info507 : Systèmes d'exploitation
TD 4 : systèmes de fichiers

Pierre Hyvernat, François Boussion, Gérald Cavallini
Pierre.Hyvernat@univ-smb.fr
Francois.Boussion@univ-smb.fr
Gerald.Cavallini@univ-smb.fr

Question 1. Dans un système de fichiers, qu'est-ce qu'un inode ?

Exercice 1 : Inode et système de fichiers ext2/3

ext2 est un système de fichiers courant sous Linux, bien que maintenant souvent remplacé par ext3 ou ext4.

Chaque inode contient plusieurs champs indiquant les propriétés du fichier. Parmi ces champs, 15 d'entre eux sont des blocs pouvant être de deux types :
- des blocs d'adresses, qui contiennent des pointeurs vers d'autres blocs ;
- des blocs de données, qui contiennent les données du fichier.

Les 12 premiers champs (sur les 15) contiennent les adresses des 12 premiers blocs de données du fichier (à raison d'une adresse par bloc). Si les blocs sur lesquels pointent les 12 premiers champs sont suffisants pour contenir le fichier, les champs 13, 14 et 15 ne sont pas utilisés.

Dans le cas contraire, en plus des 12 premiers blocs, les blocs 13, 14 et 15 sont utilisés. Ces blocs fonctionnent selon un système d'indirection. Il existe trois niveaux d'indirection :
- la simple indirection, utilisée par le champ 13 ;
- la double indirection, utilisée par le champ 14 ;
- la triple indirection, utilisée par le champ 15.

Plus le niveau d'indirection est élevé, plus le nombre final de blocs de données sur lequel pointe le champ (13, 14 ou 15) sera élevé. Ce système permet donc aux fichiers d'avoir une taille considérable.

De manière concrète, chacun de ces trois champs pointe vers un bloc d'adresses, qui pourra pointer vers un ou plusieurs blocs d'adresses ou de données. En supposant que les blocs ont comme taille 4096 octets (4ko), et que chaque adresse (dans le cas d'un bloc d'adresses) est stockée sur 32 bits (4 octets), chaque bloc d'adresses en contiendra 1024.

Question 1. Représentez de manière graphique un inode avec les blocs de données correspondants.

Question 2. Que se passe-t-il quand un fichier est trop grand pour être contenu dans les 12 blocs de données adressés par les champs de 1 à 12 ? Quelle est la taille minimale d'un tel fichier ?

Un fichier peut-il occuper exactement 1037 blocs sur le disque ? 1038 ? 1039 ?

Question 3. Quelle est la taille maximale d'un fichier valide pour le système ext2 ?

Question 4. Quelle est la taille maximale d'un fichier valide pour un système ext2 *avec des blocs de 1ko* ?

Question 5. Quels avantages et inconvénients y a t'il à utiliser des blocs de taille 1ko ? Idem pour des blocs de taille 8ko ?

Exercice 2 : Répertoires en ext2

Un répertoire est traité comme un type d'inode particulier, où les 15 blocs de données sont utilisés pour stocker une *liste chaînée* des fichiers contenus dans le répertoire.

Chaque entrée de la liste indique :
- le nom du fichier,
- le numéro d'inode de ce fichier.

Question 1. Représentez graphiquement les inodes pour un système de fichiers qui contient les répertoires `/home`, `/usr`, `/usr/bin` et `/usr/lib`, ainsi que les fichiert `/usr/bin/awk` et `/usr/bin/free`.

Question 2. Pourquoi ne faut-il pas toujours libérer les blocs de données d'un fichier lors de la commande shell `rm ...` ?

Comment l'appel système qui supprime un fichier (`unlink`) peut-il savoir s'il doit libérer les blocs de données ?

Question 3. Pourquoi est-ce que le compteur de référence d'un répertoire vide vaut-il 2 ?

Question 4. Les étapes de la suppression définitive d'un fichier sont :
- libérer les blocs de données afin qu'ils puissent être réutilisés,
- supprimer l'inode dans la liste chainée du répertoire parent,
- marquer l'inode comme libre.

Que se passe-t'il en cas de coupure de courant avant la fin de l'opération ? Est-ce que changer l'ordre des étapes permet de corriger le problème ?

Exercice 3 : Journalisation

D'une manière générale, une opération sur un système de fichiers (création de fichier, suppression, déplacement, etc.) se décompose en deux étapes :
- le système met à jour le contenu du fichier,
- puis le système met à jour les méta-données du fichier, c'est-à-dire l'inode du fichier et éventuellement d'autres inodes, comme celle du répertoire parent.

Une opération qui échoue avant la fin laisse le système de fichiers dans un état incohérent : des outils existent pour le réparer (`fsck` par exemple sous GNU/Linux), mais il serait préférable de conserver la cohérence à tout instant.

Pour cela, on peut ajouter un journal : avant de mettre à jour le système de fichiers, le système d'exploitation indique dans le journal les mises à jour qu'il va réaliser. En cas de panne, il suffit de lire le journal pour terminer les mises à jour qui ont été interrompues.

En pratique, le journal est constitué d'une liste d'opérations, qui indique pour chaque opération :
- les mises à jour à appliquer aux méta-données,
- et l'état de l'opération :
  - prévue : aucune données n'a encore été écrite,
  - écriture des données en cours,
  - écriture des données terminée, mais les méta-données n'ont pas encore été mises à jour,
  - terminée : dans ce cas, on peut supprimer l'entrée dans le journal.

Question 1. Décrivez ce qui se passe en cas de panne :
- avant l'écriture des données,
- après l'écriture des données.

Question 2. Utiliser un journal présente tout de même deux inconvénients : imaginez lesquels.