Un lien dur est une entrée supplémentaire pointant vers le même inode alors que le lien symbolique est un fichier spécial (type `l`) contenant le chemin du fichier vers lequel il pointe ; ce dernier est équivalent aux "raccourcis" pour les systèmes Microsoft Windows.

d. Considérations supplémentaires

Les deux composantes principales d'un système de fichiers Unix sont les inodes et les blocs de données. Si une de ces deux ressources n'est plus disponible, le système de fichiers est saturé.

Les systèmes de fichiers peuvent proposer encore d'autres fonctionnalités :

- Quotas : permet de limiter l'utilisation du système de fichiers suivant l'utilisateur ou le groupe.
- Journalisation : augmente la rapidité de vérification d'un système de fichiers (après un arrêt brutal de la machine par exemple). Pour cela, une trace (journal) des modifications du système de fichiers est enregistrée.
- Listes de contrôles d'accès (ACL) : étendent la gestion des droits d'accès aux fichiers.
- Libellés : il est possible de spécifier un libellé pour certains systèmes de fichiers, ce qui permet de gérer le montage en fonction d'une étiquette au lieu du nom d'unité de type bloc.
- Attributs spécifiques : les systèmes de fichiers définissent des paramètres spécifiques suivant leur architecture et les fonctionnalités qu'ils mettent en œuvre.

Il est possible d'optimiser la plupart des systèmes de fichiers Unix en jouant sur la taille des blocs et la proportion d'inodes/blocs de données à la création d'un système de fichiers. Ainsi, si les fichiers à stocker sont petits et nombreux (serveur de news ou de mail), de petits blocs de données avec une grande proportion d'inodes est préférable ; à l'inverse pour stocker peu de fichiers volumineux, un nombre restreint d'inodes avec des blocs de plus grande taille est mieux adapté.

2. ext2

Devenu un standard sous Linux depuis 1994, ext2 (*second extended filesystem*) est un système de fichiers très utilisé sur ce système d'exploitation.

Il est assez performant et peu sensible à la fragmentation mais n'est pas journalisé. En cas de redémarrage intempestif de la machine, si une partie des données n'est pas écrite mais encore en cache, la procédure de vérification sera longue et le résultat incertain. Dans cette optique un répertoire *lost+found*, présent dans la racine du système de fichiers, est utilisé avec la commande de vérification **`fsck`**.

La création d'un tel système de fichiers se fait avec :

```
mke2fs [-b <taille des blocs>] [-i <octets par inode>]
 [-c] <unité de type bloc>
```

Par exemple, pour formater en ext2 la partition */dev/sdc1* avec des blocs d'une taille de 1 ko **`(-b 1024)`**, un rapport de 1 inode pour 4 ko de données **`(-i 4096)`** soit 1 inode pour 4 blocs de données, tout en vérifiant les secteurs défectueux du disque (**`-c`**) :

```
[root]# mke2fs -b 1024 -i 4096 -c /dev/sdc1
mke2fs 1.37 (21-Mar-2005)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
245760 inodes, 979932 blocks
48996 blocks (5.00%) reserved for the super user
First data block=1
120 block groups
8192 blocks per group, 8192 fragments per group
2048 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409, 663553

Checking for bad blocks (read-only test): done                          932
Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 26 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

© Editions ENI - All rights reserved

C'est le paquetage *e2fsprogs* qui contient cette commande ainsi que l'ensemble des programmes permettant de manipuler un tel système de fichiers, notamment :

- **badblocks** : recherche les blocs défectueux sur un périphérique.
- **debugfs** : corrige des erreurs du système de fichiers.
- **dumpe2fs** : affiche des informations sur le système de fichiers.
- **e2fsck**, **fsck.ext2**, **fsck** : vérifie et répare le système de fichiers.
- **e2image** : crée l'image d'un système de fichiers ext2 dans un fichier standard.
- **e2label** : change le libellé (étiquette) du système de fichiers.
- **mke2fs**, **mkfs.ext2** : crée le système de fichiers sur un périphérique de type bloc.
- **resize2fs** : redimensionne le système de fichiers (utile lors de l'agrandissement d'un volume logique LVM par exemple).
- **tune2fs** : modifie les paramètres du système de fichiers.
- **chattr**, **lsattr** : positionne et visualise les attributs spécifiques à ext2 sur les fichiers.

3. ext3

Le système de fichiers ext3 est l'évolution de ext2 ; il ajoute la journalisation à ce système de fichiers. C'est le format de système de fichiers proposé par défaut lors du partitionnement à l'installation de Debian Sarge.

Entièrement compatible avec ext2, il est possible de convertir directement un système de fichiers ext2 vers ext3 et inversement. Le système de fichiers peut même être monté en tant que ext2 au besoin.

Néanmoins, cette compatibilité ascendante implique que le système de fichiers présente les mêmes limitations que ext2. Pire encore, l'ajout de la journalisation induit une baisse de performances par rapport à ce dernier.

Les systèmes ext2 et ext3 étant très proches, les mêmes commandes sont utilisées.

Il faudra cependant invoquer la commande **mkfs.ext3** (ou **mke2fs** avec l'option **-j**) pour mettre en place la journalisation lors de la création du système de fichiers. Si le système de fichiers existe déjà en ext2, il peut être converti en ext3 avec **tune2fs -j** :

