Projet SDISDB (SYSTEME)

1 Description et déroulement du projet

Le projet de Système va consister à construire un Mini Système de Gestion de Fichiers.

Les délégués de classe devront nous envoyer la liste des groupes.

Le travail demandé sera à effectuer en équipe de 6 personnes en partageant le travail entre chacun d'entre vous. Il est conseillé de bien modéliser en premier lieu le projet afin de pouvoir répartir les différentes tâches à effectuer entre vous.

Attention, le but du projet n'est pas forcement de tout réaliser mais il est important de mettre en évidence et d'améliorer vos compétences en système d'exploitation, en modélisation et de savoir maîtriser son temps afin de fournir un produit fonctionnel (même minimal).

Le projet devra être codé en C en implémentant le SGF, les primitives, le shell et la sauvegarde/recharge du SGF (voir plus bas). Il faudra prévoir des tests de validation en utilisant la sauvegarde/recharge du SGF.

Penser à structurer le code en séparant les primitives, les commandes shell, la sauvegarde/recharge et la structure du SGF dans des fichiers différents et à commenter les fonctions du projet.

Il serait bien de pouvoir compiler le projet avec un Makefile et utiliser par exemple git pour gérer les différentes versions (Github) (voir Bibliographie).

Vous aurez différentes étapes obligatoires à réaliser ainsi qu'une soutenance qui sera programmée le Vendredi 15 juin 2018 :

- Un rapport décrivant l'analyse du ou des besoins de l'utilisateur, définissant le système à réaliser et comment vous pensez le réaliser, donnant le cahier des charges qui devra contenir la description globale des fonctions principales du Mini Système de Gestion de Fichiers, la ou les structures de données prévues, la liste des fonctions principales et enfin la répartition des tâches entre les membres de l'équipe. Ce rapport sera à envoyer aux deux encadrants avant le 11 mai 2018 minuit.
- Un rapport et vos programmes sources (pas d'exécutable) seront à déposer sur e-campus avant le 13 juin 2018 au soir (minuit dernier délai). Le rapport doit contenir la description et le détails des algorithmes des fonctions principales de votre programme. Le but est de savoir si vous avez compris les algorithmes mis en place. Attention, nous lirons vos codes en langage C. Ce qu'on cherche à avoir de votre part, c'est une version développée par vous et non un plagiat de codes trouvés sur internet.
- Attention, les rapports et les programmes sources rendus en retard ne seront pas notés (sauf autorisation expresse d'un des deux encadrants).
- Une soutenance aura lieu le 15 juin 2018 et les diapos pour vos présentations seront à envoyer la veille de votre soutenance aux deux encadrants.
- Une séance dédiée aux projets aura lieu durant les semaines consacrées aux projets (éventuellement avant suivant les besoins) mais nous vous autorisons à nous contacter par mail pour tout problème ou blocage dans vos développements.

Les encadrants du projet sont :

- Thierry Garcia, thierry.garcia@uvsq.fr ;
- Dhekra Abouda, dhekra.abouda@gmail.com.

2 Le projet

2.1 Cahier des charges

Ce projet va consister à réaliser un émulateur de gestionnaire de fichiers (SGF) avec un interpréteur bash. Le but de ce projet est donc de créer votre propre interpréteur de commandes shell en utilisant des primitives qui interagissent avec le SGF.

Lorsqu'on exécute le projet, on a accès à une fenêtre qui attend des commandes (comme le shell de linux).

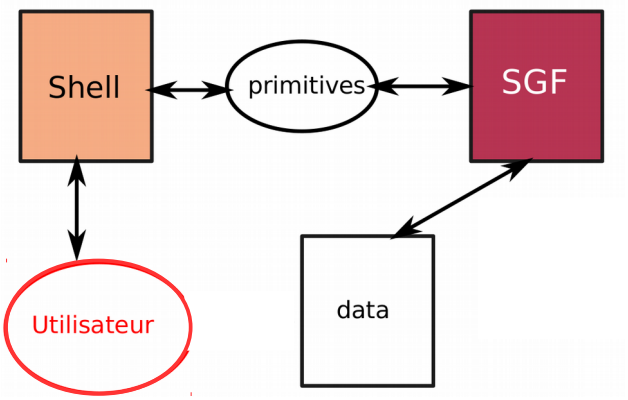
```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
[gath@home sgf]$ ./myshell
[myshell] $
[myshell] $
[myshell] $
[myshell] $ ls
myshell  myshell.c  toto
[myshell] $ ls -l
total 28
-rwxrwxr-x. 1 gath gath 13848  5 avril 15:41 myshell
-rw-rw-r--. 1 gath gath  5448  5 avril 15:41 myshell.c
drwxrwxr-x. 2 gath gath  4096  5 avril 15:42 toto
[myshell] $ mkdir titi
[myshell] $ ls -l
total 32
-rwxrwxr-x. 1 gath gath 13848  5 avril 15:41 myshell
-rw-rw-r--. 1 gath gath  5448  5 avril 15:41 myshell.c
drwxrwxr-x. 2 gath gath  4096  5 avril 15:44 titi
drwxrwxr-x. 2 gath gath  4096  5 avril 15:42 toto
[myshell] $ rmdir titi
[myshell] $ ls -l
total 28
-rwxrwxr-x. 1 gath gath 13848  5 avril 15:41 myshell
-rw-rw-r--. 1 gath gath  5448  5 avril 15:41 myshell.c
drwxrwxr-x. 2 gath gath  4096  5 avril 15:42 toto
[myshell] $ exit
[gath@home sgf]$
```

