Système d'Exploitation - N3, S6
TP 1 - Système de Fichiers

L'objectif est d'implémenter un module de gestion de système de fichiers.

Nous allons considérer un système de fichiers simpliste. Ce système est implémenté à base d'inodes et est organisé de la manière suivante :

— tout d'abord, il y a le nombre de blocs du système, sur un octet,
— puis le nombre d'inodes dans le système, sur un octet,
— puis un bitmap des blocs utilisés,
— puis un bitmap des inodes utilisés,
— puis le tableau des inodes du système,
— puis le tableau des blocs du système.

Chaque inode est sauvegardé sur 16 octets et contient :

— la taille du nom de fichier, sur un octet,
— le nom du fichier, dans un emplacement de huit octets,
— la taille du fichier en octets, sur un octet,
— six indices de bloc de données, sur un octet chacun.

Le système de fichiers complet est stocké dans un fichier (dont l'extension sera `.fs`). Chaque bloc contient 16 octets. Le nombre de blocs et d'inodes du système sont tous deux des multiples de 8.

Exercice 1.1 *Manipulation du système de fichiers*
Vous disposez d'un certain nombre de fichiers `correct-`x`.fs` contenant un système de fichiers correct. Vous pouvez utiliser l'outil `xxd`, par exemple, pour afficher en hexadécimal le contenu d'un fichier.

1. Écrivez un module **`sys_fic`** qui implémente ce système de fichier. On doit pouvoir l'enregistrer dans un fichier et le charger depuis un fichier (voir les fichiers `correct-`x`.fs` pour exemple). La structure détaillant le système de fichiers ne doit pas être visible publiquement.
2. Testez votre module en écrivant un programme qui lit le contenu du fichier contenant le système de fichiers, et qui affiche les informations de base (nombre total de blocs et nombre total d'inodes).
3. Améliorez votre programme pour qu'il affiche le nombre de inodes utilisés.
4. Améliorez votre programme pour qu'il affiche le nom de chaque fichier.
5. Améliorez votre programme pour qu'il affiche le contenu de chaque fichier (sous chaque fichier).

Exercice 1.2 *Appels Systèmes*
On va implémenter les fonctions systèmes de création, ouverture et lecture/écriture à l'aide de descripteurs de fichiers. Un descripteur de fichiers sera identifié par le système de fichiers,

l'inode du fichier associé, le numéro du bloc courant, le décalage dans le bloc courant et un cache de la taille d'un bloc (qui contient le contenu du bloc courant et dans lequel on lit/écrit d'abord si les données demandées se trouvent dans le bloc courant). On créera un module **descripteur**. Dans ce module on doit pouvoir : créer un descripteur, lire un bloc et écrire un bloc. Pour l'écriture d'un bloc, on considérera que l'écriture se fera dans le bloc du fichier directement.

1. Écrivez le module **descripteur**.
2. Écrivez un module **appel_sys** pour implémenter les fonctions de création, ouverture, lecture, écriture et positionnement du décalage. Attention : toute demande de lecture/écriture du module **appel_sys** doit passer par le descripteur.
3. Testez les deux modules en écrivant un programme qui permet de manipuler les systèmes de fichiers `correct-`x`.fs`.

Exercice 1.3 *Vérification du système*
On va maintenant ajouter au module **sys_fic** les fonctions qui permettent de vérifier/corriger le système de fichiers.

1. Vous disposez d'un certain nombre de fichiers `free-`x`.fs` contenant un système de fichiers dans lequel la table des blocs est incorrecte. En effet, certains blocs utilisés sont marqués comme libres. Écrivez une fonction qui corrige cette anomalie du système de fichiers. Testez votre fonction en écrivant un programme qui corrige un système de fichiers et qui affiche les informations du système de fichiers corrigé.
2. Vous disposez d'un certain nombre de fichiers `duplicate-`x`.fs` contenant un système de fichiers dans lequel la table des blocs est incorrecte. En effet, certains blocs sont utilisés par plusieurs fichiers. Écrivez une fonction qui corrige la duplication des blocs : les blocs utilisés par deux inodes doivent être dupliqués de manière à ce que chaque inode utilise son propre bloc, lorsque c'est possible sinon vous devez le signaler. Testez votre fonction en écrivant un programme qui corrige les blocs du système de fichiers, et affiche les informations du système de fichiers corrigé.
3. (Bonus) Vous disposez d'un certain nombre de fichiers `lost-`x`.fs` contenant un système de fichiers dans lequel la table des blocs est incorrecte. En effet, certains blocs sont marqués comme utilisés, mais n'appartiennent à aucun inode. Les blocs marqués utilisés mais affectés à aucun inode sont considérés comme des blocs perdus. Il faut considérer qu'il s'agit de blocs de données importants, mais pour lesquels l'inode a été perdu. On crée dans ce cas un fichier spécial par bloc perdu, contenant uniquement les données du bloc. Ces fichiers permettent à l'administrateur du système de fichiers de récupérer des fichiers aisément.

   Écrivez une fonction qui permet de faire cette correction. Testez votre fonction en écrivant un programme qui corrige les blocs du système de fichiers, et affiche les informations du système de fichiers corrigé.

Exercice 1.4* *Amélioration du système de fichiers*

1. Modifiez le système de fichiers pour qu'il prenne en compte les répertoires. Pour simplifier on va dire qu'un répertoire est un fichier spécial dont les données sont les adresses des inodes associées aux fichiers contenus dans le répertoire.

2. Modifiez la définition d'un inode pour que les données ne soient plus dans un espace contigu, mais sont chaînées dans une table à deux niveaux.
3. Modifiez la fonction d'écriture d'un bloc pour que l'écriture se fasse dans un cache et non directement dans le fichier en disque. Vous veillerez à ce qu'il existe une fonction système `sync` pour forcer l'écriture dans le disque. Vous devrez aussi gérer l'écriture de ce cache dans le disque lorsque l'on demande à accéder à un autre bloc (que celui qui se trouve dans le cache).