Les Systèmes de Fichiers

Objectifs

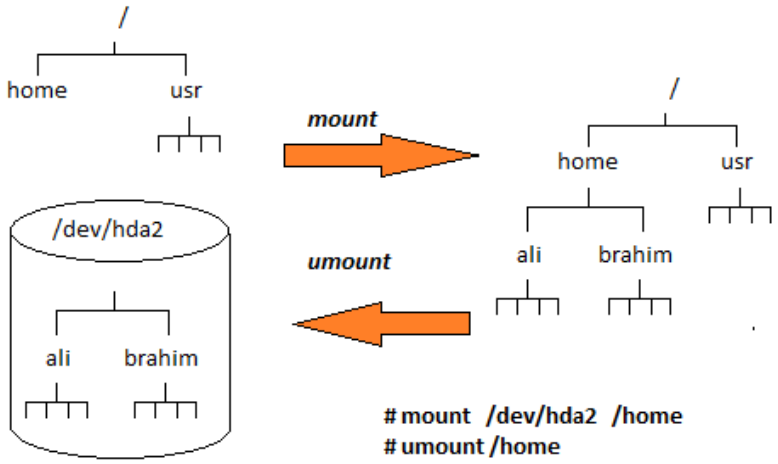
A l'issue de ce chapitre, vous connaîtrez la structure des systèmes de fichiers, leur gestion. Vous serez capable de créer, monter, démonter un système de fichiers, et automatiser le montage.

Contenu

Montage et démontage de fichiers, Les inodes
Les systèmes de fichiers journalisés
La gestion de l'espace disque, la commande df
Commandes de gestion des systèmes de fichiers
L'automatisation du montage, La gestion des quotas

1- Rappels de cours:

1.1- Montage des systèmes de fichiers



Le montage d'un système de fichiers consiste à attacher la racine de l'arbre du système de fichiers à un répertoire d'un système de fichiers déjà actif. Cette opération, qui s'appelle le montage du système de fichiers, est réalisée par la commande mount.
La suppression du lien entre le répertoire de montage et le système de fichiers est effectuée par la commande umount. Les fichiers d'un système de fichiers ne sont accessibles, par les commandes usuelles (cp, rm, mv, cat, ...), que s'il est monté. Le système Linux mémorise les paramètres du montage des systèmes de fichiers dans le fichier */etc/mnttab*.
Le disque */dev/hda2* contient les répertoires de connexion des utilisateurs. Le répertoire */home* est un répertoire vide de l'arborescence active constituée des systèmes de fichiers déjà montés. La commande de montage, **mount /dev/hda2 /home**, attache la racine du système de fichiers de */dev/hda2* au répertoire */home*. La racine du système de fichiers coïncide maintenant avec le chemin d'accès au répertoire de montage /home. Quand l'opération de démontage **umount /dev/hda2** a été réalisée, les fichiers des répertoires **ali** et **brahim** sont de nouveau inaccessibles, le répertoire /home est vide.
Remarque:
Si le répertoire de montage n'est pas vide au moment de l'exécution de la commande mount, les fichiers qu'il contient sont cachés jusqu'au démontage.

Lors du démarrage du système, le disque qui contient le système de fichiers principal, celui où se trouve le fichier racine l, doit être connu pour être automatiquement monté. Ce disque est un paramètre du démarrage du système. Il est défini par la variable root dans le fichier */etc/lilo.conf*, le fichier de configuration du chargeur lilo. En l'absence de cette variable, c'est le disque indiqué dans l'image même du noyau qui sera monté comme système de fichier racine.

1.2- Structure du système de fichiers ext2

- ***Structure du système de fichiers***

Un système de fichiers est une structure de données.
La commande mkfs, qui crée un système de fichiers, inscrit cette structure de données dans une partition.
Tous les systèmes de fichiers comportent au moins trois tables système:
• Le super bloc qui contient les informations clés concernant le système de fichiers.
• La table des inodes, c'est-à-dire la table des descripteurs des fichiers. Chaque fichier est identifié de manière unique par le numéro de l'inode qui le décrit.
• Les répertoires qui assurent une correspondance entre un nom de fichier et un numéro d'inode.

- ***Structure d'un inode***

Le terme inode désigne le descripteur d'un fichier. Il contient les attributs du fichier, ceux affichés par la commande Is -I, et une table d'accès aux blocs de données. Il existe une table d'inodes par disque et l'espace qu'elle occupe est réservé à la création du système de fichiers sur le disque. La taille de la table des inodes est donc un paramètre statique important d'un système de fichiers, car elle fige le nombre de fichiers que l'on peut au plus créer sur le disque.
Remarque:
Il faut se souvenir que tous les fichiers, y compris les fichiers spéciaux (périphériques, tubes nommés, ...) occupent un inode. L'administrateur n'a pas besoin de mémoriser la structure interne d'un inode. Sa connaissance permet cependant de mieux comprendre comment le système UNIX gère les fichiers. Il est par contre indispensable de se souvenir que le numéro d'inode d'un fichier, pour un disque donné, est l'unique moyen d'identifier sans ambiguité un fichier. La commande 'ls -i' permet de connaitre le numéro de i-node d'un fichier.

La figure ci-dessous présente la structure de inode d'un système de fichiers de type « ext2 ». C'est actuellement le système de fichiers le plus performant sous Linux. C'est, du reste, le type de système de fichiers par défaut pour la commande mkfs qui crée les systèmes de fichiers.

Dans le schéma, le sigle "BD" désigne un bloc de données de 1, 2 ou 4 ko et le sigle "BA" un bloc d'adresses.

L'inode contient l'adresse de douze blocs de données. La treizième entrée de l'inode contient l'adresse d'un bloc d'adresses de blocs de données (indirection de premier niveau). La quatorzième entrée contient l'adresse d'un bloc d'adresses de blocs d'adresses. Cela constitue une double indirection. La quinzième entrée définit une triple indirection.
Une entrée de répertoire contient les informations suivantes:
• Le numéro de l'inode.
• La longueur en octets de l'entrée.
• La longueur en caractères du nom de fichier.
• Le nom du fichier.
Le nombre de liens matériels est inscrit dans l'inode. Il est décrémenté à chaque suppression d'un lien et l'inode n'est libéré qu'à la suppression du dernier lien qui le référence.

Remarques:

• Les temps sont exprimés en secondes depuis le premier janvier 1970 à 0 heure, heure GMT.

• La destruction d'un fichier consiste à libérer l'inode et à restituer les blocs d'adresses et de données à l'ensemble des blocs libres.

	Droits d'accès et type du fichier				
	UID du propriétaire				
	GID du groupe du fichier				
	Taille en octets				
	Date et heure de création				
	Date et heure de dernière modification				
	Date et heure de dernier accès				
	Date et heure de suppression				
	Nombre de liens matériels				
	Nombre de blocs				
	Nombre de fragments				
	Drapeaux				
	Réservé				
	Fichier ACL				
	Répertoire ACL				
	Adresse de fragments				
	Nombre de fragments				
1	Adresse du premier bloc de données	BD			
2		BD			
3		BD			
4		BD			
5		BD			
6		BD			
7		BD			
8		BD			
9		BD			
10		BD			
11		BD			
12	Adresse du douzième bloc de données	BD			
13	Adresse du bloc d'indirection de niveau 1	BA	BD		
14	Adresse du bloc d'indirection de niveau 2	BA	BA	BD	
15	Adresse du bloc d'indirection de niveau 3	BA	BA	BA	BD

- ***La structure du système de fichiers ext2***

Le système de fichiers « ext2 » est le plus utilisé sous Linux. Il dérive du système de fichiers ffs (« Fast File System ») créé à l'université de Berkeley.