```
[root]# tune2fs -j /dev/sdc1
tune2fs 1.37 (21-Mar-2005)
Creating journal inode: done
This filesystem will be automatically checked every 26 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

4. ReiserFS

Disponible sous Debian Etch en tant que module noyau, c'est aussi un système de fichiers journalisé.

Il utilise des arborescences de répertoires équilibrés au lieu de listes linéaires de répertoires, ce qui procure de meilleures performances lorsque les répertoires contiennent des milliers de fichiers.

Cependant le noyau Linux ne supporte pas actuellement la gestion des quotas avec ce système de fichiers.

Les commandes principales du paquetage *reiserfs-utils* sont :

- **mkreiserfs** : crée le système de fichiers.
- **debugreiserfs** : corrige le système de fichiers.
- **reiserfsck** : vérifie le système de fichiers.
- **resize_reiserfs** : redimensionne le système de fichiers.

> Dans l'évolution du système de fichiers ReiserFS, une version majeure nommée Reiser4 est sortie en août 2004, avec des performances encore accrues et un fonctionnement complètement différent. Elle apporte des nouveautés comme les transactions atomiques, un mécanisme nommé le "dancing trees" qui permet de gérer des fichiers de grande taille et, de plus, Reiser4 intègre un mécanisme de plugins permettant d'en faciliter son évolution.

5. ramfs

Disques virtuels

Nous avons abordé au chapitre précédent la notion de disques virtuels */dev/ram** créés en mémoire vive dont la taille est fixée en paramètre du noyau ; ces périphériques pouvant être manipulés comme n'importe quelle autre unité de type bloc. Par exemple :

```
[root]# mke2fs /dev/ram0
mke2fs 1.37 (21-Mar-2005)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
2048 inodes, 8192 blocks
409 blocks (4.99%) reserved for the super user
First data block=1
1 block group
8192 blocks per group, 8192 fragments per group
2048 inodes per group
Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 31 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
[root]# mkdir /mnt/virt1
[root]# mount /dev/ram0 /mnt/virt1
[root]# ls /mnt/virt1
lost+found
```

La commande **mount** est abordée un peu plus loin dans le chapitre.

Une fois ces disques formatés avec un système de fichiers donné, ils occupent un espace mémoire égal à leur capacité.

Système de fichiers ramfs

Depuis les noyaux Linux en version 2.4, une seconde approche des espaces de stockage en mémoire vive existe. Elle fonctionne directement au niveau du système de fichiers virtuel (VFS) et non plus au niveau des périphériques de type bloc.

Ainsi, pour obtenir tel espace de stockage, il suffit de monter un système de fichiers de type ramfs sur un répertoire vide (*/mnt/virt2*) de l'arborescence, sans préciser de périphérique de type bloc (**none**) :

```
[root]# mkdir /mnt/
mont3  virt1  virt2
[root]# rmdir /mnt/virt2
[root]# mkdir /mnt/virt2
[root]# mount -t ramfs none /mnt/virt2
[root]# ls /mnt/virt2
[root]# mount
/dev/sda6 on / type ext3 (rw,errors=remount-ro)
...
/dev/ram0 on /mnt/virt1 type ext2 (rw)
none on /mnt/virt2 type ramfs (rw)
```

L'avantage de ramfs par rapport aux disques virtuels */dev/ram** est de n'utiliser que la quantité de mémoire vive correspondant à la taille des données stockées sur le système de fichiers. Si la taille du système de fichiers ramfs est fixée à 20 Mo et qu'un seul fichier de 3 Ko soit stocké dessus, seuls 3 Ko de mémoire seront employés. Une fois le système de fichiers démonté, la mémoire utilisée est entièrement libérée.

