1 ère année Master : RSD **Année : 2021/2022**

Cours : Programmation réseaux et systèmes

Chapitre : Système de Gestion des Fichiers.

Introduction

Un fichier désigne un ensemble d'informations stockées sur le disque. Le système de fichiers est la partie du système d'exploitation qui se charge de gérer les fichiers. La gestion consiste en la création (identification, allocation d'espace sur disque), la suppression, les accès en lecture et en écriture, le partage des fichiers et leur protection en contrôlant les accès.

Qu'est ce qu'un fichier ?

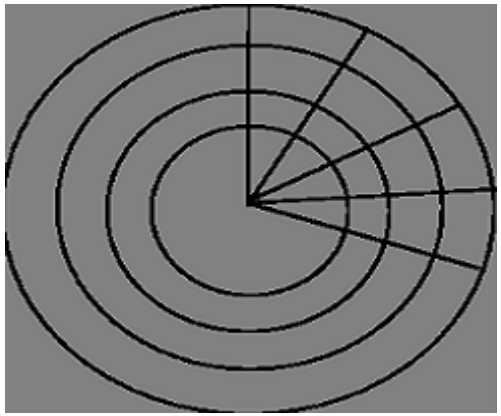
Pour le système d'exploitation, un fichier est une suite d'octets. Par contre, les utilisateurs peuvent donner des significations différentes au contenu d'un fichier (suites d'octets, suite d'enregistrements, arbre, etc.).

Chaque fichier est identifié par un nom auquel on associe un emplacement sur le disque (une référence) et possède un ensemble de propriétés : ses attributs.

Organisation du disque dur

Plateaux, cylindres, secteurs

Un disque dur possède plusieurs plateaux, chaque plateau possède deux faces, chaque face possède plusieurs secteurs et plusieurs cylindres (voir figure ci-dessous).



Organisation du disque dur

Le disque possède un secteur de boot, qui contient des informations sur les partitions bootables et le bootloader, qui permet de choisir le système sous lequel on souhaite booter.

Un disque est divisé en partitions (voir la figure au dessus). Les données sur les partitions (telles que les cylindre de début et de fin ou les types de partition) sont stockées dans une table des partitions. Le schéma d'organisation d'une partition UNIX est montré sur la figure 1.

Gérer les partitions et périphériques de stockage

L'outil fdisk permet de modifier les partitions sur un disque dur.

Par exemple, pour voir et modifier les partitions sur le disque dur SATA /dev/sda, on lance fdisk :

