Unix : les FICHIERS

Laure Danthony

1 Introduction

Un fichier UNIX est une suite d'octets (linéaire : pas de trous). Il a des droits que l'on peut modifier. Il possède des caractéristiques, comme son PID (numéro d'identification).

2 Création, écriture, lecture, fermeture, ...

- **Création ou ouverture** : `open(char *path, int mode, (int rights))`, par exemple `open("toto", O_CREAT|O_WRONLY)`. `open` renvoie le FD, ou -1 si il y a un plantage.

  Remarque 1 Le FD pointe sur une case des fichiers ouverts (mode, offset). Si on ouvre deux fois le même fichier, on pointe sur une autre case.

- **Fermeture** : `close(Fd)`

- **Ecriture** : `write(int Fd, char *data, int size)`, renvoie le nombre d'octets écrits (size ou moins), exemple `write(Fd,"coucou",6)`.

- **Lecture** : `read(int Fd, char *data, int size)`, renvoie `size`, ou moins, s'arrête avec 0 en fin de fichier.

  Remarque 2 Toutes ces fonctions renvoient -1 en cas d'erreur.

- **Position de lecture** : `lseek(int Fd, int position, int whence)` `whence` est la position du curseur au début du changement de position : `SEEK_SET`(début du fichier) ou `SEEK_CUR` (position courante) ou `SEEK_END` (fin du fichier), `position` peut être négative.

3 Entrées, sorties, buffer

- **Redirections** : le shell lit sur `stdin`, écrit sur `stdout` ; on peut rediriger la sortie standard avec `<` et la sortie standard avec `>`.
- **Primitives** *C* : `Fread`, `Fwrite` permettent une plus grande gestion (`fprintf`, flottants, ...), il y a un **buffer** (c'est-à-dire une table en mémoire tampon où il y a un offset, de temps en temps on écrit la totalité du buffer sur le disque, on gère ce tampon à l'aide de `setbuff`).
- La fonction `Fgets(char *buff, size_t max, FILE *f)` lit les caractères sur le fichier `f` et les écrit dans le tableau `buff`

4 Architecture et gestion des fichiers

- voir schéma récapitulatif.
- Un **I-nœud** est accessible par `stat(char *path, struct stat *buff)`, `stat` contient des infos (mode, ino, numéro disque, taille) sur le fichier.
- Le **buffer cache** (mémoire tampon) permet de réduire les appels disque. Les entités de base sont les blocs (numéro, numéro de disque, occupé/libre, dirty/not dirty) qui sont accessibles à partir d'une table de hachage qui pointe sur des listes chaînées. Les blocs libres sont aussi chaînés (de façon à ce que le plus vieux soit en tête de file).
- La fonction `get_bloc` fait la recherche d'un bloc.