Par contre, ramfs étant un système de fichiers et non un périphérique de type bloc, il n'est pas possible d'utiliser cette fonctionnalité pour créer un système de fichiers ext2 en mémoire.

© Editions ENI - All rights reserved

6. Autres systèmes de fichiers

Linux supporte encore un grand nombre de systèmes de fichiers :

- Unix : minix, ext, xiafs ;
- Unix journalisés : xfs, JFS ;
- distribués : GPFS ;
- Microsoft : FAT12, FAT16, VFAT, NTFS ;
- Cdrom : ISO9660 ;
- autres OS : HPFS (OS/2), HFS (Macintosh), BeFS (BeOS), AFFS (Amiga) ;
- réseau : NFS, SMBFS, NCPFS ;
- spéciaux : /proc, /sys, devfs ;
- autre : UMSDOS (système de fichiers Unix sous DOS), etc.

D. Montages

Nous avons vu que pour accéder au contenu d'un système de fichiers, il fallait "accrocher" celui-ci à l'arborescence Linux. Cette opération se nomme "montage" et s'effectue à l'aide de la commande **mount**. Sa syntaxe est la suivante :

```
mount [-t <type>] [-o options[,...]] <périph.> <rep.>
```

Les valeurs permettant de spécifier le type de système de fichiers à monter sont mentionnées dans la page de manuel et dans les sources du noyau ; les plus utilisées sont **ext2**, **ext3**, **reiserfs**, **vfat**, **iso9660**, **ramfs**, **nfs**, **smbfs**. Si l'option **-t** n'est pas mentionnée ou si le type **auto** est précisé, la commande essaiera de déterminer automatiquement le type du système de fichiers à monter.

Les options principales introduites par **-o** sont :

- **auto/noauto** : peut être monté ou non par l'option **-a**.
- **defaults** : utilise les options par défaut : **rw**, **suid**, **dev**, **exec**, **auto**, **nouser** et **async**.
- **dev/nodev** : interprète ou non les fichiers spéciaux de périphériques présents sur le système de fichiers.
- **exec/noexec** : permet ou non l'exécution de fichiers binaires.
- **suid/nosuid** : tient compte ou non des bits SUID et SGID.
- **user/nouser** : autorise ou non les utilisateurs ordinaires à monter le système de fichiers.
- **ro/rw** : monte le système en lecture seule ou en lecture écriture.
- **async/sync** : toutes les entrées/sorties sur le système de fichiers seront asynchrones ou synchrones (avec ou sans cache disque).
- **remount** : tente de remonter un système de fichiers déjà monté. Cette option est utilisée principalement pour modifier les caractéristiques du montage actuel ; par exemple pour passer de lecture seule à lecture écriture.

> L'utilisation de l'option **user** entraîne automatiquement l'utilisation des options **noexec**, **nosuid** et **nodev**, à moins que celles-ci soient explicitement surchargées.

Il existe encore un très grand nombre d'options spécifiques aux systèmes de fichiers ; leur description est disponible dans la page de manuel de la commande **mount**.

Enfin, le nom du périphérique contenant le système de fichiers doit être spécifié, ainsi que le point de montage qui est caractérisé par un répertoire vide existant.

> Le montage d'un système de fichiers sur un répertoire non vide a pour effet de masquer le contenu de celui-ci jusqu'à son démontage.

La commande **mount** employée seule indiquera tous les montages effectifs sur le système. Les mêmes informations sont disponibles dans le fichier */etc/mtab*.

Par exemple, pour monter le système de fichiers ext3 créé précédemment sur */dev/sdc1* :

```
[root]# mount
/dev/sda6 on / type ext3 (rw,errors=remount-ro)
...
[root]# mkdir /mnt/mont3
[root]# mount -t ext3 /dev/sdc1 /mnt/mont3
[root]# mount
/dev/sda6 on / type ext3 (rw,errors=remount-ro)
...
/dev/sdc1 on /mnt/mont3 type ext3 (rw)
```

Démontage

En plus de rendre inaccessible le contenu du système de fichiers, l'action de démontage permet de synchroniser les données en mémoire (cache disque) avec celles réellement inscrites sur le média. Le démontage des systèmes de fichiers est une opération majeure effectuée à l'arrêt du système.

Pour inscrire immédiatement de façon ponctuelle les données du cache sur les disques, il existe la commande **sync**.

