Université de Versailles – Saint-Quentin-en-Yvelines
Institut des Sciences et Techniques des Yvelines

Pré-Rapport
Création d'un Mini Système de Gestion de Fichier

Réalisé par :

- O'nicé Coelho
- Mohammed Amine Ouasti
- Omar Saidi
- Rania Tounsi
- Marouane Ouzzman
- El Mehdi Guemrani

Encadré par :

- Mr. Thierry Garcia
- Mme. Dhekra Abouda

Année universitaire 2017-2018

Tables des matières

1. Introduction:

Dans le cadre de notre formation en IATIC3 à l'Institut des Sciences et Techniques des Yvelines, nous sommes amenés à réaliser un projet pédagogique. Cela consiste à créer un Mini Système de Gestion de Fichier qui simule le fonctionnement du SGF Unix en utilisant le language C.

Notre rôle sera d'analyser les besoin d'utilisateur afin de construire notre système en répondant au caractères pédagogiques visés par nos résponsables M. Thierry Garcia et Mme. Dhekra Abouda.

La première étape de ce projet réside dans la réalisation du cahier des charges. Ce cahier va nous permettre de définir clairement les bases, objectif et contraintes du projet. C'est le point de départ de tout projet et sa rédaction va conditionner la bonne réalisation du jeux.

2. Généralités:

2.1. SGF :

Le Système de Gestion de Fichiers (SGF) la partie la is visible, plus d'un Système d'exploitation Qui se charge de le et Gérer le stockage de la manipulation Fichiers (Sur une unité de stockage: partition, disque, CD, disquette.

Un SGF a pour rôle principal de gérer les fichiers et d'offrir les primitives pour manipuler ces fichiers. Il se passe généralement les tâches suivantes:

- Une interface conviviale est mise en disposition pour manipuler les fichiers (vue aportée à l'utilisateur). Il s'agit de simplifier la gestion de fichiers pour l'utilisateur (généralement, les utilisateurs sont les attributs et l'extension du fichier, les autres attributs sont impliqués implicitement par le SGF). Cette interface donne la possibilité d'effectuer plusieurs opérations sur les fichiers. Ces opérations permettent entre autre d'ouvrir, fermer, copieur, renommer des fichiers et des répertoires.
- La gestion de l'organisation des fichiers sur le disque.
- La gestion de l'espace libre sur le disque dur.
- La gestion des un environnement Dañs Fichiers Multi-Utilisateurs, la donnée le diagnostic verser utilitaires, la récupération en d'CAS Erreurs, l'organisation des Fichiers Que la sécurité AINSI et le Contrôle des Fichiers.

2.2. Le SGF sous Unix

Dans le cas des systèmes de fichiers Unix, les fichiers et les répertoires sont identifiés par un numéro unique, le numéro d'inode. Ce dernier permet d'accéder à une structure de données (inode) regroupant toutes les informations sur un fichier à l'exception du nom, notamment la protection d'accès en lecture, en écriture ou en listes de dates, ainsi que le moyen d 'en retrouver le contenu. Le nom est stocké dans le répertoire associé à un numéro d'inode. Cette organisation présente l'avantage qu'un fichier unique sur le disque peut être connu du système sous plusieurs noms.

2.2.1. Les différents types de fichiers:

- Fichiers normaux (-) :
  - texte : courrier, sources des programmes, scripts, configuration ;
  - exécutables : programmes en code binaire

La commande ls –l donne la liste des fichiers en ligne avec toutes les informations connexes. Par exemple : -rwxrw-r-- 1 etudiant 2LR 34568 avril 3 14 :34 mon-fichier

- Fichiers répertoires (d) : ce sont des fichiers conteneurs qui contiennent des références à d'autres fichiers. Ils permettent d'organiser les fichiers par catégories.

La commande ls –l sur un répertoire donne

drwxr-x--- 1 etudiant 2LR 13242 avril 2 13 :14 mon-répertoire

- Fichiers cachés : Ils se distinguent des autres par la présence d'un point (.) en première position dans leur nom. La commande de listage des fichiers ls ne les affichera pas par défaut. Par contre ls suivi de l'attribut –a est très pratique pour lister ce type de fichier.
- Fichiers spéciaux : situés dans /dev, ce sont les points d'accès préparés par le système aux périphériques. Le montage va réaliser une correspondance de ces fichiers spéciaux vers leur répertoire "point de montage". Par exemple, le fichier /dev/hda permet l'accès et le chargement du 1er disque IDE.

