi que l'adresse d'une structure de type « *struct stat ** ». Cette structure a été simplifiée dans SO3, les champs disponibles peuvent être consultés dans le fichier « ***usr/libc/include/bits/stat.h*** ».

⇒ Le champ « ***st_mode*** » contient le type ainsi que les attributs d'accès. Pour la correspondance des bits, les détails se trouvent dans le fichier « ***usr/libc/include/sys/stat.h*** ».

⇒ La chaîne de caractères à afficher doit contenir le symbole correspondant aux attributs d'accès, ou « - » s'il n'est pas actif. Bien que le système de fichiers FAT ne supporte pas les bits d'attributs, et que ceux-ci sont tous actifs par défaut, il faut tout de même implémenter la comparaison dans un but d'apprentissage.

⇒ Concernant la date et l'heure, le champ « ***st_mtim*** » correspond au temps de la dernière modification apportée dans le fichier. La fonction « ***localtime()*** » permet de récupérer la date correspondante ; elle devra être convertie selon le format suivant :

```
<Année>-<mois>-<jour> <heure>:<minutes>
```

⇒ La date ne tient pas compte du fuseau horaire.

⇒ Finalement, le champ « ***st_size*** » donne la taille actuelle du fichier (nombre d'octets dans le fichier).

b) ***(bonus)*** Faire en sorte que la taille soit exprimée en K/M/etc.

Etape 3 – Création de liens symboliques avec FAT-32

Cette étape propose de réaliser une application « ***ln*** » qui permet de créer des liens symboliques (lien « *soft* »).

(Pour info, le système de fichiers FAT ne supporte pas les liens symboliques. En revanche, l'implémentation de FAT dans SO3 a été étendue pour supporter ce type de fichier).

Dans SO3, l'appel système « *sys_symlink()* » permet la création d'un lien symbolique (le premier argument est le descripteur du fichier correspondant au fichier auquel un lien symbolique sera créé, le second argument est le nom du fichier correspondant au lien symbolique).

a) Compléter le fichier « ***ln.c*** » afin de pouvoir créer un lien symbolique comme suit :

```
so3% ln lorem.txt symbolic_link
```

b) Valider le bon fonctionnement en créant un lien symbolique vers le fichier « *lorem.txt* », puis faire une lecture du fichier de type lien.

Maintenant que la possibilité de créer des liens symboliques est présente, il serait utile de pouvoir les visualiser avec la commande « ***ls*** ».

c) Modifier l'application « *ls* » afin de faire apparaître l'attribut « ***l*** » ainsi que le fichier cible, comme le montre l'exemple ci-dessous :

```
so3% ln lorem.txt symbolic_link
so3% ls –l
…
-rwx | 21-1-4 11:48 | 5.6K | lorem.txt
…
lrwx | 17-5-1 0:0 | 10 | symbolic_link -> lorem.txt
```

⇒ Le champ « *st_mode* » contient un attribut correspondant au lien symbolique.

⇒ Le nom du fichier cible est obtenu en « ouvrant » le fichier de type lien. Deux *flags* passés à la fonction « *open()* » permettent de ne pas suivre le lien et d'obtenir le chemin du fichier pointé par le lien symbolique (voir les pages *man*). Il est ensuite possible de lire le nom du fichier pointé.

⇒ La date pour les nouveaux fichiers créés est fausse car la gestion du temps (RTC) n'est pas encore supportée dans SO3.