Un système de fichiers est démonté avec :

umount <périph.> ou **umount <rep.>**

Ainsi :

```
[root]# umount /mnt/mont3 /mnt/virt1 /mnt/virt2
[root]# mount
/dev/sda6 on / type ext3 (rw,errors=remount-ro)
```

Un système de fichiers ne peut être démonté que si aucun processus n'y accède en même temps ; si un utilisateur se trouve dans un répertoire du système de fichiers par exemple, l'administrateur doit rechercher le processus associé et le terminer avant de pouvoir démonter le système de fichiers. La recherche des processus manipulant un fichier est réalisée avec **fuser**.

Par exemple, pour connaître tous les processus d'un système Linux utilisant la partition racine montée sur */dev/sda6* :

```
[root]# fuser -vm /dev/sda6

                     USER        PID ACCESS COMMAND
/dev/sda6            root          1 .rce.  init
                     root          2 .rc..  migration/0
                     root          3 .rc..  ksoftirqd/0
                     root          4 .rc..  migration/1
                     root          5 .rc..  ksoftirqd/1
                     root          6 .rc..  events/0
                     root          7 .rc..  events/1
                     root          8 .rc..  khelper
                     root          9 .rc..  kacpid
                     root         44 .rc..  kblockd/0
                     root         45 .rc..  kblockd/1
                     root         55 .rc..  pdflush
                     root         56 .rc..  pdflush
                     root         57 .rc..  kswapd0
...
```

Les accès sont indiqués par différentes lettres, à savoir :

- **c** : répertoire courant ;
- **e** : programme en cours d'exécution ;
- **f** : fichier ouvert (omis dans l'affichage par défaut de la commande) ;
- **r** : répertoire racine ;
- **m** : bibliothèque partagée ou fichier copié en mémoire.

© Editions ENI - All rights reserved

Dans le même ordre d'idée, il existe aussi l'outil **lsof**. Invoqué sans option ni argument, il permet de lister tous les fichiers ouverts sur le système :

```
[root]# lsof
COMMAND    PID   USER   FD  TYPE  DEVICE    SIZE    NODE NAME
init         1   root  cwd   DIR     8,6    4096       2 /
init         1   root  rtd   DIR     8,6    4096       2 /
init         1   root  txt   REG     8,6   31432  667831 /sbin/init
init         1   root  mem   REG     8,6   90248   81457 /lib/ld-2.3.2.so
init         1   root  mem   REG     8,6 1254468   81486 /lib/tls/libc-2.3.2.so
init         1   root   10u FIFO    0,11             4062 /dev/initctl
migration    2   root  cwd   DIR     8,6    4096       2 /
migration    2   root  rtd   DIR     8,6    4096       2 /
migration    2   root  txt unknown /proc/2/exe
...
```

Cette commande propose un très grand nombre d'options pour filtrer les fichiers ouverts. Les critères qui nous intéressent le plus ici sont le nom de l'utilisateur et le PID d'un processus.

Par exemple, pour afficher tous les fichiers ouverts par les utilisateurs **nicolas** (UID 1002) et **franck** (UID 1003), on peut écrire :

```
lsof -u 1002,franck
```

La syntaxe pour retrouver tous les fichiers ouverts par les processus 542 et 8754 est :

```
lsof -p 542, 8754
```

Au contraire, pour lister tous les processus accédant à un fichier ou un système de fichiers, il suffit de passer le nom du fichier correspondant sur la ligne de commandes.

Par exemple, pour connaître les processus qui accèdent à la partition `/dev/sdb3` montée sur `/home` :

```
[root]# lsof /dev/sdb3
COMMAND  PID USER   FD   TYPE DEVICE SIZE   NODE NAME
bash    8471 yann  cwd    DIR   8,19 4096 289729 /home/yann
bash    8481 yann  cwd    DIR   8,19 4096 289729 /home/yann
```

1. Montages prédéfinis

Il est possible de prédéfinir les montages des différents systèmes de fichiers grâce au fichier */etc/fstab*. Ceci présente deux avantages :

- Permettre le montage de systèmes de fichiers au démarrage de la machine. En effet, la commande **mount -a** fait partie des scripts d'initialisation du système.
- Alléger la ligne de commande **mount**. Si toutes les informations nécessaires au montage d'un système de fichiers se trouvent dans ce fichier, seul le nom du périphérique ou du point de montage suffira pour monter le système de fichiers associé.