Sa structure est la suivante:

• Un secteur de boot de 512 octets qui ne fait pas partie du système de fichiers. Ce secteur peut être vide.

• Une suite d'ensemble de blocs.

Chaque ensemble de blocs contient les informations suivantes:

• Une copie du super bloc.

• Une table de descripteurs qui indique l'emplacement des tables suivantes.

• La table d'allocation des blocs de l'ensemble courant sous forme de bitmaps.

• La table d'allocation des inodes de l'ensemble courant sous forme de bitmaps.

• La table des inodes de l'ensemble courant.

• Les blocs de données des fichiers de l'ensemble courant.

Le super bloc contient notamment les informations suivantes:

• La taille des blocs.

• La taille en bloc du système de fichiers.

• Le nombre de blocs libres.

• Le nombre de blocs réservés à root.

• Le nombre de inodes

• Le nombre de inodes libres
• L'état du système de fichiers: monté ou démonté.
• Le nombre de blocs par ensemble de blocs.
• Le nombre d'inodes par ensemble de blocs.
• Le nombre maximal de montage avant de réaliser un fsck automatique.
• Le nombre de montages réalisés depuis le dernier fsck.
• Date du dernier fsck.
• Temps maximum entre deux fsck.
• La table des inodes.
Un inode a été décrit dans le paragraphe précédent.
Dans la table des inodes, plusieurs entrées sont réservées, notamment:
• Le premièr inode qui mémorise les « bads blocs» du système de fichiers.
• Le second inode qui est celui du répertoire racine du système de fichiers.
Remarque:
Les informations présentées dans ce chapitre sont décrites dans les fichiers d'inclusion du répertoire /usr/include/linux : ext2Js.h, ext2JsJh et ext2Js_sb.h

- ***Les différents types de SdF***

- minix : le premier FS utilisé par Linux
- ext2 : Le FS standard du système Linux
- msdos : Le FS FAT16 de MS-DOS et Windows
- vfat : Le FS FAT32 de Windows
- sysv : Le FS d'UNIX System V
- smb : Le FS utilisant le protocole SMB de Microsoft
- nfs : Le FS réseau de Sun (Network File System)
- Iso 9660: Le FS utilisé par les CD-ROM

Remarques:
• Le problème du choix du type de système de fichiers à installer sur un nouveau disque est souvent un faux problème. Le système Linux privilégie un choix, le système ext2.
• Le montage d'un système de fichiers sur un système de fichiers d'un type différent est possible et complètement transparent pour l'administrateur.
Les paramètres fondamentaux de création d'un système de fichiers sont:
• Le type, si ce n'est pas celui par défaut du système.
• La taille, souvent celle du disque où l'on crée le système de fichiers.
• Le nombre d'inodes.
• La taille d'un bloc.
• Le nombre de groupes de blocs.

1.3- La gestion de l'espace disque, les commandes **df** et **du**:

L'espace disque est une ressource précieuse, même si les capacités des disques ont considérablement évolué ces dernières années. L'administrateur doit en contrôler l'usage.
Il dispose pour cela de commandes simples et pratiques dont nous ne faisons ici que rappeler la signification et donner quelques options significatives.

La commande **df** indique l'espace libre des disques contenant des systèmes de fichiers montés. Les tailles sont affichées en kilo-octets. Les principales options sont:
-i Affiche les informations sur l'utilisation des inodes et non des blocs.
-k Affiche les tailles en kilo-octets. L'option est significative dans le cas où la variable POSIXLY_CORRECT est définie
-T Affiche également le type de système de fichiers.

La commande **du** affiche le nombre de blocs d'un kilo-octet utilisés par une arborescence qui peut coïncider avec celle d'un système de fichiers.
Les principales options sont:
-s Affiche le total seulement.
-k Affiche les tailles en kilo-octets

Remarque:
Si la variable d'environnement POSIXLY _CORRECT est définie, les commandes **du** et **df** affichent le nombre de blocs de 512 octets.
La commande **find** permet de rechercher des fichiers selon différents critères dont celui de la taille et de la date du dernier accès.

exemple:
find /home -size +100k -atime +90 -print
Il existe, en sus, des fichiers qui peuvent être détruits ou purgés, avec une périodicité qui varie selon les cas :
Les fichiers créés par les applications dans les répertoires */tmp* et */var/tmp.*
Les fichiers *core* qui sont générés quand un processus meurt sur réception de certains signaux, souvent émis par le noyau à la suite d'une anomalie de fonctionnement. Ils contiennent l'image du processus quand il est mort et sont destinés aux programmeurs qui veulent effectuer, à chaud, une analyse post-mortem de l'application. Ils peuvent être détruits sinon sans remord.
Les fichiers *log* qui enregistrent des informations relatives au suivi d'un service, d'un logiciel. Ils peuvent être purgés dès que l'administrateur a analysé leur contenu et les a archivés si nécessaire.

Remarque:
Pour vider un fichier, il suffit d'exécuter la commande:
cp */dev/null* fichier

Les principaux fichiers *log* sont contenus dans le répertoire *var log:*

Fichiers *log* du service acct (comptabilité).

Fichier *log* des principaux messages du système de log associé au démon syslogd.
Fichier *log* des tentatives de connexions infructueuses.
Fichier *log* de la messagerie.
Fichier *log* du service news (Internet).
Fichier *log* du service UUCP.

Fichier *log* du service cron.
Fichier *log,* en binaire, des commandes **init** et login. Ce fichier est exploité par la commande last.
Fichiers *log* du service Web.
Fichiers *log* du service Samba (accès aux disques Windows).
Fichier *log* des dernières connexions des utilisateurs (fichier binaire), utilisé par la commande lastlog.
Fichier *log* des erreurs de connexion du système de connexion en mode graphique.
Fichier des messages affichés au démarrage du système.

1.4- Panorama des commandes de gestion des systèmes de fichiers

- **Commandes génériques**

• mkfs Crée un FS
• mount Monte un FS
• umount Démonte un FS
• fsck Vérifie un FS
• df Espace libre
• du Espace occupé
• lsof Identifie les processus

- **Commandes propres à ext2**

• mke2fs Crée un FS
• e2fsck Vérifie un FS
• tune2fs Paramètre un FS
• dumpe2fs 1nfos sur le super bloc et les groupes de blocs
• debugfs Débogue un FS

Les commandes de gestion de systèmes de fichiers sont nombreuses. Dans Linux, on trouve deux familles de commandes :
• Les commandes génériques héritées du système UNIX, comme mkfs et fsck.
• Les commandes spécifiques à Linux et particulièrement au système de fichiers ext2.

- **mkfs**

Parmi les commandes standard d'UNIX, la commande mkfs est bien sûr la plus importante. C'est elle qui crée un système de fichiers sur un disque. Elle comporte les trois paramètres fondamentaux d'un système de fichiers:
• Le nom du disque où le système de fichiers doit être créé.
• Le type du système de fichiers à créer.
• La taille du système de fichiers à créer.
Elle peut aussi comporter d'autres options, spécifiques à un type de systèmes de fichiers, et nous renvoyons l'administrateur au manuel de référence de son système.

