OS théorie - Synthèse

Brice Delcroix

Janvier 2023

1 Système UNIX

1.1 Définitions

1.1.1 Noyau

Le **noyaux** est la partie centrale du système d'exploitation : Il gère les files systems, l'occupation de la mémoire, le cpu sheduling etc...

1.1.2 Programme

Un Programme est un fichier executable qui à été créé par un éditeur de liens (linker : qui prend les fichiers objets produits par le compilateur et les assemble en un fichier exécutable complet, en résolvant toutes les dépendances et les références nécessaires pour que le programme puisse s'exécuter correctement.) **et se trouve sous forme de fichier sur le disque.**

1.1.3 Processus

Un processus est un programme en cours d'execution. La seule façons d'en créer un nouveau en Unix et l'appel du système **fork**. Sur un système multitache, plusieurs processus peuvent être executés en même temps et un même programme peut être divisé en plusieurs processus, lesquels peuvent être executé en même temps.

2 Le file system.

2.1 Introduction aux concepts du système.

2.1.1 Inode

Représentation interne d'un fichier. Contient : Une description de l'aspect disque *(manière dont les données du fichier sont stockées physiquement sur le disque dur)* et d'autres informations d'ordre "administratif".
Chaque fichier à UN SEUL inode mais un inode peut avoir plusieurs nom (faisant tous référence au même inode.)Nom = Link.
Les inodes disques [1] contiennent :

- Type du fichier
- Permissions
- Identifications du propriétaire.

[1] **Inode** : C'est le terme général pour la structure de données mentionnée ci-dessus. Lorsqu'on parle d'un inode dans un contexte Unix, on fait généralement référence à cette structure de données.
Inode disque : Ce terme est parfois utilisé pour désigner spécifiquement la représentation physique de l'inode sur le disque dur. Il s'agit de la zone où les informations de l'inode sont physiquement stockées. Quand un fichier est accédé, les informations de son inode disque sont lues en mémoire.
Les deux termes sont souvent utilisés de manière interchangeable, car ils se réfèrent à des aspects complémentaires de la même entité.

- Nombres de links
- taille du fichier
- temps du dernier accès au fichier
- Temps de la dernière modifications du fichier.
- Une table pour les adresses disque des données (les blocs qui les contiennent)

Remarque : Les inodes ne spécifie pas le(s) pathname(s) du fichier.

2.1.2 In-core-Inode table (IIT)

Table contenant la copie des inodes en mémoire où le noyau les manipules.

2.1.3 Autres structures

- file table
- user file descriptor table

La **file table** garde une trace en byte du lieu de l'action (écriture / lecture), ainsi que des droit d'accès du fichier pour le processus qui l'a ouvert.
Chaque entrée de cette table contient un pointeur sur l'inode du fichier (dans IIT).
La **user file descriptor table** identifie les fichier ouvert par un processus. Les trois premère entrées de la UFDT font référence à des fichier ouverts automatiquement par tout processus. STDIN, STDOUT, STDERR (standard input/output/error).

2.1.4 File system

C'est une manière d'organiser et de stocker des fichiers sur un dispositif de stockage comme un disque dur. Dans le livre on parle de portion logique.
Le noyau s'occupe de gèrer les files system. Il considère chaque file system comme un device logique indentifié par un logical device number.
Le file system est constitué d'une séquence de blocs logiques, contenant chacun un multiple de 512 bytes.

2.1.5 Structure du File system

- boot block: Il contient le chargeur de démarrage qui est utilisé pour démarrer le système d'exploitation.
- Super block : **Contient des informations essentielles**, sa taille, le nombre de fichiers qu'il peut contenir etc ...
- Liste des inodes : contient les inodes pour tous les fichiers et répertoires.
- Blocs de données : Ce sont les espaces où les données réelles des fichiers et des répertoires sont stockées. Un bloc est alloué à un seul fichier.

2.2 Structure d'un fichier ordinaire.

2.2.1 Algorithme BMAP

L'algorithme "bmap", ou "block mapping" est une méthode utilisée pour trouver l'emplacement physique sur le disque dur des blocs de données d'un fichier. Cet algorithme est crucial car **les fichiers sont stockés sur le disque en blocs**, qui peuvent être dispersés à différents endroits. Voici une explication simplifiée de l'algorithme bmap :

- L'algorithme commence avec deux informations : **l'inode du fichier** (qui contient les métadonnées du fichier et des pointeurs vers ses blocs de données) qui contient **un offset en octets** (qui indique la position spécifique dans le fichier où l'opération de lecture ou d'écriture doit commencer).
- bmap utilise l'offset pour déterminer quel bloc de données du fichier correspond à cette position. Par exemple, si l'offset est à 5000 octets et que chaque bloc fait 4096 octets, bmap sait que les données sont dans le deuxième bloc du fichier.
- **Determiner le Niveau d'indirection** : Lorsqu'un fichier est trop grand pour que tous ses blocs de données soient référencés par des pointeurs directs, l'inode utilise un pointeur indirect.
- bmap va donc parcourir les bloc à la recherche du pointeur direct et renvera l'offset à l'intérieur du bloc de données trouvé. Cela indique la position exacte à l'intérieur du bloc où commencer la lecture ou l'écriture.

2.3 Appels systèmes interfaçant avec le file system.