Voici un exemple de ce fichier :

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
/dev/sda6       /               ext3    defaults,errors=remount-ro 0 1
/dev/sda2       /boot           ext3    defaults        0       2
/dev/sdb3       /home           ext3    defaults        0       2
/dev/sda5       /tmp            ext3    defaults  <+>     0       2
/dev/sdb1       /var            ext3    defaults        0       2
/dev/sdb2       none            swap    sw              0       0
/dev/scd1       /media/cdrom0   iso9660 ro,user,noauto  0       0
/dev/scd0       /media/cdrom1   iso9660 ro,user,noauto  0       0
/dev/fd0        /media/floppy0  auto    rw,user,noauto  0       0
```

Les différents champs représentent :

- le nom du périphérique ou son libellé ;
- le point de montage ;
- le type du système de fichiers ;
- les différentes options de **mount** ;
- une indication pour la commande de sauvegarde **dump** ;
- l'ordre de vérification du système de fichiers au démarrage par **fsck**.

Service HAL

Le programme **fstab-sync** met à jour le fichier */etc/fstab* et crée/supprime les répertoires de montages correspondants dans */media* suivant les informations renvoyées par le démon HAL (*Hardware Abstraction Layer*) et géré par le service udev ; il n'est pas invoqué directement sur la ligne de commandes mais par un des scripts présents dans le répertoire de configuration du service HAL.

Le démon HAL ayant pour fonction de détecter les modifications matérielles de la machine, cela permet de garder le fichier */etc/fstab* cohérent avec le matériel, principalement avec les périphériques amovibles (lecteurs de disquettes et lecteurs optiques, périphériques de stockage USB...).

Les lignes du fichier */etc/fstab* gérées par l'outil **fstab-sync** sont repérées par la pseudo-option **managed**, ce qui permet de les identifier facilement.

2. Automontages

L'automontage consiste à monter (ou démonter) certains systèmes de fichiers automatiquement par un processus démon. Si un système de fichiers n'est pas monté et qu'un utilisateur tente d'y accéder, il sera automatiquement (re)monté. Ceci est particulièrement intéressant dans des environnements réseau où les systèmes de fichiers ne sont pas constamment accessibles. C'est aussi utile pour les périphériques amovibles comme les CD-Rom et DVD-Rom.

Il existe deux types d'automonteurs sous Linux : AMD et autofs. Le premier est indépendant du noyau et le second interagit avec le VFS du noyau.

Nous traiterons ici d'autofs disponible sous Debian Etch dans le paquetage du même nom. Les noyaux fournis par Debian Etch ont le module autofs4 compilé et pris en compte.

Configuration

Il existe au moins deux fichiers de configuration pour paramétrer ce service.

Le premier, */etc/auto.master*, définit des ensembles de montages. Une ligne de ce fichier ressemble à :

```
/misc   /etc/auto.misc  --timeout=60
```

La signification des trois champs est la suivante :

- **/misc** correspond à un répertoire existant dans l'arborescence qui accueillera les différents montages.
- **/etc/auto.misc** est le fichier "map" définissant un ensemble de montages.
- **--timeout=60** signifie que les systèmes de fichiers correspondants seront démontés automatiquement au bout de 60 secondes.

Le fichier */etc/auto.misc* ressemble à :

```
cd              -fstype=iso9660,ro,nosuid,nodev      :/dev/scd1
floppy          -fstype=auto                         :/dev/fd0
```

Ce second fichier contient les informations nécessaires à chaque montage :

- Une "clé" : cette clé associée au répertoire de base */misc* constitue le point de montage du système de fichiers.
- Les options de montage identiques à celles de la commande **mount**.
- Le nom du périphérique.

© Editions ENI - All rights reserved

Lancement du service

Le script de lancement de ce service est :

```
/etc/init.d/autofs start
```

Maintenant, un simple accès au répertoire */misc/cd* montera automatiquement le média se trouvant dans le lecteur de CD-Rom en ce point de montage.

Soit l'exécution suivante avec les fichiers de configuration précédents :

```
[root]# /etc/init.d/autofs status
Configured Mount Points:
------------------------
/usr/sbin/automount --timeout=60 /misc file /etc/auto.misc