- **mke2fs**

La commande mke2fs permet de créer un système de fichiers de type ext2. Sa syntaxe est la suivante :
mke2fs [options] disque [taille_duFS_en_blocs]
Le disque est désigné par un fichier spécial de type bloc. A défaut de préciser la taille en blocs de 1 kilo-octet du système de fichiers, le système de fichiers occupe la totalité du disque où il est créé.
Les principales options sont:
-b taille_du_bloc Indique la taille du bloc en octets.
-i octet_par_inode Permet de calculer le nombre d'inodes. Pour chaque « octet_par_inode» sur disque, un inode est créé. La valeur minimale est de 1024. Pour une capacité disque de 100 Mo et une valeur de 4096 pour l'attribut « octet_par_inode », le nombre d'inodes sera de 25000.
-N nombre_de_inodes Indique le nombre d'inodes qu'il y aura sur le disque.
-m pourcentage_réservé Indique le pourcentage d'espace disque réservé à l'administrateur. La valeur par défaut est de 5%.
-S La commande mke2fs ne crée que le super bloc et le descripteur des groupes de blocs. Il faut ensuite exécuter
e2fsck pour les initialiser à partir des autres données du système de fichiers. Cette solution extrême n'est à mettre en oeuvre que si le super bloc et ses copies sont endommagés.

Cette option est reconnue mais elle n'est pas encore implémentée. Elle permettra de fragmenter un bloc. Cela évitera une inoccupation inutile de place disque pour les petits fichiers.
-f taille_du _fragment
Exemples
Création d'un système de fichiers de type Minix sur disquette.
mkfs -t minix /dev/fd0
480 inodes
1440 blocks
Firstdatazone= 19 (19)
Zonesize= 1024
Maxsize=268966912
Création d'un système de fichiers sur disquette avec mke2fs.
mke2fs /dev/fd0
mke2fs 1.12, 9-Jul-98 for EXT2 FS 0.5b, 95/08/09
Linux ext2 filesystem format
Filesystem label=
360 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block= 1
Block size= 1024 (log=0)
Fragment size= 1024 (log=0)
1 block group

8192 blocks per group, 8192 fragments per group

30 inodes per group

Writing inode tables: 0/ 1 done

- **mount**

La commande mount réalise l'attachement d'un système de fichiers d'un disque Linux à un répertoire d'un système de fichiers déjà monté. Elle comporte principalement le nom du disque et le chemin absolu du répertoire de montage. Sa syntaxe est la suivante:
mount [-hV]
mount -a [-furvw] [-t type]
mount [-furvw] [-0 options [,...]] périph | rép
mount [-furvw] [-t type] [-0 options] périph rép
-h Affiche un message d'aide.
-V Affiche le numéro de version.
-a Procède au montage de tous les systèmes de fichiers décrits dans le fichier */etc/fstab*
-t type Le type indique le type de système de fichiers qui réside sur le disque à monter. Il est obligatoire si ce n'est pas ext2.
-f L'exécution de la commande est simulée mais elle n'est pas exécutée.
-n Le système de fichiers monté n'est pas mémorisé dans /etc/mtab. C'est obligatoire si l'on souhaite que */etc* soit sur un disque différent *de /*.
-r Le système de fichiers est monté en lecture seulement.
-w Le système de fichiers est monté en lecture et en écriture. C'est la valeur par défaut.
-v Le mode bavard.
-0 options Les options qui suivent « -o » sont séparées par des virgules. Certaines sont communes à tous les types de systèmes de fichiers, d'autres sont spécifiques.
Options Communes
suid/nosuid Prendre en compte, ou non, les bits « s » des fichiers.
async/sync Les entrées/sorties sont asynchrones ou synchrones.
ro/rw Equivalent à « -r » et « -w ».

remount Remonter un système de fichiers déjà monté. Cette possibilité est intéressante quand on souhaite modifier les attributs de montage dynamiquement.

loop=[/dev/loop<n>] Monte un fichier ordinaire contenant l'image d'un système de fichiers, par exemple créé par dd. Le système de fichiers est associé à un périphérique */dev/loop<n>,* où <n> est compris entre 0 et 7. A défaut, c'est le premier disponible qui est utilisé.

Valides pour ext2 et ext3

noatime dans les inodes, la taille de dernière consultation n'est pas modifiée. Quand cette information a peu d'importance et qu'il existe de nombreux fichiers, souvent lus, cette option peut améliorer les performances. Cela peut être le disque contenant le courrier des utilisateurs.

Spécifiques à nfs

rsize=8192 wsize=8192 Fixe la taille des buffers pour les opérations de lecture et d'écriture à 8192 octets au lieu de 1024 par défaut. Ceci améliore la vitesse du système nfs.

Exemple:
Montage d'un système de fichiers.
mount */dev/hda3* /opt/appli
La commande mount, exécutée sans argument, affiche les systèmes de fichiers et les attributs de montage.
mou nt
/dev/hda5 on / type ext2 (rw)
none on */proc* type proc (rw)
localhost:(pid264) on */net* type nfs
(intr,rw,port= 1023,tirneo=8,retrans= 110,indirect,map=/etc/amd.conf,dev=00000002)
/dev/fd0 on /mnt/floppy type ext2 (rw)
/dev/hdal on */win* type vfat (rw)
Création et montage d'un système de fichiers dans un fichier ordinaire.
dd if *=/dev/fd0 of=/tmp/filesys* bs=8k
mount -o loop /tmp/filesys */mnt/filesys*
mount
/trnp/filesys on /mnt/filesys type ext2 (rw,loop=/dev/loop0)
Remarque
Si un système de fichiers est monté sur un répertoire non vide, les fichiers du répertoire sont cachés pendant toute la durée du montage. Il existe une arborescence dédiée au montage des systèmes de fichiers: *mnt, /mnt/floppy, /mnt/cdrom.*

- **umount**

La commande umount, qui démonte un système de fichiers, rompt le lien qui existe entre le répertoire de montage et le système de fichiers. Les fichiers du disque sont à nouveau inaccessibles jusqu'au prochain montage.
umount */dev/hda3*
ou
umount */mnt*
L'argument de la commande umount est indifféremment le nom du disque ou le répertoire de montage (le lien est unique).

- **fsck, e2fsck**

e2fsck, est propre au système de fichiers e;t2. Elle évite d'avoir à préciser le type du système de fichiers à contrôler. La commande fsck est, en général, automatiquement exécutée sur tous les systèmes de fichiers où cela est nécessaire pendant le démarrage du système. Lors de son utilisation manuelle, les systèmes de fichiers qui sont contrôlés doivent être démontés, saufle système de fichiers principal root, pour qu'il n'y ait pas d'activité sur le disque pendant l'exécution de fsck. Le principal argument de la commande fsck est le nom du disque à contrôler. La commande fsck propose également de nombreuses options dont les plus significatives sont:
-y La commande fsck n'interroge plus l'utilisateur, la réponse est oui à toutes les questions.
-n La commande fsck n'interroge plus l'utilisateur, la réponse est non à toutes les questions.
-t type « type» désigne le type du système de fichiers (ext2, ...).
-b # # désigne le numéro du bloc qui contient la copie du super bloc à utiliser (cas où le super bloc est corrompu).