fdisk /dev/sda

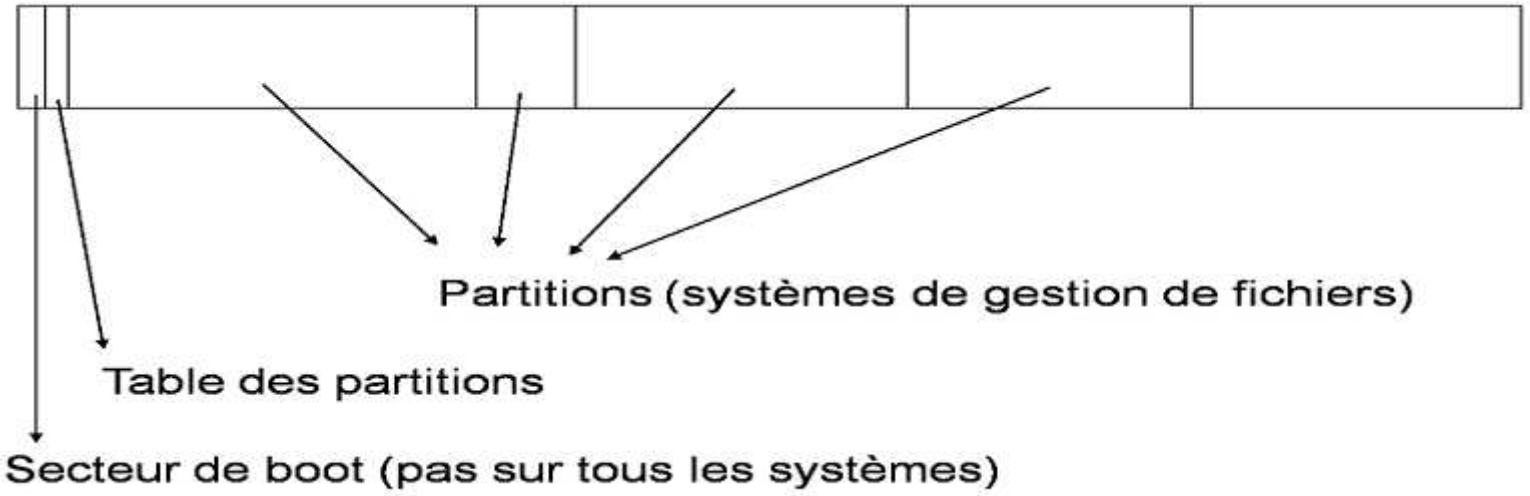


Figure1 : Organisation d'un disque du

Gérer les partitions et périphériques de stockage

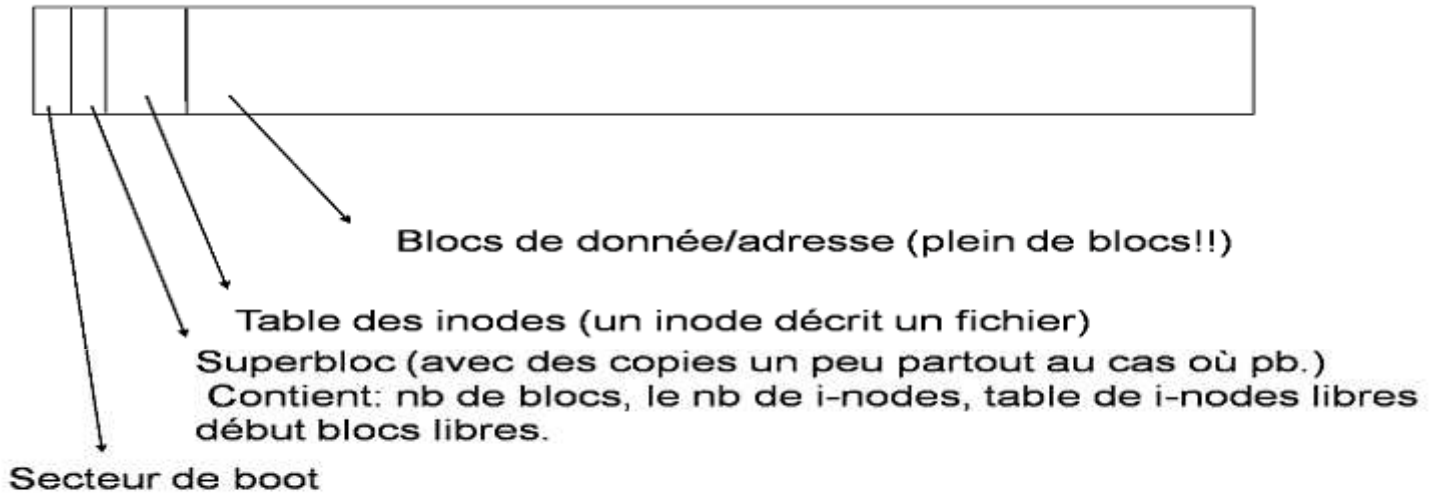


Figure 2: Organisation d'une partition

Fichiers, inodes et liens

Un inode identifie un fichier et décrit ses propritées, telles qu'in peut les vois par ls -l :

- données sur le propriétaire :UID et GID ;
- Droits d'accès ;
- Dates de création, de dernière modification, de dernier accès ;
- Nombre de fichier ayant cet inode (en cas de liens durs) ;
- Taille en octets ;
- Adresse d'un block de données.

Il existe dans un disque dur des liens, lorsque plusieurs noms de fichiers conduisent aux mêmes données.

Fichiers, inodes et liens

Ces liens sont de deux types :

On parle d'un lien dur lorsque deux noms de fichiers sont associés au même inode (les différents liens durs a exactement les mêmes propriétés mais des noms différents).

En particulier, l'adresse des données est la même dans tous les liens durs. La commande rm décrémente le nombre de liens durs. La suppression n'est effective que lorsque le nombre de lien durs (visible par ls -l ou stat) devient nul.

Fichiers, inodes et liens

Les liens durs sont créés par la commande ln.

-On parle d’un lien symbolique lorsque le block de données d’un fichier contient l’adresse du bloc de données d’un autre fichier. Les liens symboliques sont créés par la commande ln -s (optin -s).