Active Mount Points:
--------------------
/usr/sbin/automount -pid-file=/var/run/autofs/_misc.pid --timeout=60 /misc file
/etc/auto.misc
[root]# mount
...
automount(pid6933) on /misc type autofs (rw,fd=4,pgrp=6933,minproto=2,
maxproto=4)
[root]# ls /misc/cd
README.html          README.txt   debian  install     pics
README.mirrors.html  autorun.bat  dists   isolinux    pool
README.mirrors.txt   autorun.inf  doc     md5sum.txt  tools
[root]# mount
...
automount(pid6933) on /misc type autofs (rw,fd=4,pgrp=6933,minproto=2,
maxproto=4)
/dev/scd1 on /misc/cd type iso9660 (ro,nosuid,nodev)
```

E. Quotas

Les quotas permettent à root de limiter l'utilisation des systèmes de fichiers à ses administrés. Il peut restreindre, par utilisateur et/ou par groupe, le nombre de fichiers (ou inodes) et la quantité de données (ou blocs de données) enregistrés sur le disque. Pour cela, il peut fixer :

- Une limite "dure" ("hard") : tout dépassement de cette limite sera refusé par le système et entraînera une erreur d'écriture.
- Une limite "douce" ("soft") : le dépassement de cette limite entraînera la délivrance d'un avertissement à l'utilisateur et le décompte d'un délai de grâce. Au-delà de ce sursis, l'avertissement sera remplacé par une erreur comme pour la limite dure.

Les quotas sont définis par système de fichiers. Il est donc nécessaire de distinguer chaque répertoire faisant l'objet de restrictions spécifiques en créant plusieurs systèmes de fichiers. Par exemple, si les utilisateurs du système ont droit à 200 Mo d'espace dans leur répertoire personnel mais seulement à une boîte aux lettres contenant au maximum 50 Mo, il faudra créer un système de fichiers pour le répertoire */home* et un autre pour le répertoire */var/spool/mail*.

Le noyau Linux de Debian Etch intègre d'origine le support des quotas. Il suffit de monter le système de fichiers désiré avec le support des quotas, installer le paquetage du même nom, de configurer les limites voulues et d'activer la gestion des quotas.

1. Montage avec support des quotas

Le montage d'une partition avec les options **usrquota** et **grpquota** positionne les quotas pour les utilisateurs et les groupes. Ces options peuvent être spécifiées sur la ligne de commandes **mount** ou dans le fichier */etc/fstab* pour que ce soit permanent et automatique au démarrage.

Par exemple, pour imposer les quotas sur le système de fichiers */dev/sdc1* monté sur */limited*, il faudra ajouter ou modifier la ligne du fichier */etc/fstab* correspondante :

```
/dev/sdc1       /limited      ext3      defaults,usrquota,grpquota      0 0
```

2. Configuration

Avant toute chose, il faut initialiser les fichiers de configuration des quotas.
Ces fichiers se trouvent dans la racine du système de fichiers et se nomment *aquota.user* et *aquota.group*.

La commande **quotacheck** permet de vérifier et de réparer ces fichiers ; elle peut donc être utilisée pour initialiser ces fichiers avec les options **-u** (vérification des quotas par utilisateur), **-g** (vérification des quotas par groupe) et **-v** (mode verbeux de la commande) :

```
[root]# mount
...
/dev/sdc1 on /limited type ext3 (rw,usrquota,grpquota)
[root]# cd /limited
[root]# quotacheck -ugv /dev/sdc1
quotacheck: Scanning /dev/sdc1 [/limited] quotacheck: Cannot stat old user quota
file: No such file or directory
quotacheck: Cannot stat old group quota file: No such file or directory
quotacheck: Cannot stat old user quota file: No such file or directory
quotacheck: Cannot stat old group quota file: No such file or directory
done
quotacheck: Checked 3 directories and 1 files
quotacheck: Old file not found.
quotacheck: Old file not found.
[root]# ls
aquota.group  aquota.user  lost+found
```

Les anciennes implémentations des quotas sous Linux utilisaient les fichiers suivants : *quota.user* et *group.user*. Ceux-ci sont toujours supportés et peuvent être convertis avec la commande **convertquota**.

S'en suit le paramétrage des quotas avec **edquota** ; plusieurs options :

- **-u** : édite les quotas d'un utilisateur donné.
- **-g** : édite les quotas associés à un groupe.
- **-f** : édite les quotas uniquement pour le système de fichiers spécifié. Sans cette option, les modifications seront prises en compte pour tous les systèmes de fichiers mettant en œuvre les quotas.
- **-t** : permet de spécifier le délai associé à la limite douce.

Par exemple :