- Fichiers liens symboliques (l) : ce sont des fichiers qui ne contiennent qu'une référence (un pointeur) à un autre fichier. Cela permet d'utiliser un même fichier sous plusieurs noms sans avoir à le dupliquer sur le disque.

La commande ls –l pour un lien donne

lrwxrwxrwx 1 root root 14 Aug 1 01:58 Mail -> ../../bin/mail*

Rappelons que sous UNIX, un fichier, quel que soit son type, est identifié par un numéro d'inode (i-noeud). Ainsi, derrière la façade du shell, un répertoire n'est qu'un fichier, identifié aussi par un inode, contenant une liste d'inode représentant chacun un fichier.

Pour connaître le numéro d'inode d'un fichier, on utilise la commande $ls -i mon-fichier

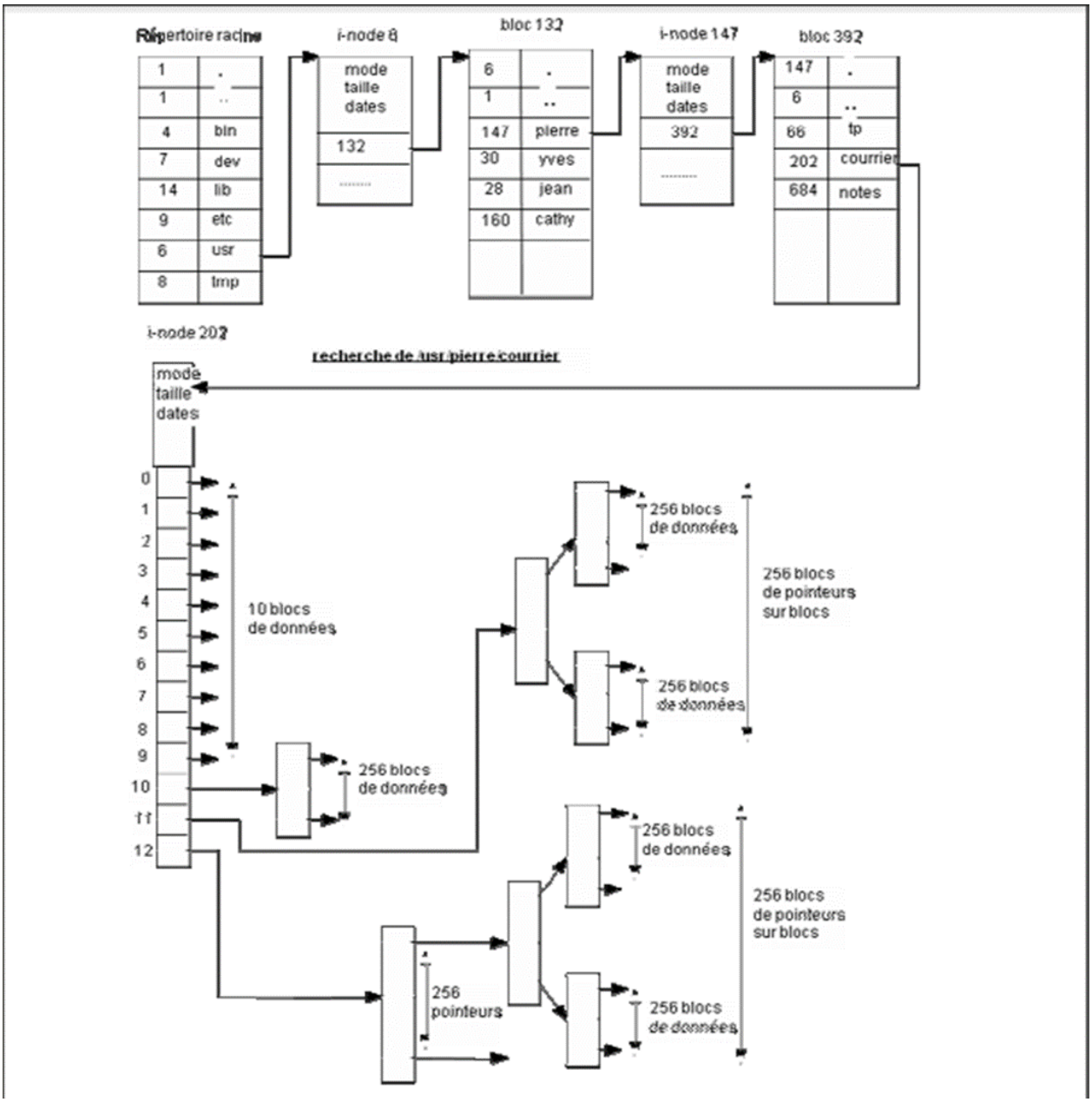
2.2.2. Structure d'un i-node

La structure d'I-Node est utilisée par le Système de Gestion de Fichier ext3fs d'Unix ou GNU/Linux (ext3fs pour third extented file system). Un nœud d'index est constitué d'attributs décrivant le fichier ou le répertoire et d'adresses de blocs contenant des données. Cette structure possède plusieurs entrées, elle permet au système de disposer d'un certain nombre de données sur le fichier :

- la taille,
- l'identité du propriétaire et du groupe : un fichier en Unix est créé par un propriétaire qui appartient à un groupe,
- les droits d'accès : pour chaque fichier, Unix définit trois droits d'accès (lecture (r), écriture (w) et exécution (x)) pour chaque classe d'utilisateurs (trois types d'utilisateur {propriétaire, membre du même groupe que le propriétaire, autres}). Donc, à chaque fichier, Unix associe neuf droits,
- les dates de création, de dernière consultation et de dernière modification,
- le nombre de références existant pour ce fichier dans le système,
- les dix premiers blocs de données,
- d'autres entrées contenant l'adresse d'autres blocs (on parle alors de bloc d'indirection).

La structure d'I-Node est conçue afin d'alléger le répertoire et d'en éliminer les attributs du fichier ainsi que les informations sur l'emplacement des données.

