

# Projet NSY103 sous LINUX

## 1 Description et déroulement du projet

Ce mini-projet va consister à construire un Mini Système d'Exploitation assez simple. Le projet sera à effectuer en équipe 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.

Dans ce projet, 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. Le sujet de ce projet est un cadre minimal à mettre en place mais il est possible si et seulement si celui-ci est fonctionnel, de faire preuve de créativité et de développer tout ce qui vous paraît intéressant. Il devra être codé en langage C et il faudra veiller à structurer le code en séparant les fonctions, les commandes shell et la structure du SGF dans des fichiers différents et à commenter les fonctions du projet.

Il vous est demandé de réaliser :

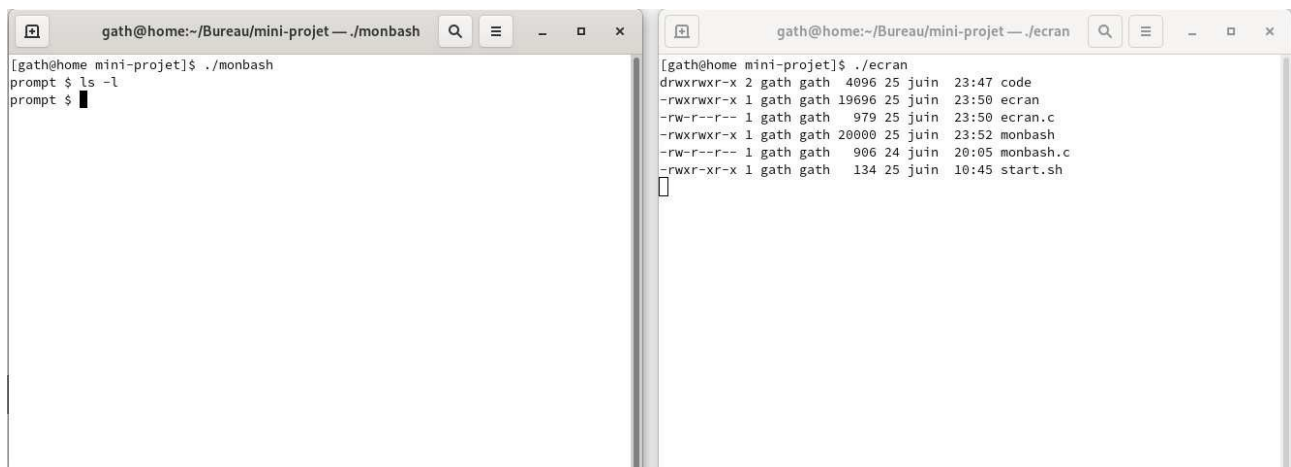
- Un rapport décrivant l'analyse du ou des besoins de l'utilisateur, définissant le système à réaliser et comment vous pensez le réaliser, donnant la description globale des fonctions principales, la ou les structures de données prévues, la liste des fonctions principales et enfin la répartition des tâches entre les membres de l'équipe.
- Un rapport et vos programmes sources (pas d'exécutable) seront à déposer sur lecnam.net. Le but est de savoir si vous avez compris les algorithmes mis en place.

## 2 Le projet

Remarques générales : une bibliographie/sitographie est donnée à la fin de ce document. Cela permettra de vous aider.

### 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 fonctions qui interagissent avec un SGF. Lorsqu'on exécute le projet, on doit pouvoir avoir accès à une fenêtre qui attend des commandes (comme le shell de linux) et une fenêtre qui affiche le résultat des instructions.