```
root# edquota -u nicolas -f /dev/sdc1
Disk quotas for user nicolas (uid 1002):
  Filesystem      blocks     soft     hard     inodes     soft     hard
  /dev/sdc1            0    40960    51200          0      600      800
root# edquota -g formation -f /dev/sdc1
Disk quotas for group formation (gid 1001):
  Filesystem      blocks     soft     hard     inodes     soft     hard
  /dev/sdc1            0    51200    61440          0     3000     5000
[root]# edquota -f /dev/sdc1 -t
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
  Filesystem             Block grace period     Inode grace period
  /dev/sdc1                     7days                  7days
```

3. Activation des quotas

Enfin, il faudra activer la gestion des quotas avec **quotaon** pour qu'ils soient pris en compte.

La commande **quotaon -aug** faisant partie des scripts d'initialisation du système Debian Etch (fichier */etc/rcs.d/S35quota*), le support des quotas est activé automatiquement pour tous les systèmes de fichiers ayant l'option **usrquota** et/ou **grpquota** dans */etc/fstab* (option **-a** de **quotaon**), au niveau des utilisateurs (option **-u**) et des groupes (option **-g**).

© Editions ENI - All rights reserved

4. Informations sur les quotas

Deux commandes sont disponibles pour obtenir des informations sur les quotas :

- **quota** : retourne l'occupation actuelle des fichiers de l'utilisateur qui invoque la commande ainsi que les quotas par système de fichiers.
- **repquota** : réalise un état de l'utilisation du système de fichiers dont le point de montage est passé en argument ; le rapport fourni détaille l'utilisation de chaque utilisateur par défaut ou de chaque groupe avec l'option **-g**. Les informations affichées concernent uniquement les utilisateurs ou groupes employant le système de fichiers, qu'ils aient un quota fixé ou non. Seul **root** peut lancer cette commande.

Par exemple :

```
[root]# repquota /limited
*** Report for user quotas on device /dev/hdb1
Block grace time: 7days; Inode grace time: 7days
                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --   11139       0       0              87     0     0
nicolas   --       8   40960   51200               3   600   800
franck    --       3       0       0              1     0     0
stephane  -    6308    6000    7000  6days     1914     0     0
willy     --   41929       0       0           2136  2200  3000

[root]# repquota -g /limited
*** Report for group quotas on device /dev/hdb1
Block grace time: 7days; Inode grace time: 7days
                        Block limits                File limits
Group           used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --   11139       0       0              87     0     0
formation --      11   51200   61440               4  3000  5000
technique --   48237       0       0           4050     0     0
```

Premièrement, le total des fichiers pour le groupe formation est la somme des fichiers des utilisateurs **franck** et **nicolas** qui ont ce groupe comme groupe principal (cf. chapitre 5 - Gestion des utilisateurs).

Ensuite, l'utilisateur **stephane** a dépassé son quota soft et il ne pourra plus créer de fichier d'ici six jours. S'il tente de dépasser son quota hard, une erreur est renvoyée :

```
[stephane]$ quota
Disk quotas for user stephane (uid 502):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
      /dev/hdb1    6308*   6000    7000   6days    1914       0       0
[stephane]$ cp /tmp/grosfichier /limited
hdb1: write failed, user block limit reached.
cp: écriture de `/limited/grosfichier': Débordement du quota d'espace disque
[stephane]$ quota
Disk quotas for user stephane (uid 502):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
      /dev/hdb1    6997*   6000    7000   6days    1915       0       0
```

Enfin, l'utilisateur **willy** est en passe de dépasser son quota soft en nombre de fichiers ; il peut dépasser ce quota tant qu'il reste au-dessous de la limite hard mais un message d'avertissement est affiché.

```
[willy]$ quota
Disk quotas for user willy (uid 504):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
      /dev/hdb1   41929       0       0            2136    2200    3000
[willy]$ cp -R /lib /limited/lib
hdb1: warning, user file quota exceeded.