La création de cette fenêtre n'est pas trop compliquée en créant un processus fils et en le recouvrant par une primitive exec.

Cependant, les objectifs demandées sont plus complexes … il est demandé dans ce projet de :

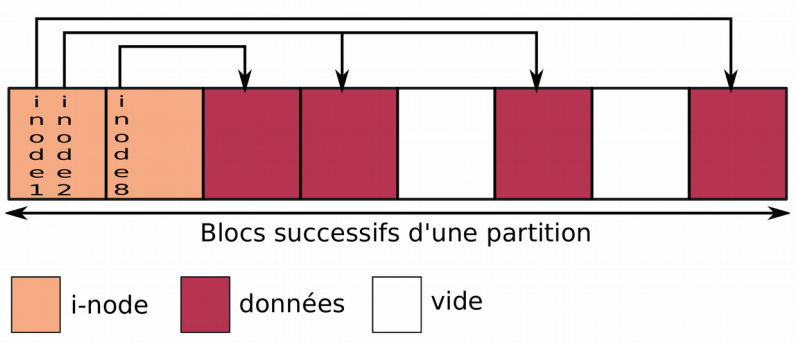
- Faire un interpréteur basique (mini bash) ;
- Reproduire la structure interne des inodes/blocs et l'arborescence des fichiers (faire le SGF) ;
- Définir un ensemble de primitives qui permettent d'interagir avec le SGF ;
- Définir un ensemble de commandes shell qui utilisent ces primitives.

Dans le schéma ci-dessous, un utilisateur va lancer un terminal puis va exécuter des commandes. Chaque commande va faire appel à des primitives permettant d'interroger/modifier le SGF qui lui-même utilise un fichier de données pour stocker les informations.

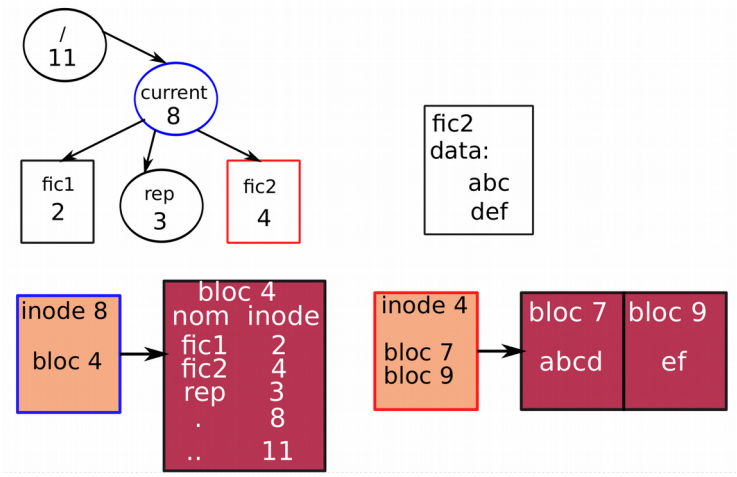


2.2 Système de fichiers

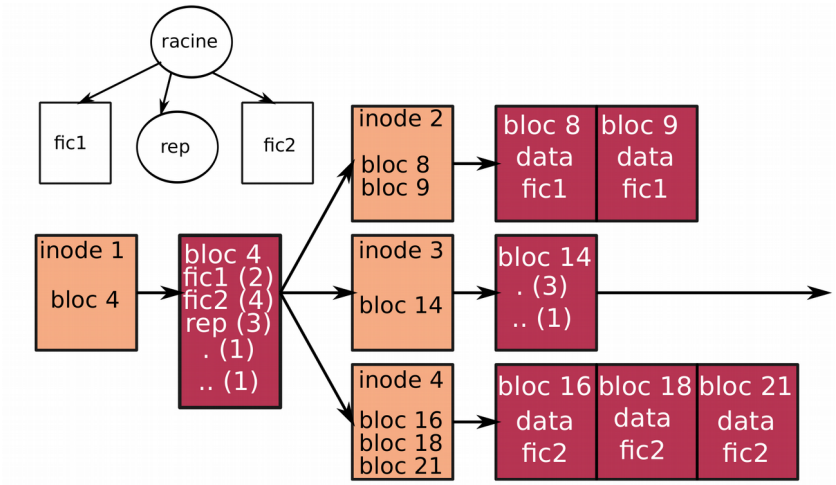
On pourrait imaginer un système de fichier qui s'inspire du SGF de Linux. Par exemple, le système de fichiers ci-dessous permet de simuler une partition. Les i-nodes permettent (entre autres) de connaître la position des blocs de données dans le SGF.