La commande stats de debugfs donne les caractéristiques des groupes de blocs. Pour chaque ensemble de blocs, la commande affiche l'emplacement des « Block bitmap ». La copie du super bloc se trouve deux blocs plus loin. Si le « block bitmap » est en 32770, le super bloc est en 32768. On peut activer directement la commande stats :
« debugfs -R stats /dev/hda6 ».

Le temps d'exécution de la commande fsck peut être relativement long, compte tenu du parcours croisé qu'elle effectue entre la table des inodes et les répertoires de l'arbre pour contrôler la cohérence du système de fichiers. Elle se déroule en plusieurs phases de contrôle et demande à l'administrateur de décider, oui ou non, des réparations qui ne peuvent être faites automatiquement.

Quand la commande rencontre un fichier perdu, qui possède un inode mais plus de lien, elle crée un lien pour ce fichier dans le répertoire « *lost+found* », situé dans la racine du système de fichiers. Le lien attribué est le numéro d'inode, ce qui évite les conflits de nom. Si le répertoire « lost+found » n'a pas été créé par la commande mkfs, il faut le faire manuellement.
mount /dev/hda3 */mnt*
cd */mnt*
mkdir lost+found
La démarche de l'administrateur qui récupère des fichiers dans le répertoire « lost+found » pour les identifier peut être la suivante:
mount /dev/hda3 */mnt*
cd */mnt/lost+found*
ls -l # pour connaître le propriétaire, le groupe et le type du fichier
file * # pour connaître le type du contenu des fichiers ordinaires
more fichier # pour un fichier de type texte (« text »)
od fichier # pour les fichiers de données (« data»)

Exemple de contrôle par e2fsck
Le nombre de liens de l'inode 13 a été forcé à trois.
e2fsck */dev/fd0*
e2fsck 1.12, 9-Jul-98 for EXT2 FS 0.5b, *95/08/09*
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts

Inode 13 ref count is 3, should be 1. Fix<y>? yes
Pass 5: Checking group summary information
/dev/fd0: ***** FILE SYSTEM WAS MODIFIED *****
/dev/fd0: 15/360 files (0.0% non-contiguous), 69/1440 blocks
Le nombre de blocs de l'inode 12 a été forcé à cent.
e2fsck -f /dev/fd0
e2fsck 1.12, 9-Jul-98 for EXT2 FS 0.5b, 95/08/09
Pass 1: Checking inodes, blocks, and sizes
Inode 12, i_blocks is 100, should be 10. Fix<y>? yes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/fd0: ***** FILE SYSTEM WAS MODIFIED *****
/dev/fd0: 15/360 files (0.0% non-contiguouS), 69/1440 blocks

- **debugfs**

La commande debugfs permet de déboguer un système de fichiers. L'administrateur peut ainsi lire ou modifier n'importe quel bloc du système de fichiers. La commande debugfs ouvre, par défaut, le système de fichiers en lecture seulement. 11 est évident que cette commande ne doit être utilisée qu'en dernier recours et par des administrateurs qui maîtrisent parfaitement la structure du système de fichiers.
Syntaxe
debugfs [-R requête] [-w] disque
L'option « -w » permet d'ouvrir le disque en lecture et en écriture.
L'option « -R requête» permet de faire exécuter une seule requête en mode non interactif, ce qui peut être utile dans un script.
Afficher l'inode d'un fichier.
ls -i /rnntlfloppy
total 18
13 -rw-r--r-- 1 root root 691 mai 8 10:58 fic.user
14 -rw-r--r-- 1root root 0 mai 810:58 fichel
15 -rw-r+r+ 1 root root 0 mai 8 1058 fiche2

umount /dev/fd0

debugfs /dev/fd0
debugfs 1.12, 9-Jul-98 for EXT2 FS 0.5b, 95/08/09
debugfs: mi <13>
debugfs: Mode [0100644]
User ID [0]
Group ID [0]
Size [691]
Creation time [926153880]
Modification time [926153880]
Access time [926153880]
Deletion time [O]
Link count [1]
Block count [2]
File flags [OxO]
File acl [0]
High 32bits of size [0]
Fragment address [0]
Fragment number [0]
Fragment size [0]

Direct Block #0 [68]
Direct Block # 1 [0]
Direct Block # Il [0]
Indirect Block [0]
Double Indirect Block [O]
Triple Indirect Block [0]
debugfs: quit
Retrouver les noms de fichiers à partir d'un numéro d'inode.
debugfs /dev/fd0
debugfs 1.12, 9-Jul-98 for EXT2 FS 0.5b, 95/08/09
debugfs: ncheck 13 14 15
debugfs: Inode Pathname
13 /fic.user
14 /fichel
15 /fiche2
debugfs: quit

- **du et dt**

Les commandes du et df permettent de gérer l'espace disque.

- **lsot**

La commande lsof permet de lister les fichiers ouverts et de connaître les applications qui y accèdent.
lsof # liste des fichiers ouverts
lsof /dev/hda5 # liste les processus qui accèdent au système de fichiers de la
partition /dev/hda5
kill $(lsof -t /dev/hda5) # tue les processus identifiés précédemment

- **tune2fs**

Elle permet de modifier les paramètres ajustables d'un système de fichiers de type ext2, selon la syntaxe suivante:

tune2fs [-l] [-c max_montage] [-i intervalle[d|m|w]][-m blocs_réservés] device
-l Affiche le super bloc.
-c max_montage Fixe le nombre maximum de montage entre deux vérifications du système de fichiers.
-i intervalle[d|m|w] Fixe le nombre de jours (d), mois (rn) ou semaines (w) entre deux vérifications du système de fichiers, six mois par défaut.
-m blocs_réservés Fixe le pourcentage de blocs du système de fichiers réservés à root.

La commande tune2fs ne doit jamais être exécutée sur un système de fichiers en cours d'utilisation, monté en lecture et en écriture.

- **dumpe2ts**

La commande dumpe2fs affiche des informations sur le super bloc et les groupes de blocs.

- **badblocks**

La commande badblocks recherche les blocs physiques endommagés d'une partition.

- **ext2resize**

La commande ext2resize permet de modifier la taille d'un système de fichiers de type ext2. Son auteur la définit comme dangereuse et à utiliser avec précaution. La commande ext2resize modifie la taille du système de fichiers, mais ne modifie pas la taille de la partition qui le contient. Sa syntaxe est suivante:
ext2resize disque nouvelle_taille

Exemple de modification de la taille d'un système de fichiers créé sur disquette.
mke2fs /dev/fd0
ext2resize /dev/fd0 500000
ext2 -resize -fs
direct hits 0 indirect hits 0 misses 0
tune2fs -1 /dev/fd0
tune2fs l.l8, II-Nov-1999 for EXT2 FS 0.5b, 95/08/09
Filesystem state: clean
Errors behavior: Continue
Filesystem OS type: Linux
Inode count: 184
Block count: 488
Block size: 1024
ext2resize /dev/fdO 1300000
ext2_resize_fs
ext2_rows_fs
ext2_block_relocate
ext2_block_relocate_grow
ext2_grow_group

direct hits 780 indirect hits 0 misses 1

tune2fs -1/dev/fd0
tune2fs 118, 11-Nov-1999 for EXT2 FS 0.5b, 95/08/09
Filesystem state: clean
Filesystem OS type:
Inode count:
Block count:
Linux
184
1269

- **e2label**

La commande e21abel change l'étiquette d'un système de fichiers ext2.