[willy]$ quota
Disk quotas for user willy (uid 504):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
      /dev/hdb1   52277       0       0           2605*   2200    3000   7 days
```

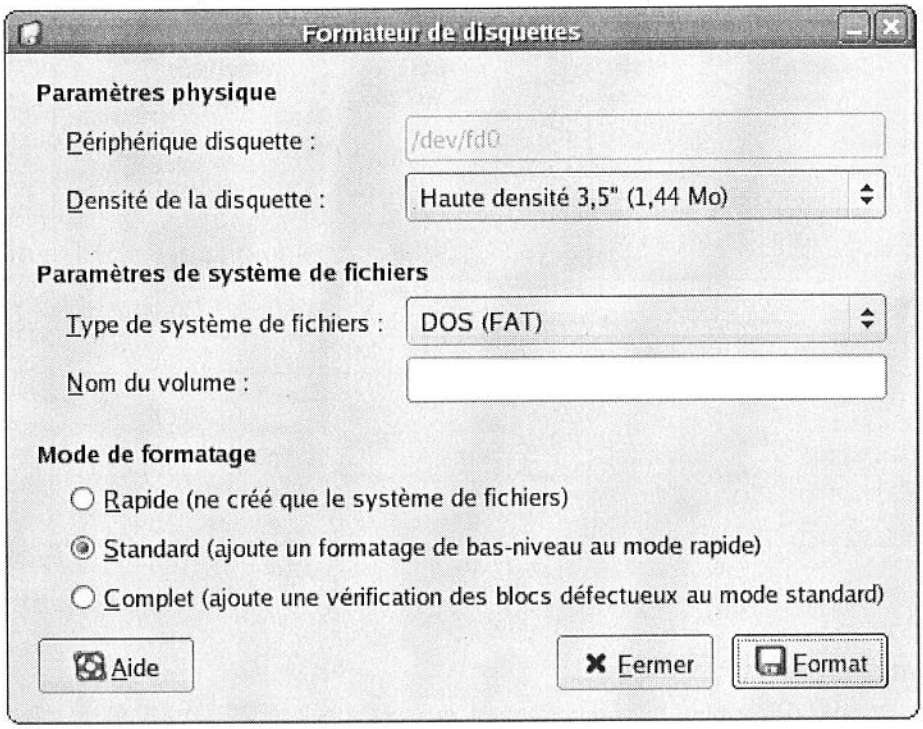
F. Outils graphiques

Ce paragraphe présente les principaux outils graphiques fournis avec la distribution Debian Etch, relatifs à la création de systèmes de fichiers et à leur montage.

1. Formatage de disquettes

gfloppy

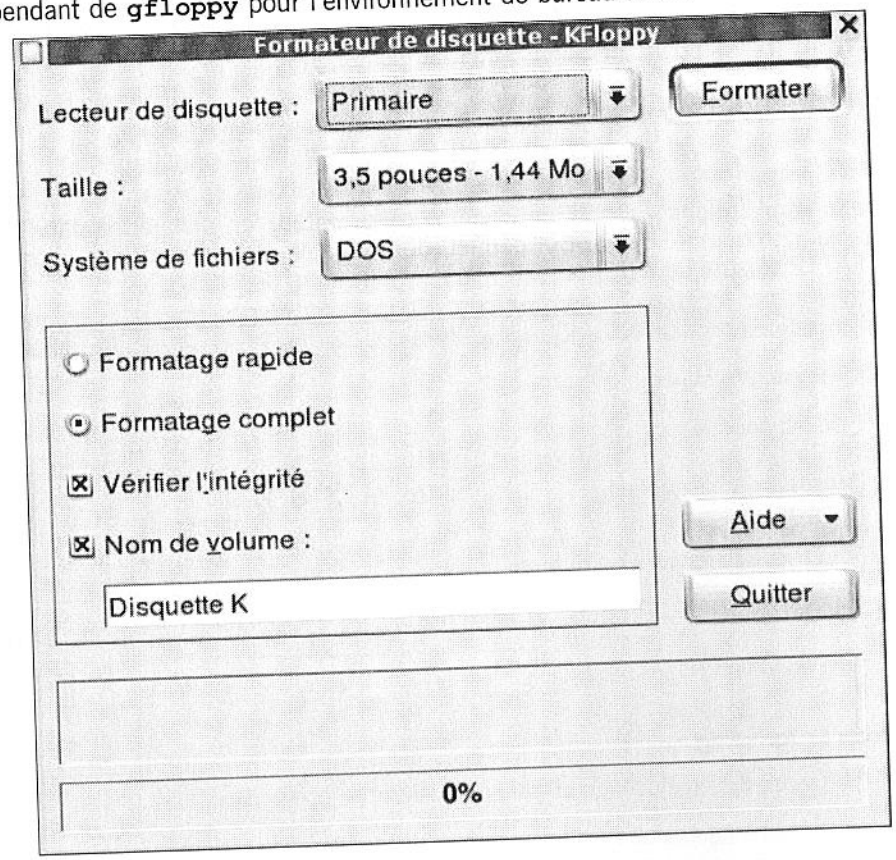
Accessible via le menu **Applications - Outils de système - Formateur de disquettes** dans l'environnement de bureau GNOME, la commande **`gfloppy`** permet de formater simplement des disquettes avec un système de fichiers FAT12 (disquette DOS) ou ext2, suivant plusieurs densités :



© Editions ENI - All rights reserved

kfloppy

Cet outil est le pendant de **gfloppy** pour l'environnement de bureau KDE :



2. Montages

gnome-volume-properties

Cette application GNOME détermine le comportement du système lors de l'insertion de médias amovibl (automontages et actions à entreprendre) ; elle se trouve dans le menu **Environnement de bureau - Pré rences - Périphériques et média amovibles** :