Université de Rouen
UFR des sciences et techniques
Projet annuel de licence 3

Encadrant : Pascal Caron

Manuel d'utilisation

Christopher Coat - Guillaume Dauster
Florian Guilbert - Étienne Reith

Rouen, le 2 mai 2011

Table des matières

Introduction

Ce manuel concerne le logiciel de Système de Gestion de Fichiers (SGF) créé dans le cadre du projet annuel de 3ème année de Licence d'informatique à l'Université de Rouen.

Quatre étapes sont nécessaires pour pouvoir l'exploiter : la compilation des fichiers sources (un simple `make` suffit), la création du fichier contenant le SGF lui même (grâce a la commande `zmkfs`), le lancement du kernel recevant et exécutant les commandes, et enfin le lancement du (ou des) `shell`, permettant les interactions avec l'utilisateur.

1 ZMKFS : Création du système de fichiers

Cette commande permet la création d'un fichier contenant toutes les informations (= opération de formatage) permettant sa bonne gestion en tant que système de fichier.

1.1 Utilisation par défaut

Cette commande peut ne contenir qu'un seul argument : le `pathname` du fichier que l'on veut créer. Si aucune autre option n'est spécifiée, la taille du fichier sera de 104 857 600 octets, soit 100Mio et sera découpé en blocs de 1024 octets. Si le fichier existe déjà, un message d'erreur invitant l'utilisateur à forcer le formatage est affiché et le programme se termine.

```
./zmkfs pathname
```

1.2 Forcer le formatage

Si le fichier existe déjà mais que nous voulons forcer le formatage de celui-ci, il suffit de rajouter l'option `-f`.

```
./zmkfs pathname -f
```

1.3 La taille du fichier

On peut préciser la taille du fichier via l'option `-t entier[K, M, G]`. Si aucune lettre n'est ajoutée après l'entier, la taille est exprimée en octets. Si K, M ou G est ajouté, l'entier représentera respectivement des Kio, Mio ou Gio. Les deux commandes suivantes seront donc équivalentes :

```
./zmkfs pathname -t 1024
./zmkfs pathname -t 1K
```

1.4 La taille des blocs

La taille des blocs peut être fixée à 1024 octets (par défaut), 2048 octets ou 4096 octets. L'option `-b entier[K]` permet de le fixer. Les deux commandes suivantes seront équivalentes :

```
./zmkfs pathname -b 1024
./zmkfs pathname -b 1K
```

2 KERNEL : Lancement du noyau du système de fichiers

Ensuite il faut lancer le `kernel` avec comme argument un système de fichiers préalablement créé.

```
./kernel monSGF.fs
```

Le programme affiche sur la sortie standard tout un descriptif sur le système de fichiers utilisé.

Régulièrement, vous pourrez observer la mention `synchronisation` sur la sortie du kernel, cela signifie que le kernel a synchronisé ses données avec le système de fichiers.

Pour fermer le kernel, vous pouvez envoyer les signaux suivants : SIGINT, SIGQUIT, SIGTERM.

3 SHELL : Lancement du/des client(s)

Si l'étape précédente est accomplie, nous pouvons lancer un shell, en prenant comme argument un système de fichiers monté par un kernel. Le système pouvant gérer plusieurs utilisateurs, vous pouvez lancer plusieurs clients sur un même système de fichiers.

```
./shell monSGF.fs
```

Une fois le programme lancé, on voit sur la sortie standard, un prompt de cette forme : `u@h|sgf :wd*`

u : correspond au nom de l'utilisateur.

h : correspond au nom d'hôte de la machine.

sgf : correspond au nom du système de fichiers utilisé.

wd : correspond au répertoire de travail.

*** :** deux possibilités, $ si `u` est un simple utilisateur et # si `u` est `root`.

3.1 Commandes disponibles

Après l'affichage du prompt, vous pouvez lancer les commandes suivantes :

zmd rep : crée un répertoire de nom `rep`, ou produit une erreur si un fichier ou un répertoire du même nom existe.

zln fichier lien : crée un lien physique `lien` pointant vers `fichier` qui ne peut être un répertoire.

zrd rép : supprime `rép`, il faut qu'il soit vide pour que la suppression s'effectue.

zrm fichier : supprime `fichier`.

zcd (rep) : place l'utilisateur dans le répertoire spécifié, s'il n'y a pas de répertoire spécifié, l'utilisateur est placé à la racine.

zmv source dest : change le nom de `source` vers `dest`, permet de déplacer un fichier en précisant un nom dans un autre répertoire.

zls (-l) (fichier) : liste le contenu de `fichier` si celui-ci est un répertoire sinon affiche des informations sur le fichier. Avec l'option `-l` la liste obtenu est pourvue d'informations sur chaque fichier du répertoire, l'option n'a aucun effet lorsque l'argument est un simple fichier. Si aucun fichier n'est spécifié, la commande agit sur le répertoire courant.

zcat fichier : affiche le contenu du fichier.

zcp source dest : copie le contenu de `source` vers `dest`.

zu2cp sourceUnix destSGF : copie un fichier du système hôte vers le système de fichiers en cours d'utilisation.

zcp2u sourceSGF destUnix : copie un fichier du système courant vers le système hôte.

zchmod droits fichier : donne les droits spécifiés à `fichier` les droits doivent être sur quatre caractères et de la forme : `rw--` par exemple pour donner les droits de lecture/écriture au propriétaire.

zchown propriétaire fichier : donne la propriété de `fichier` à l'utilisateur spécifié.

ztouch fichier : cela permet de mettre à jour la date de dernière modification.

quit ou exit : quitte le programme ; envoyer les signaux : SIGINT, SIGQUIT, SIGTERM a le même effet.

3.2 Disposition à prendre en cas de multi-utilisateurs

Pour utiliser le système de fichiers avec plusieurs utilisateurs, il faut bien faire attention à donner les droits de lecture, d'éctirure et d'exécution aux utilisateurs que vous voulez utiliser pour lancer le programme `shell`. Les droits d'exécution afin d'exécuter le programme, et lecture et écriture car le programme va faire transiter des informations entre le kernel et lui par des sockets unix, il faut donc que l'utilisateur du programme soit en mesure de lire et écrire dans des fichiers du répertoires.