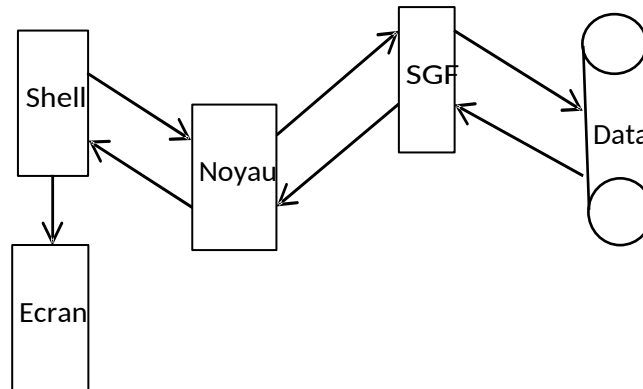
```
[gath@home mini-projet]$ ./monbash
prompt $ ls -l
prompt $
```

```
[gath@home mini-projet]$ ./ecran
drwxrwxr-x 2 gath gath 4096 25 juin 23:47 code
-rwxrwxr-x 1 gath gath 19696 25 juin 23:50 ecran
-rw-r--r-- 1 gath gath 979 25 juin 23:50 ecran.c
-rwxrwxr-x 1 gath gath 20000 25 juin 23:52 monbash
-rw-r--r-- 1 gath gath 906 24 juin 20:05 monbash.c
-rwxr-xr-x 1 gath gath 134 25 juin 10:45 start.sh
```

Il est demandé dans ce mini-projet de :

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

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



## 2.2 Le Shell

Le shell est un programme (processus) qui va s'exécuter et se mettre en attente d'une commande. Ce processus est chargé de créer un processus fils (noyau). Le shell va envoyer la commande au noyau et ce dernier va utiliser la bibliothèque de fonctions du SGF afin de traiter la commande et renvoyer les résultats vers l'écran.

## 2.3 L'Ecran

L'écran est un terminal classique (ce n'est pas un fils du shell) dans lequel un programme en C s'exécute et attends des données à afficher. Lors des simulations, on peut ouvrir manuellement 2 terminaux et mémoriser les pid avec un ps + grep (ou autres). Sinon, le script ci-dessous devrait fonctionner :

```
#!/bin/bash gnome-terminal -- /bin/sh -c 'echo $$>pid.txt  
&& exec bash' gnome-terminal -- /bin/sh -c 'echo $  
$>>pid.txt && exec bash'
```

Ces lignes sont à enregistrer dans un fichier nommé par exemple start.sh. Il faudra donner les droits d'exécution à ce fichier avec chmod. Puis lancer le script comme un programme C ./start.sh. Ce script ouvre 2 terminaux (il faut être sous l'environnement gnome sinon il faut l'adapter) et mémorise leur pid dans le fichier pid.txt.

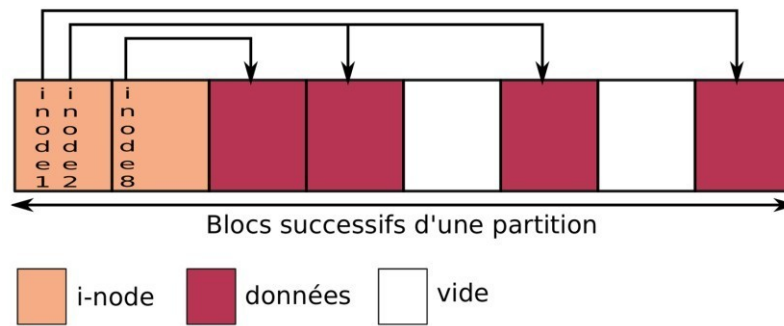
## 2.4 Les communications

Une réflexion sur les moyens de communication dans ce système est à faire. Quels moyens de communication peuvent être utilisé pour communiquer entre le processus Shell et son fils le processus Noyau ? Même question pour la communication entre le processus Shell et le processus Ecran.

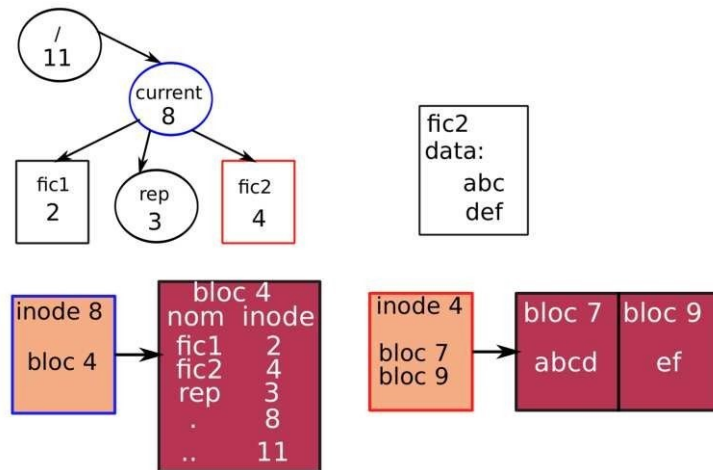
Dans tous les cas, on ne désire pas utiliser les sockets et on désire que le moyen de communication utilisé entre le processus Shell et son fils soit différent de celui utilisé entre le Shell et l'Ecran. Par exemple, on considérera que les tubes anonymes et nommés ne sont pas différents et ne peuvent pas être utilisés dans les deux cas. Il faudra donc se baser sur d'autres moyens de communication comme par exemple les files de messages ou la mémoire partagée.