Une entrée dans un I-Node d'un répertoire contiendra donc un nom d'un fichier ou sous-répertoire et l'I-Node associé.



2.2.3. Notion de répertoire

L'ensemble des répertoires est organisé en une hiérarchie de répertoire et de sous répertoires ressemblant à un arbre inversé, dont le tronc est représenté par la racine du disque et les branches par les répertoires.

Comme dans un arbre, il existe plusieurs chemins (sous Linux) pour se rendre de la racine au répertoire voulu.

Tous ces éléments sont hiérarchisés les uns par rapport aux autres. Cette contrainte est représentée par la notion de chemin d'un fichier. En effet, un fichier est localisable de façon exacte et unique par son chemin. Ce dernier représente la succession des répertoires à parcourir pour accéder au fichier (navigation dans l'arbre). Les répertoires sont séparés par un slash noté / dans l'écriture du chemin (attention, c'est le même symbole qui représente la racine : point de départ de l'arborescence).

2.3.4. Chemin d'un fichier :

Un chemin (« path ») est une séquence de répertoires imbriqués, avec un fichier ou un répertoire à la fin, séparés par le caractère /.

- Chemin relatif : docs/cours/unix.html
  - Le chemin relatif désigne la succession des répertoires à parcourir depuis le répertoire courant pour accéder au fichier spécifié.

Exemple: ../monprog.c pour accéder au fichier monprog.c lorsqu'on se trouve dans le répertoire tpC. La présence du répertoire parent dans ce chemin relatif permet de remonter dans l'arbre.

- Chemin absolu : /home/john/docs/cours/unix.html
  - Le chemin absolu désigne la succession des répertoires à parcourir depuis la racine pour accéder au fichier spécifié.

Exemple : /home/h-etie00/tpC/tp3.c pour accéder au fichier tp3.c du système de fichier ou qu'on se trouve dans le système.

- Le répertoire parent est celui hiérarchiquement supérieur au répertoire courant. Il est noté deux points ..
- Le répertoire courant est celui dans lequel on se trouve à un instant donné durant la navigation dans le système de fichiers. Il est noté point .