Les fichiers et répertoires peuvent être représentés de la manière suivante (exemple) :



La structure arborescente peut être représentée de la manière suivante (exemple) :



Afin de simuler un disque dur (sur le disque dur physique de votre machine), il est possible de créer un gros fichier sur le disque qui va avoir un fonctionnement s'approchant du fonctionnement du SGF Linux (c'est par exemple ce qui se passe dans une machine virtuelle - il y a un gros fichier image sur le disque dur).

Dans ce gros fichier (binaire ? texte ?), il faut réfléchir à certains points : comment on peut accéder aux différents fichiers, comment les créer, comment les détruire, comment les placer (contiguë ? par bloc ? chaîné ? <-- si on détruit un fichier comment récupère-t-on la place) ?

Suivant les décisions prises concernant ce fichier, il faut créer des primitives qui permettent d'y accéder (par exemple myopen permet d'ouvrir le fichier (on cherche où il est), myread permet de lire son contenu, mycreat permet de le créer etc).

2.3 Primitives

Les primitives des fichiers à définir pourraient être :

- mycreat(nom,mode): créé un fichier et retourne son inode
- myopen(nom,mode): ouvre/créé un fichier et retourne son inode
- myclose(inode): ferme un fichier
- myread(inode,buffer,nombre): lit nombre octets mis dans buffer
- mywrite(inode,buffer,nombre): écrit nombre octets pris dans buffer
  - gérer l'éventuelle allocation de nouveaux blocs pour les données

Les primitives des répertoires/fichiers à définir pourraient être :

- mkdir(nom): créer un répertoire
- rmdir(nom): supprime un répertoire vide
- link(nom1,nom2): fait le lien entre une nouvelle entrée du répertoire et un fichier déjà existant
- unlink(nom): efface une entrée du répertoire
  - si aucune entrée sur un fichier, supprimer son inode et ses blocs de données

2.4 Interpréteur shell

La surcouche shell devra pouvoir effectuer les commandes suivantes :

- Commandes sur des fichiers : cp rm mv cat ln 'echo "texte" > file'
- Commandes sur des répertoires: ls mkdir rmdir cd
- df: informations sur le superbloc
  - Nombre de blocs disponibles
  - Nombre d'inodes disponibles
  - Taille en octets de l'espace libre sur le disque
- Ajouter des méta données sur les inodes
- Droits d'accès (gérer les fichiers en lecture/écriture seule)
- Date de modification
- Bonus :
- Gérer plusieurs utilisateurs dans le shell
- Ajouter une version réduite de find et grep dans le shell
- Étendre le mini shell suivant vos idées

2.5 FAQ

Q : On aimerait savoir dans la page 3 dans le schéma pourquoi vous avez mis bloc 4 et inode 4 est ce qu'il vont occuper la même case ce qui n'est pas possible ou c'est juste une faute de frappe .

R : Un inode va représenter dans une structure les informations associées à un fichier. Mais comme un répertoire est considéré comme un fichier, l'inode représente aussi un répertoire. Voir http://langevin.univ-tln.fr/cours/UPS/extra/chapitre3_sgf.pdf paragraphe 3.4 et figure 3.4. Les transparents 13, 16 et 17 de http://www-sysdef.lip6.fr/~hyon/SupportsCours/SYS_L3_SGF.pdf doivent pouvoir vous aider. Les inodes sont indépendants des blocs de données sur disque. Dans notre cas, l'inode 4 "pointe" vers les blocs disque 16, 18 et 21 qui contiennent des données et le bloc 4 contient les infos concernant l'inode 1.

Q : Est ce que l'inode 1 est un répertoire ?

R : Dans notre cas, il représente la racine. Il représente un répertoire. Ses données sont dans le bloc 4
Q : Pour le bloc 14 : .(3) et ..(1) veut dire la traçabilité du chemin du bloc ? Si c'est le cas pourquoi dans le bloc 4

pourquoi vous avez mis deux fois 1 : .(1) et ..(1) ?

