mickael.hoerdt@hesge.ch 2016/2017

Mini-projet de programmation Système/Systèmes d'exploitation

Le système de fichier ext2

ext2 est un système de fichier qui a été conçu pour le système d'exploitation Linux par Remy Card dans les années 90. Il est encore utilisé de nos jours sur certains périphériques de stockage de type flash. Les structures de données de base utilisés par ext2 sont très proches de celles du système de fichier MINIX vu en cours. La différence principale de ext2 par rapport MINIX réside dans la présence de groupes de blocs, dans une taille d'inode et de bloc qui peut différer d'un volume à l'autre, ainsi que dans la structure de donnée en liste chainée des répertoires.

L'objectif de ce projet est d'implémenter un système de fichier ext2 accessible en lecture seule depuis un fichier image préalablement formaté au format ext2.

L'accès au fichier image contenant le système de fichier ext2 devra être implémenté en python en utilisant les primitives d'accès aux fichiers classiques.

La manipulation des structures de données stockées sur l'image récupérée se fera exclusivement en langage python **version 2**, par l'intermédiaire du canevas des classes python et des tests unitaires fournis avec le projet.

Pour simplifier les traitements, l'entièreté de la table des inodes du système de fichier sera chargée en mémoire dans une liste, donc l'index 0 ne contiendra qu'un inode vide. Pour des raisons d'efficacité, dans un système de fichier réel, seul un sous-ensemble des inodes est en mémoire à un instant donné.

Étape 1 : Manipulation locale des structures de données de base stockées sur l'image

Récupérer le canevas du projet sur :

https://hepia.infolibre.ch/syst-exploitation-2016-2017/ext2fs.tgz

- Compléter les méthodes `__init__` et `read_bloc` de la classe `bloc_device` du fichier `bloc_device.py`

- Compléter la méthode `__init__` de la classe `ext2` du fichier `fs.py`. Celle-ci devra en particulier initialiser le champs `superbloc` à partir de la classe `ext2_super_bloc`, initialiser une instance `bloc_device` avec la bonne taille de bloc, initialiser une liste de descripteur de groupes de blocs `ex2_bgroup_desc` et initialiser un champs `inode_map` de type `bitarray` à partir du premier bitmap d'inodes libres du premier groupe de bloc du volume. Elle devra aussi initialiser un champs `bloc_map` de même type à partir du bitmap des blocs de données libres du premier bloc de groupe sur le fichier image.

- L'initialisation de la classe `ext2` devra aussi remplir le champs `inodes_list` dans la méthode `__init__` pour qu'il contienne la liste de l'ensemble des inodes du fichier image à son ouverture.

- Compléter les méthodes `bmap()`, `lookup_entry()`, `namei()`, de la classe `ext2` d'après le

cours donné sur les système de fichiers. Ces méthodes devront fonctionner avec une taille de bloc et une taille d'inode variable.

Étape 2 : modèle d'API d'accès aux fichiers selon les appels systèmes de type UNIX

- Compléter la classe `ext2_file_api` pour qu'elle réponde correctement aux tests unitaires du canevas. Le fichier `ext2fuse.py` contient une application utilisant cette API pour permettre le mountage du système de fichier dans l'espace utilisateur et ainsi y exécuter n'importe quel programme effectuant des lectures ou des écritures. Son utilisation requiert l'installation du package `fusepy` et de la librairie fuse sur votre système.

Étape 3 : Récupération de contenu de fichiers effacés (uniquement pour les groupes de 3)

- A l'aide du canevas complété dans les étapes 1 et 2, écrire un outil permettant de récupérer le contenu de fichiers qui auraient été récemment effacés avec la commande « rm » sur le fichier image. Votre outil devra être assorti d'un script de démonstration/test.

Informations indispensables

La documentation sur le système ext2 située à l'adresse http://www.nongnu.org/ext2-doc/ext2.html est d'une très grande utilité pour comprendre comment est structuré le système de fichier ext2.

Les modules ou classes python suivants/tes vous seront utiles/indispensables pour compléter ce projet :

- `file` pour les opérations sur le fichier image, `open()` pour retourner un objet `file`
- `bitarray` pour gérer les bitmaps des inodes et des blocs de données libres/occupés.
- `struct` pour transformer des types binaires en type python et inversement.
- `hexdump` pour afficher une chaîne de bytes en hexadécimal.

Ces modules/classes sont tous documentés dans l'aide en ligne de python (voir la commande help(<nom_du_module/nom_de_la_classe>), après avoir été importée dans l'interpréteur avec la commande `import <nom_du_module>` sauf si il s'agit d'une classe du module `__builtin__`

Important : Pour tester/vérifier les étapes 1 et 2 de votre projet à la conformité de votre implémentation à travers les programmes de test unitaires `tester.py` et `tester2.py` sont fournis dans le canevas. L'évaluation de votre implémentation sera basée sur la conformité de votre code en utilisant le programme `tester.py` et `tester2.py` mais sur un jeu de données différent que celui fournit dans le canevas. Le projet à réaliser obligatoirement en **python 2.**

Travail à rendre

Le projet est réalisable par groupe de 2 ou 3 personnes. Pour les groupes de 2, les étapes 1 et 2 obligatoires alors que l'étape 3 est optionnelle. Pour les groupes de 3, l'étape 3 est obligatoire. Le projet devra impérativement être rendu avant le **21/6/2017** sous la forme d'une archive contenant le code complété du canevas à renvoyer par mail. Le projet comptera pour 50 % de la note finale de la matière.