- **Un exemple complet**

Création d'un système de fichiers. Les blocs font 4096 octets, 10% du disque est réservé à l'administrateur et des fragments de 1024 octets
mke2fs -b 4096 -f 1024 -m 10 /dev/sda7
mke2fs U8, 11-Nov-1999 for EXT2 FS 0.5b, 95/08/09
Warning: fragments not supported. Ignoring -f option
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
52224 inodes, 52203 blocks
5220 blocks (10.00%) reserved for the super user
First data block=O
2 block groups
32768 blocks per group, 32768 fragments per group
26112 inodes per group
Superblock backups stored on blocks:
32768
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
Visualisation des informations de gestion.

tune2fs -1/dev/sda7 > /tmp/fic
tune2fs U8, ll-Nov-1999 for EXT2 FS 0.5b, 95/08/09
Filesystem volume name: <none>
Last mounted on: <not available>
Filesystem UUID: 2706277e-b296-4c 15-b3df-6adbeI135397
Filesystem magic number: OxEF53
Filesystem revision #: 1 (dynamic)
Filesystem features: filetype sparse_super
Filesystem state: clean
Errors behavior: Continue
Filesystem OS type: Linux
Inode count: 52224
Block count: 52203
Reserved block count: 5220
Free blocks: 50558
Free inodes: 52213

First block: 0

Block size: 4096
Fragment size: 4096
Blocks per group: *32768*
Fragments per group: 32768
Inodes per group: 26112
Inode blocks per group: 816
Last mount time: Thu Jan 1 01:00:00 1970
Last write time: Mon Oct 30 16:32: 172000
Mount count: 0
Maximum mount count: 20
Lastchecked: Mon Oct 3016:32:142000
Check interval: 15552000 (6 months)
Next check after: SatApr 2817:32:142001
Reserved blocks uid: 0 (user root)
Reserved blocks gid: 0 (group root)
First inode: ll
Inode size: 128

Montage du système de fichiers sur /mnt/appli

mount /dev/sda7 /mnt/appli
#ls-iR
.:
13 fracinel 12 fracine2 11 lost+found 26113 r1
./lost+found:
./r 1:
26114 ls

Vérification du disque après un « crash»

e2fsck /dev/sda7
e2fsck 118, ll-Nov-1999 for EXT2 FS 0.5b, 95/08/09
/dev/sda7 was not cleanly unmounted, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Unconnected directory inode 26113 (/???)
Connect to /lost+found? yes
Pass 4: Checking reference counts
Unattached inode 13

Connect to /lost+found? yes
Inode 13 ref count is 2, should be 1. Fix? yes
Inode 26113 ref count is 5, should be 2. Fix? yes
Pass 5: Checkiug group summary information
/dev/sda7: ***** FILE SYSTEM WAS MODIFIED *****
/dev/sda7: 15/52224 files (00% non-contiguous), 1659/52203 block

mount /dev/sda7 /mnt/appli
12 fracine2 11 lost+found 26113 r1
ls -iR
./
12 fracine2 11 lost+found
./lost+found:
13 #13 26113 #26113

./lost+found/#26113:

LE répertoire "lost+found" contient 2 fichiers trouvés. Les liens sont les numéros d'inode des fichiers.

On essaie de reconstituer les liens d'origine.
cd lost+found/
ls -l
total 8
-rw-r--r-- 1 root root 6 oct 30 16:49 #13
drwxr-xr-x 2 root root 4096 oct 30 16:49 #26113

file \#13
#13: ASCII text
ls -R \#26113
#26113:
ls
On reconstitue si possible.
mv \#13 ../fracinel
mv \#26113 ../r1

Modification du nombre de blocs réservés à l'administrateur
tune2fs -m 15 /dev/sda7
tune2fs 1.18, ll-Nov-1999 for EXT2 FS 0.5b, 95/08/09
Setting reserved blocks percentage to 15 (7830 blocks)

Vérification du disque après un «crash» en utilisant une copie du super bloc
Destruction du super bloc.
dd if=/dev/zero of=/dev/sda7 bs=4k count=1
1+0 enregistrements lus.
1+0 enregistrements écrits.
mount /dev/sda7 /mnt/appli
mount: you must specify the filesystem type
On indique l'adresse de la copie.
e2fsck -b 32768 /dev/sda7
e2fsck 1.18, ll-Nov-1999 for EXT2 FS 0.5b, 95/08/09
/dev/sda7 was not cleanly unmounted, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure

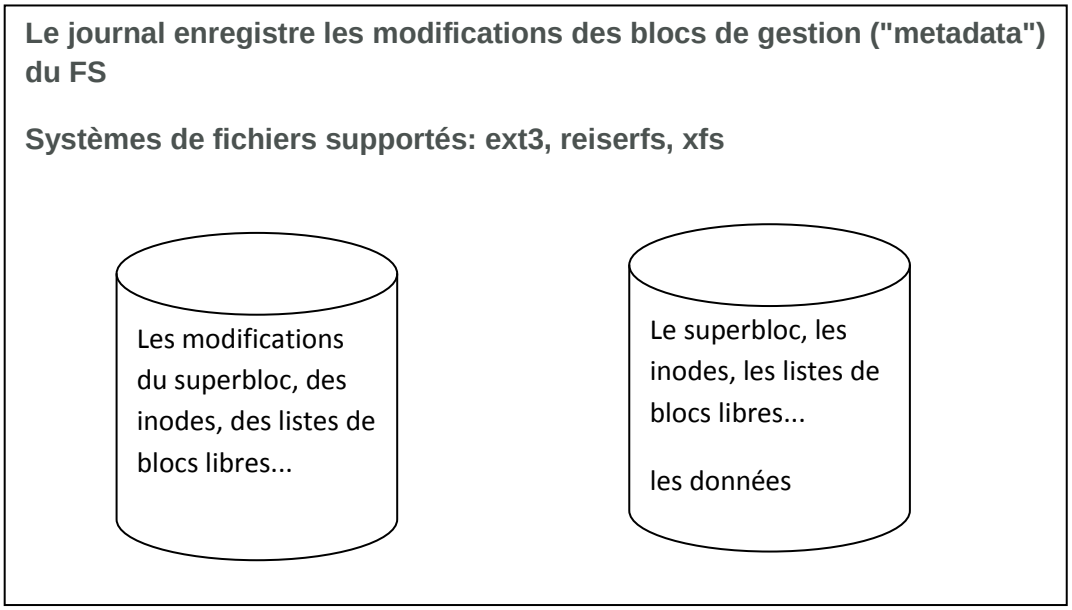
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sda7: ***** FILE SYSTEMWAS MODIFIED *****
/dev/sda7: 15/52224 files (0.0% non-contiguous), 1659/52203 blocks
On remonte le système de fichiers.
mount /dev/sda7 /mnt