## 2.5 Système de fichiers

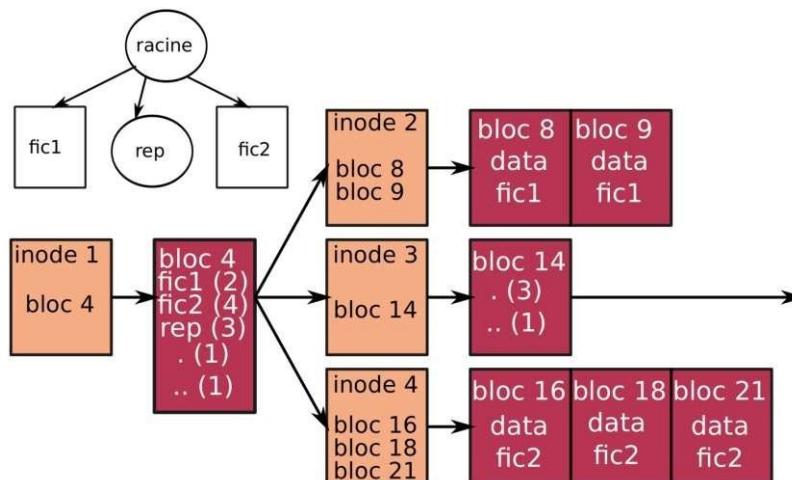
Le but serait d'imaginer un système de fichier qui s'inspire du SGF de Linux. Par exemple, le système de fichiers cidessous permet de simuler une partition. Les i-nodes permettent (entre autres) de connaître la position des blocs de données dans le SGF.



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 peut-on accéder aux différents fichiers, comment les créer, comment les détruire, comment les placer (contiguë ? par bloc ? chaîné ? <-- si on détruit un fichier comment récupérer-t-on la place) ?

Suivant les décisions prises concernant ce fichier, il faut créer des fonctions 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).

## 2.6 Fonctions SGF

Les fonctions de manipulation des fichiers pourraient être :

- mycreat(nom,mode): crée un fichier et retourne son inode
- myopen(nom,mode): ouvre/crée 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 fonctions des répertoires/fichiers 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

## 2.7 Interpréteur shell et langage de commande

La surcouche Shell devrait pouvoir permettre à l'utilisateur d'effectuer les commandes suivantes :

- Commandes sur des répertoires : ls, mkdir, rmdir, cd
- Prendre en compte les arguments comme par exemple : ls -l, rmdir rep1 rep2, mkdir rep1, rep2
- Commandes sur des fichiers : cp, rm, mv, cat, echo "texte" > file
- Autres commandes : df qui permettra d'avoir des infos sur votre SGF, chmod qui permet de changer les droits

## 2.8 Interpréteur shell - bonus

- Étendre le mini shell suivant vos idées – cela sera pris en compte !!!

## 2.9 Bibliographie/sitographie

Makefile

<http://gl.developpez.com/tutoriel/outil/makefile/>

<http://www.labri.fr/perso/billaud/Resourses/resources/AP2-POO-0910/060-faire-makefile.pdf>

Bash + SGF

[http://langevin.univ-tln.fr/cours/UPS/extra/chapitre3\\_sgf.pdf](http://langevin.univ-tln.fr/cours/UPS/extra/chapitre3_sgf.pdf)

<https://cours.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/mini-sgf.zip>

<https://github.com/brenns10/lsh>