330 Répertoires

INF3173
Principes des systèmes d'exploitation

Jean Privat

Université du Québec à Montréal

Hiver 2021

Répertoires

Associent noms de fichiers et inodes

- Rappel: le nom des fichiers n'est pas dans la table des inodes
- « données » d'un répertoire = liste d'entrées
- Chaque **entrée** associe un nom de fichier à un numéro d'inode
- Certains SF y dupliquent de l'information (type du fichier, etc.)

Exemple

inode 253 (répertoire) :

```
253 .
146 ..
540 ficelle
490 repondeur
```

inode 490 (répertoire) :

```
490 .
253 ..
679 fictif
831 fichtre
```

API POSIX (portable)

- Structure opaque `DIR *`
- `opendir(3)`, `readdir(3)`, `closedir(3)`

```
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <dirent.h>

int main(int argc, char **argv) {
  DIR *d = opendir(argv[1]);
  if (!d) { perror(argv[1]); exit(1); }
  struct dirent *de;
  while((de = readdir(d)))
    printf("%10li %s\n", de->d_ino, de->d_name);
  closedir(d);
  return 0;
}
```

Accès en vrai

- Les répertoires sont des fichiers spéciaux
- Contenu pas directement accessible à l'utilisateur

Question

- Peut-on utiliser `open(2)` et `read(2)` pour lire les répertoires ?
- Comment fonctionnent `opendir(3)` et `readdir(3)` en vrai ?

Chemin $\rightarrow$ inode en théorie

1. Découper le chemin en éléments $e_1/e_2/.../e_n$
2. Partir du répertoire de base i_k avec $k = 0$ (racine, courant, etc.)
3. Charger le contenu du répertoire i_k depuis l'espace de donnée du disque
4. Chercher dedans l'élément suivant e_{k+1}
5. Charger l'inode associé i_{k+1} depuis la table des inodes
6. Vérifier que i_{k+1} est bien un répertoire, les droits, etc.
7. Si besoin, continuer en 3 avec $k = k + 1$

- Beaucoup d'accès nécessaires au disque (au moins $2n$)
- À faire : droits, liens symboliques, points de montage
- Attention à la concurrence
  - Un processus résout un chemin
  - Pendant qu'un autre modifie les répertoires

Chemin → inode en pratique : cache

- `dentry` (*directory entry*) et `dcache` (*dentry cache*)

Représentation **globale** interne au SE

- Vue (partielle) en mémoire de la hiérarchie globale
- Associe une entrée à son inode et son système de fichiers
- Mise en cache au fur et à mesure
- Libération si la mémoire est demandée pour autre chose

Sert de cache

- Pas besoin de relire les répertoires sur le disque à chaque fois
- Sauf dans certains cas (ex. disques réseau)
  → validation et synchronisation

Efficace

- Accès rapide aux entrées : table de hachage
- Échec rapide : stocke entrées inexistantes (*negative dentry*)

Liens durs (*hard link*)

Définition

Des entrées de répertoires

- Avec un ou **plusieurs** noms
- Dans un ou **plusieurs** répertoires
- Qui référencent un **même** inode

$\rightarrow$ Le champ **nombre de liens durs** compte le nombre de références

Piège

- Les liens durs ne sont pas des liens « fichier $\rightarrow$ fichier »
- Mais des liens « entrée $\rightarrow$ inode »
- Appelé aussi « lien direct », « lien physique » ou juste « lien »

Manipulation des liens durs

Création de liens durs

- `ln(1)` et `link(2)`
- Pas de distinction entre l'original et le lien

$\rightarrow$ Les deux entrées désignent le **même** fichier (inode)

Suppression

- `rm(1)` et `unlink(2)`
- Décrémente le nombre de liens durs
- Si 0, le fichier (inode) est réellement supprimé
- Note: `creat(2)` et `unlink(2)` ne sont pas symétriques

Renommage et déplacement

- `mv(1)` et `rename(2)`
- Le nombre de liens durs reste inchangé
- Attention, seulement sur le même système de fichiers

Limites de liens durs

- Forcément sur le même système de fichiers
- Pas de liens durs entre répertoires
- Pas forcément l'effet voulu lors de l'écrasement de fichiers (perte d'identité)

Questions

- Comment la commande `mv(1)` sait déplacer entre systèmes de fichiers (alors que `rename(2)` ne sais pas faire) ?
- Pourquoi il existe une commande `cp(1)` mais pas d'appel système de copie ?
- Comment supprimer tous les liens durs d'un fichier ?
- Si on pouvait utiliser `link(2)` sur les répertoire, comment créer des répertoires détachés de la racine ?