cd / mnt /appli

ls -iR

.:
13 fracine1 12 fracine2 11 lost+found 26113 r1
./lost+found:
./rl:
261141s

1.5- Les systèmes de fichiers journalisés



- **Introduction à la journalisation**

A une époque où la taille des systèmes de fichiers devient de plus en plus importante, la journalisation vise principalement à diminuer la durée du contrôle et de la réparation d'un système de fichiers. Elle assure aussi une meilleure intégrité des fichiers en cas de crash.
Dans le monde UNIX, on rencontre principalement le système de fichiers « vxfs » (« *Veritas File System* »), le système «JFS» *(1< Journaled FileSystem* ») de la société IBM et le système « XFS » de la société Silicon Graphics.
Dans le monde Linux, on rencontre le système de fichiers « ext3 », compatible avec son prédécesseur « ext2 », le système « Reiser FS » et «XFS» déjà nommé.
L'idée maîtresse des systèmes journalisés est de conserver l'ensemble des données de gestion du système de fichiers (« *metadata* ») dans un journal enregistré sur disque, plutôt que dans les mémoires cache du système. En cas de crash, la commande fsck contrôle la cohérence du système de fichiers, rapidement, en quelques secondes, à partir des informations contenues dans le journal, sans avoir à analyser la totalité des inodes et de l'arborescence des répertoires.

- **Le système de fichiers ext3**

Le système de fichiers ext3 est un système de fichiers journalisé qui assure une compatibilité complète, ascendante et descendante avec son ancêtre ext2. Un système de fichiers ext2 peut être doté d'un journal et donc devenir ext3 et, inversement, un système de fichiers ext3 peut être, sans aucune difficulté, monté en tant que systèmes de fichiers ext2. Le système de fichiers ext3 est supporté par le noyau 2.4. Pour les versions plus anciennes de noyau, il faut charger
un patch.
Il faut d'ailleurs noter qu'il n'existe pas de commandes propres à ext3 et que l'on continue à utiliser les commandes mke2fs, tune2fs, e2fsck. Elles ont évidemment été enrichies de quelques options liées à la journalisation, de même que la commande **mount**.
La journalisation est assurée par une APl totalement indépendante de ext3. La journalisation pourrait, pour cette raison, être mise en œuvre par un autre type de système de fichiers que ext3. On peut dire que ext3, c'est ext2 plus un journal.

ext3 ne connaît la journalisation qu'à travers le concept de transaction. Une transaction désigne les mises à jour des blocs de gestion opérées entre le début et la fin de la transaction. La couche qui prend en prend en charge la journalisation garantit qu'après un crash, la totalité d'une transaction a été prise en compte ou totalement ignorée. Les dernières modifications apportées à un fichier n'ont peut être pas été enregistrées, mais le inode du fichier est cohérent par rapport aux données.

La commande mke2fs
La commande mke2fs ajoute quelques options relatives à la journalisation :

- L'option «-j »génère la création d'un système de fichiers ext3. Le journal est interne, il est stocké dans le système de fichiers.
- Les autres options sont de la forme option=valeur, .. Les options sont :
    - size=taille journal (L'unité est le méga octets. Le journal doit avoir une taille d'au moins 1000 fois la taille du bloc du système de fichiers.)
    - device=journal_ device (Cette option indique le nom de la partition qui contient le journal *(cf option -O).*
- -F révision : Les valeurs possibles de révision sont 0 ou 1. Cette option peut être considérée comme obsolète. Nous la mentionnons car la commande mount permet de convertir un système de fichiers de révision 0 en révision 1.
- -O caractéristiques : Les caractéristiques sont de la forme caractéristique, ... Les caractéristiques sont :
    - hasjournal qui est équivalente à -j
    - journal_dev qui est utilisé pour créer un journal externe au système de fichiers *(cf exemples).* plusieurs systèmes de fichiers peuvent se partager le même journal.
    - sparse _super qui crée un système de fichiers avec un nombre réduit de copies du super bloc. Cette option peut être utile pour un très gros système de fichiers.
    - filetype pour mémoriser le type des fichiers dans les répertoires.

- ***La commande tune2fs***

La commande tune2fs offre, comme principale nouveauté, de doter un système de fichiers de type ext2 d'un journal.

- ***La commande e2fsck***

La commande e2fsck permet d'indiquer la localisation du système de fichiers quand il est externe.

-j journal : L'argument de l'option « -j » indique la localisation du journal.

- ***La commande mount***

La commande mount, pour ext3 comme pour tout autre système de fichiers, permet de préciser des options spécifiques. Les options sont de la forme -o option, ... Voici les principales options utilisables à la suite de l'option -o:

Description

- -o journal=update

- -o data=mode

Cette option convertit un système de fichiers de révision 0 en révision 1.
L'argument « mode» indique le type de journalisation. Les choix possibles sont les suivants:

- data=journal

Quand le système de fichiers est monté avec ce mode, les blocs de données sont aussi écrits dans le journal. Cela nécessite un journal d'une taille conséquente. Les blocs de

données sont écrits deux fois sur disque. Les performances s'en trouvent donc notablement réduites.

- data=ordered

C'est le mode par défaut des systèmes de fichiers de révision 1. Dans ce mode, seuls les blocs de gestion sontjournalisés. Les blocs de données sont écrits sur disque avant que la transaction soit terminée. Cela garantit le type d'intégrité que nous présentons dans l'introduction.

- data=writeback

Comme dans le mode précédent, seuls les blocs de gestion sont journalisés, mais il est possible que des blocs de données d'anciens fichiers soient alloués en double. On retrouve les possibilités d'erreur du système de fichiers ext2.

Exemples

Création d'un système de fichiers ext2 sur le disque /dev/hda9

mke2fs */dev/hda9*
mke2fs 1.23, 15-Aug-2001 for EXT2 FS 0.5b, *95108/09.*
Superblock backups stored on blocks:
8193,24577,40961,57345,73729
Writing inode tables: *0/13 1/132/133113* 4/13 5/13 *6/13 7/13 8/139/13 10/13* 11/13
12/13 done
Writing superblocks and filesystem accounting information: done

mount *Idev/hda9* Imnt/appli9
mount
/dev/hda9 on Imnt/appli9 type ext2 (rw)
#df

Filesystem	1k-blocks	Used	Available	Use%	Mounted on
/dev/hda9	102454	13	97151	1%	/mnt/appli9

Création d'un système de fichiers ext3 sur le disque /dev/hda9
mke2fs -j /dev/hda9
mke2fs 1.23, 15-Aug-2001 for EXT2 FS 0.5b, *95108/09*
Creating journal (4096 blocks): done
mount
/dev/hda9 on Imnt/appli9 type ext3 (rw)
#df
Filesystem Ik-blocks Used Available Use% Mounted on
/dev/hda9 102454 4127 93037 5% /mnt/appli9
Le nombre de blocs utilisés est de 4127 blocs pour le type ext3 et de 13 pour le système de type ext2. Notons que le journal est de 4 Mo.