Descripteurs de fichiers

Un descripteur de fichier est un entier qui identifie un fichier dans un programme C. Ne pas confondre un descripteur de fichier avec un pointeur de fichier. La fonction fdopen permet d'obtenir un pointeur de fichier à partir d'un descripteur.

Ouverture et création d'un fichier

La fonction open permet d'obtenir un descripteur defichier à partir du nom de fichier sur le disque, de la même façon que fopen permet d'obtenir un pointeur de fichier. Une grande différence est que la fonction open offre beaucoup plus d'options pour tester les permissions.

Descripteurs de fichiers

Le prototype de **la fonction open** est le suivant :

int open(const char *pathname, int flags, mode_t mode);

La fonction retourne une valeur strictement négative en cas d'erreur.

Le paramètre flags permet de préciser si, pour le programme, le fichier est en lecture seule (masque O_RDONLY), écriture seule (masque O_WRONLY), ou lecture-écriture (masque O_RDWR). Par un ou bit à bit (|), on peut aussi préciser si le fichierest ouvert en mode ajout (masque O_APPEND), ou si le fichier doit être écrasé (masque O_TRUNC) ou ouverten mode création (si le fichier n'existe pas il sera créé, masque O_CREAT).

Descripteurs de fichiers

Le prototype de la fonction openest le suivant :

int open(const char *pathname, int flags, mode_t mode);

La fonction retourne une valeur strictement négative en cas d’erreur.

Le paramètre flags permet de préciser si, pour le programme, le fichier est en lecture seule (masque O_RDONLY), écriture seule (masque O_WRONLY), ou lecture-écriture (masque O_RDWR).

Descripteurs de fichiers

Par un ou bit à bit (|), on peut aussi préciser si le fichier est ouvert en mode ajout (masque O_APPEND), ou si le fichier doit être écrasé (masque O_TRUNC) ou ouvert en mode création (si le fichier n'existe pas il sera créé, masque O_CREAT). Le mode permet de fixer les permissions lors de la création du fichier (le cas échéant), lorsque le paramètre flag est spécifié avec O_CREAT. Les permissions peuvent être définies par des chiffres octaux (7 pour rwx, 0 pour aucune permission) pour l'utilisateur, le groupe et les autres utilisateur.

Descripteurs de fichiers

exemple

```
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>
int main()
{
char c;
int in, out;
in = open(“file.in”, O_RDONLY);
out = open(“file.out”, O_WRONLY|O_CREAT, S_IRUSR|S_IWUSR);
while(read(in,&c,1) == 1)
write(out,&c,1); }
```

Descripteurs de fichiers

Lecture et écriture via un descripteur

Pour écrire des octets via descripteur de fichier, on utilise la **fonction write**:

ssize_t write(int descripteur1, const void *bloc, size_t taille);

```
#include <unistd.h>
#include <stdlib.h>
int main()
{
if ((write(1, “Here is some data\n”, 18)) != 18)
write(2, “A write error has occurred on file descriptor 1\n”,46);
exit(0);
}
```

Descripteurs de fichiers

Le fichier (ou tube ou socket...) doit être ouvert en écriture (options O_WRONLY, O_RDWR) la taille est le nombre d'octets qu'on souhaite écrire, et le bloc est un pointeur vers la mémoire contenant ces octets.

Pour lire des octets via un descripteur de fichiers, on utilise la **fonction read**:

ssize_t read(int descripteur0, void *bloc, size_t taille);

Descripteurs de fichiers

```
#include <unistd.h>
#include <stdlib.h>
int main()
{
char buffer[128];
int nread;
nread = read(0, buffer, 128);
if (nread == -1)
write(2, “A read error has occurred\n”, 26);
if ((write(1,buffer,nread)) != nread)
write(2, “A write error has occurred\n”,27);
exit(0);
}
```

Descripteurs de fichiers

Le fichier (ou tube ou socket...) doit être ouvert en lecture (options O_RDONLY, O_RDWR) On peut aussi utiliser fdopen pour obtenir un FILE* ce qui permetd'utiliser des fonctions plus haut niveau telles que fprintf et fscanf ou fread et fwrite.

Référence

Rémy Malgouyres Programmation Système En C sous Linux (Debian et Ubuntu)