3. Structures utilisées:

L'objectif visé par ce projet système étant de créer un Système de Gestion de Fichiers, le programme écrit en langage C devra donc être capable de gérer et manipuler les fichiers ordinaires et les répertoires à travers un interpréteur de commandes. Des primitives ainsi que des commandes Shell ont donc étés implémentées.

Pour ce faire, les structures ont été définies :

3.1. Struct info_inode:

```
struct info_inode {

      bool existe;                    //Si le fichier existe

      int taillefichier;              //Longueur du fichier

      char nomfichier[80];            //Nom du fichier

      int type;                       //1: Fichier; 2: Dossier

      char permissions[9];            //rwxr--r—

      int blocutilise[30];           //Quels sont les blocs utilisés (max
      30 ici)

      int rep;                        //Si dossier (num inode vaut num
      rep)

      int monrep;                     //Répertoire des dossier & fichier
      créé

};
```

La structure info_inode contient des informations sur un fichier ou un répertoire du SGF permissions , type , taille…

3.2. Struct unbloc

```
struct unbloc

{

    char donnees[1024]; // par exemple 1024 octets

};
```

La structure unbloc contient la taille de chaque bloc du disque notamment 1024 octets.

3.3. Struct disk

```
struct disk

{

    struct infoinode inode[15]; // 15 inodes

    struct unbloc bloc[30]; // 30 blocs de 1024 octets

};
```

La structure disk contient la totalité des inodes (15 inodes) ainsi que 30 blocs de 1024 octets chacun. Chaque inode contient au moins un bloc qui lui permet de stocker son contenu.

4. Les commandes Shell définies

4.1. Liste des commandes implémentées :

- `cd:` changer de répertoire
- `ls:` lister les fichiers et dossiers du répertoire courant (option -l)
- `touch:` créer un fichier
- `cp:` copier le contenu d'un fichier dans un autre
- `mv:` déplacer ou renommer un fichier
- `rm:` supprimer un fichier
- `cat:` sortir de façon standard du contenu d'un fichier
- `echo:` écriture dans un fichier et/ou création d'un fichier s'il n'existe pas
- `mkdir:` créer un répertoire
- `rmdir:` supprimer un répertoire

4.2. Algorithmes de quelques commandes

- La commande cd
  - Chaque inode (fichier) contient un « nomrep » (struct infoinode) qui représente le répertoire courant de l'endroit où il a été créé. De plus, chaque dossier contient un « rep »(répertoire) qui lui est unique.
  - cd (uniquement) : Il se déplace au répertoire d'utilisateur connecté.

```
if(compare(commande1, "cd") == 0)

{

      rep=Repertoire_Racine(16);

      nomrep=bash;

}
```

- cd [nomrep] :

```
for(int i=0;i<15;i++)//Parcours toutes les inodes

      if([nomfichier]==disk0.inode[i].nomfichier&&disk0.inode[i].exis
te==1  &&  disk0.inode[i].type==2/*Fichier  de  type  Dossier*/  &&
disk0.inode[i].monrep == rep)//Dossier dans le répertoire courant

      {

            rep(courant)=disk0.inode[i].rep;//Repertoire courant prend
      la valeur du repertoire rep du dossier

            nomrep=disk0.inode[i].nomfichier;//Nomrep prend la valeur
du nom du dossier

      }
```

5. Conclusion :

Ce projet est l'occasion pour nous tous de réaliser quelque chose de formidable que nous aurons eu du mal à croire possible avant notre arrivée à l'ISTY. Nous allons pouvoir développer toutes les composantes d'un systèmes de fichiers, tout en conservant une grande liberté sur sa réalisation et sa mise en oeuvre. C'est aussi pour nous l'occasion d'apprendre à travailler en groupe, à s'organiser en gérant les contraintes au niveau du temps, de son propre temps et de celui des autres, au niveau des connaissances qu'il faut acquérir au fur et à mesure sans perdre de vue l'objectif final et sans s'enliser dans des détails et au niveau de la structure interne des systèmes de fichiers bien sûr puisque après la réalisation de ce projet nous pourons dire que nous maîtrisons une grande partie de la programmation système. Nous avons la motivation nécessaire à réaliser ce qui nous laît maintenant il nous reste plus qu'à montrer que nous sommes prêts à aller jusqu'au bout.