Conversion d'un système de fichiers ext2 en ext3
mke2fs Idev/hdal0
mount
/dev/hda10 on /mnt/appli10 type ext2(rw)

tune2fs -j /dev/hda10

tune2fs 1.23, 15-Aug-2001 for EXT2 FS 0.5b, 95/08/09
Creating journal inode: done
mount
/dev/hdalO on /mntlapplilO type ext3 (rw)

Montage d'un système defichiers ext3 en tant que ext2

mount -t ext2 /dev/hda9 /mnt/appli9
/dev/hda9 on /mnt/appli9 type ext2 (rw)

Création d'un journal externe, ici /dev/hda11
Création du journal
mke2fs -0 journaldev /dev/hda11
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
0 inodes, 5662 blocks
0 blocks (0.00%) reserved for.the super user
First data block=0
0 block group
32768 blocks per group, 32768 fragments per group
0 inodes per group
Superblock backups stored on blocks:
Zeroing journal device: done
Création du système de fichiers
mke2fs -J device=/dev/hda11 /dev/hda9
mke2fs 1.23, 15-Aug-2001 for EXT2 FS 0.5b, 95/08/09
Filesystem label=
Adding journal to deviee /dev/hda11: done
Writing superblocks and filesystem accounting information: done
mount -t ext3 /dev/hda9 /mnt/appli9
Remarques:
La création d'un journal externe nécessite un paquetage e2fsprogs trés récent, de version 1.25 ou plus de préférence. '
Quand le journal est externe, il est nécessaire d'indiquer explicitement le type ext3.

Quelques montages du système de fichiers ext3

- Conversion d'un système de fichiers de la revision 0 à la revision 1.

mount -t ext3 -o journal=update /dev/hda9 /mnt/appli9

- Journalisation des blocs de gestion et des données.

mount -t ext3 -o data=journal /dev/hda9 /mnt/appli9

- Journalisation des blocs de gestion et garantie d'intégrité des fichiers

mount -t ext3 -o data=ordered /dev/hda9 /mnt/appli9

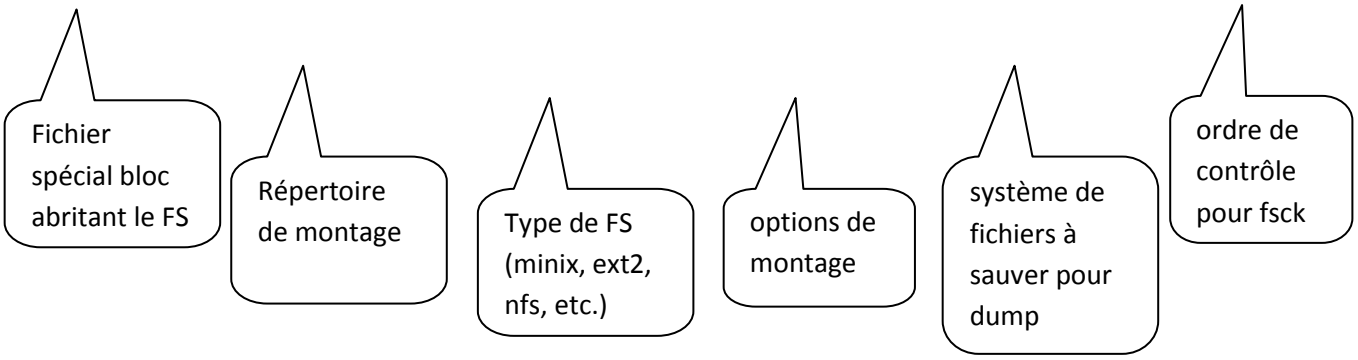
- Journalisation des blocs de gestion et comportement ext2 pour les données des fichiers

mount -t ext3 -0 data=writeback /dev/hda9 /mnt/appli9

Références pour ext3
ftp://ftp.linuxsvmposium.orglols2000/2000-07-20 15-05-22 A 64.mp3
La documentation RedHat
http://wwv.·.redhat.com/support/wpapers/redhatlext3/tuning.html
http://www.redhat.com/support/wpapers/redhatlext3/tuning.html
Le HOWTO http://www.symonds.netl-rajesh/howto/ext3/ext3-5.html
Le patch du noyau pour les noyaux antérieurs au noyau 2.4.16

1.6- Automatiser le montage des systèmes de fichiers

```
# more /etc/fstab
/dev/hda5       /           ext2        defaults        1       1
/dev/hda6       swap        swap        defaults        0       0
/dev/cdrom      /mnt/cdrom  iso9660     noauto,ro       0       0
```



- Prédéfinition des paramètres de montage

Les disques qui contiennent des systèmes de fichiers doivent, en général, être montés à chaque démarrage du système et démontés à chaque arrêt, le montage étant toujours réalisé sur le même répertoire.
Pour automatiser ces opérations, l'administrateur d'un système Linux doit modifier le fichier */etc/fstab* qui contient la liste des disques à monter automatiquement. La prise en compte sera réalisée dès le prochain démarrage du système. C'est la seule
opération à réaliser!
C'est la commande mount -a, exécutée par les scripts de démarrage, qui prend en compte le contenu du fichier */etc/fstab.* Ce fichier contient une ligne par système de fichiers. Les champs sont respectivement:
• Le nom du disque en mode bloc (le fichier spécial de type b).
• Le chemin d'accès absolu au répertoire de montage.
• Le type du système de fichiers.
• Les attributs de montage. L'attribut le plus fréquent est « rw» [« *read/write* ») qui donne un accès complet au système de fichiers. L'accès aux fichiers dépend, quant à lui, des droits dont on dispose sur eux. L'attribut « ro» limite l'accès à la lecture seulement. L'attribut « noauto » indique que le système de fichiers ne doit pas être pris en compte par l'exécution de la commande mount -a. L'existence d'une ligne de montage d'un système de fichiers avec l'attribut « noauto » permet de simplifier l'exécution de la commande mount à laquelle il suffit de fournir en argument le nom du disque ou le chemin de montage.

```
# more /etc/fstab
/dev/cdrom /mnt/cdrom is09660 noauto,ro 0 0
# mount /dev/cdrom
```

ou

```
# mount /mnt/cdrom
```

Il est possiole de mentionner d'autres attributs:
«suid », «nosuid» qui indiquent la prise en compte [« suid ») ou non (« nosuid ») du bit « s » des fichiers exécutables. Par défaut l'attribut est « suid ».
« usrquota » et « grpquota » qui signifient que la gestion des quotas est active pour ce système de fichiers.
Remarques:
Les attributs de montage peuvent être précisés dans la ligne de commande mount.

Le choix des attributs de montage et l'automatisation peuvent être facilement réalisés par l'outil intégré d'administration, ce qui évite de mémoriser la forme du fichier d'automatisation.
• L'indication pour la commande de sauvegarde dump si le système de fichiers doit être sauvegardé. Si le champ est à 0, le système de fichiers ne sera pas sauvegardé.
• L'indication pour la commande fsck de l'ordre dans lequel elle doit contrôler les systèmes de fichiers.