R : . est le répertoire actuel dans le bloc 14 : .(3) indique l'inode 3
.. est le répertoire au dessus : ..(1) indique que le répertoire "père" est l'inode 1
De même pour .(1) et ..(1) - on est à la racine - les informations sur le répertoire courant sont données par l'inode 1 et le répertoire précédent est le même puisque qu'il n'y en a pas d'autres.

Q : Comment créer un SGF, en stockant le tout dans un fichier txt qui ferait office de disque dur ... Séparer nos différents éléments avec un caractère défini. Placer chaque partie d'un fichier dans un inode à taille fixe.

Concernant le projet, la version minimum est de pouvoir voir la notion d'inode et de bloc. En fait l'inode contient des informations sur un fichier ou un répertoire du SGF. Dans ce dernier cas, il va permettre de retrouver tous les blocs disque contenant les données du fichier sachant que les blocs ne sont pas forcement contigus. De plus, il va contenir toutes les informations utiles : permissions, type de fichier ... Concernant les répertoires, on a simplement une liste de noms et d'inodes … Voir aussi : http://lipn.univ-paris13.fr/~cerin/SE/SF1.pdf

Pour notre simulation, on peut avoir par exemple une table d'inodes suivi des fichiers et répertoires. La question est de savoir comment implémenter cette structure de donnée. On peut décider qu'il n'y a pas de dynamisme (nombre d'inodes définis, nombre de blocs définis). Par exemple : disque = tableau d'inode + tableau de blocs. On pourrait imaginer (ici on simule un disque dur (petit et non dynamique) - il faudrait des pointeurs, traiter la longueur des fichiers, ...) :

```
#include <stdio.h>
struct infoinode
{
   char permissions[8]; // rwxr--r--
   int typefichier; // 1 = texte, 2 = binaire , ...
   int blocutilise[30]; // quels sont les blocs utilisés (max 30 ici)
   // peut être mettre la longueur du fichier
};

struct unbloc { char donnees[1024]; // par exemple 1024 octets };

struct disk {
   struct infoinode inode[15]; // 15 inodes
   struct unbloc bloc[30]; // 30 blocs de 1024 octets };

int main()
{
  struct disk disk0; // mon disque
  int i,j,k;

  // formatage du disque (on met tous les inodes à 0
  for (i=0;i<15;i++)
  {
       disk0.inode[i].typefichier=0;
       for (j=0;j<15;j++) disk0.inode[i].blocutilise[j]=-1;
  }

  // simulation d'un inode
  disk0.inode[0].typefichier=1;
  disk0.inode[0].blocutilise[0]=0;
  disk0.inode[0].blocutilise[1]=2;
  // simulation d'un fichier de 2048 octets dans les blocs 0 et 2
  for (i=0; i<1023; i++) disk0.bloc[0].donnees[i]='a';
  for (i=0; i<1023; i++) disk0.bloc[2].donnees[i]='b';

  // simulation de lecture d'un fichier à partir de son inode
  for (i=0; i<15; i++)
       // on simule un cat du fichier
       if (disk0.inode[i].typefichier == 1) // fichier texte
```

```
        for (j=0; j<15; j++)
           if (disk0.inode[i].blocutilise[j]!=-1)
              for (k=0; k<1023; k++) printf("%c",disk0.bloc[disk0.inode[i].blocutilise[j]].donnees[k]);

  return 0; }
```

2.6 Bibliographie

Makefile
http://gl.developpez.com/tutoriel/outil/makefile/
http://www.labri.fr/perso/billaud/Resources/resources/AP2-POO-0910/060-faire-makefile.pdf

Git
http://rogerdudler.github.io/git-guide/index.fr.html
http://bioinfo-fr.net/git-premiers-pas

SGF
http://langevin.univ-tln.fr/cours/UPS/extra/chapitre3_sgf.pdf
http://www-sysdef.lip6.fr/~hyon/SupportsCours/SYS_L3_SGF.pdf
http://www.groupes.polymtl.ca/inf2610/documentation/notes/chap11.pdf
http://lipn.univ-paris13.fr/~cerin/SE/SF1.pdf

Exemples
http://jean-luc.massat.perso.luminy.univ-amu.fr/ens/systeme/tp9-sgf.html
http://jean-luc.massat.perso.luminy.univ-amu.fr/ens/systeme/tp10-sgf.html
http://jean-luc.massat.perso.luminy.univ-amu.fr/ens/systeme/mini-sgf.zip

http://www.lsis.org/brennerl/enseignement/ENSIN6U3/TP_9_10_2018.pdf
http://www.lsis.org/brennerl/enseignement/ENSIN6U3/
http://www.lsis.org/brennerl/enseignement/ENSIN6U3/code/mini-sgf.zip

https://github.com/brenns10/lsh