- **Montage à la volée d'un système de fichiers avec « autofs** »

Le démon « automount » réalise automatiquement le montage d'un système de fichiers si un processus manipule des fichiers situés en dessous du répertoire de montage et le démonte quand l'arborescence n'est plus utilisée. Pour le mettre en œuvre, il faut installer le paquetage « autofs »,
Une fois le paquetage installé, on dispose du script */etc/rc.d/init.d/autofs* pour démarrer ou arrêter le démon automount. Le démarrage du service est normalement automatiquement réalisé au chargement du système. L'automontage est particulièrement intéressant pour les disques amovibles et les systèmes de fichiers distants.
La configuration du service est réalisée au moyen de deux types de fichiers:
1. Le fichier */etc/auto.master* qui indique les répertoires racines en dessous desquels sont réalisés les montages et les noms des fichiers qui contiennent le détail des montages. Si le fichier a un nom différent, il faut l'indiquer au démarrage du
démon automount.
Chaque ligne du fichier *letc/auto.master,* qui n'est pas un commentaire, a la structure suivante:
Point_de_montage Fichier_détail [Options]
Parmi les options, notons principalement -timetout n, qui indique au bout de combien de temps d'inactivité le système de fichiers est démonté, 5 minutes par défaut.

```
# more /etc/auto.master
/mnt/auto /etc/auto.mnt --timeout 60
```

2. Les fichiers qui décrivent le détail des montages. Chaque ligne d'un fichier de détail a la structure suivante:
Chemin_complémentaire [-Options _de_montage, ...] Système _de_fichiers
La localisation du système de fichiers est de la forme
[hôte]:système_de_fichiers

```
# more /etc/auto.mnt
cdrom -fstype=iso9660,ro,nosuid :/dev/cdrom
```

Remarque
L'installation du paquetage crée un fichier */etc/auto.master* et un fichier */etc/auto.misc* comme modèle de détail.
Exemple

```
# ls /mnt/auto # le répertoire /mnt/auto est vide
# mount # Les systèmes de fichiers montés
/dev/sda5 on / type ext2 (rw)
none on /proc type proc (rw)
/dev/sda1 on /boot type ext2 (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
# /etc/rc.d/init.d/autofs start                    # Démarrage du service
Starting automounter: [ OK ]
# ps -e | grep automount
933 pts/0 00:00:00 automount
# cd /mnt/auto/cdrom
# ls
COPYlNG RELEASE-NOTES RedHat autorun doc images rr_moved
README RPM-GPG-KEY TRANS.TBL boot.cat dosutils mise
```

```
# cd /mnt/auto/appli
# ls
fiel fic2
# mount
/dev/sda5 on / type ext2 (rw)
none on /proc type proc (rw)
/dev/sdal on /boot type ext2 (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
automount(pid933) on /mnt/auto type autofs
(rw,fd=5,pgrp=933,minproto=2,maxproto=3)
/dev/hdc on /mnt/auto/cdrom type is09660 (ro,nosuid,nodev)
aurore6:/mnt/appli on /mnt/auto/appli type nfs (rw,nosuid,addr=192.0.0.6)
# mount #Les systèmes de fichiers sont démontés
/dev/sda5 on / type ext2 (rw)
none on /proc type proc (rw)
/dev/sda1 on /boot type ext2 (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
automount(pid933) on /mnt/auto type autofs
(rw,fd=5,pgrp=933,minproto=2,maxproto=3)
```

- **Références pour autofs:**

Le mini-HOW-TO

Automount, Automount mini-HOWTO

La documentation.
/usr/doc/autofs-3.1.4

Le manuel.
mount(8), autofs(5) , auto.master(5)

1.7- Les quotas:

- **Introduction:**

Les principales commandes:
• edquota : Permet de fixer les quotas (Utilisateur par utilisateur ou groupe par groupe, disque par disque, pour l'espace disque, pour les inodes)
Un quota possède deux limites:
« *hard limit»* limite infranchissable
« *soft limit»* limite provoquant un « warning »
• quotaon : Active la vérification des quotas
• quotaoff : Désactive la vérification des quotas
• quota, repquota : Liste les quotas
• quotacheck : Vérifie, a priori, si les quotas sont respectés

La mise en œuvre des quotas va permettre à l'administrateur de limiter le nombre de fichiers ou le nombre de blocs d'un utilisateur ou d'un groupe, sur un disque. Cette opération est assez peu utilisée dans le monde UNIX. Elle sert principalement sur les machines de développement où les programmeurs ont rarement le sens de la mesure quant à l'utilisation des ressources disque. A l'administrateur de faire accepter les quotas avec psychologie.
Les quotas offrent un plus grand intérêt dans Linux qui est souvent utilisé comme serveur de fichiers (Samba) ou comme serveur de messagerie (Sendmail, Postfix ..).
Pour les fichiers aussi bien que les blocs, il existe deux limites:
• La limite « hard » qui est infranchissable. Un utilisateur ou un groupe qui atteint sa limite «hard » de fichiers ne pourra pas en créer un de plus. L'éditeur de texte vi refusera ainsi d'exécuter la commande de sauvegarde :w.
• La limite « soft» peut être franchie pendant un certain nombre de jours consécutifs, sept par défaut. Si, au terme de ce laps de temps, l'utilisateur n'est pas redescendu en dessous de sa limite « soft », le point atteint devient à son tour infranchissable, jusqu'au retour à la normale.

- **Mise en oeuvre**

La démarche de l'administrateur pour installer des quotas sur un disque est la suivante:
1. Monter le système de fichiers avec l'une des options de montage « usrquota » ou « grpquota » ou les deux. Les options « usrquota » et « grpquota » indiquent si les quotas s'appliquent aux utilisateurs ou aux groupes.
Comme dans tous les cas, l'administrateur peut préciser les options de montage sur la ligne de commande ou bien modifier le fichier */etc/fstab.*

```
# vi /etc/fstab
/dev/hda9       /mnt/appli9     default, usrquota       1       2
# mount /dev/hda9

# mount -o usrquota /dev/hda9 /mnt/appli9

# mount

...
/dev/hda9 on /mnt/appli9 type ext2 (rw,usrquota)
```

2. Créer les fichiers qui mémorisent les quotas des utilisateurs et des groupes. Ces fichiers doivent être créés dans le répertoire de montage du système de fichiers. Il existe deux versions de quotas dans le monde Linux :

- La plus ancienne est la version 1. Les fichiers y ont pour nom *quota.user* pour les utilisateurs et *quota.group* pour les groupes. On les crée en exécutant la commande touch.

```
# cd /mnt/appli9
```

touch quota.user

- La plus récente est la version 2. Les fichiers y ont pour nom *aquota.user* et *aquota.group.* Ils doivent être créés par la commande quotacheck. Ces fichiers, à la différence des précédents, ne sont pas initialement vides.

```
# quotacheck /mnt/appli9
# Is -l /mnt/appli9
total 18
-rw------- 1 root        root     6144jan 17 14:31 aquota.user
drwxr-xr-x 2 root        root     12288jan 17 14:0210st+found
```

La nouvelle version des quotas supporte des UlDs et des GIDs sur 32 bits et fonctionne sur des systèmes de fichiers autres que ext2 ou ext3, tel reiserFS. La commande convertquota crée les fichiers *aquota. user* et *aquota.group* à partir de leurs prédécesseurs *quota. user* et *quota.group.Elle* devra être exécutée lors d'une mise à jour de Linux qui nécessite le passage de la version 1 à la version 2.

3. Editer les quotas des utilisateurs.

```
# edquota -u ali
Disk quotas for user ali (uid 501):
Filesystem blocks soft hard inodes soft hard
/dev/hda9 0 0 0 1 5 10
```

Editer les quotas des groupes.

```
# edquota -g etude
Disk quotas for group etude (gid 2000):
Filesystem blocks soft hard inodes soft hard
/dev/hda9 0 0 0 2 0 0
```

La commande edquota fait appel à l'éditeur vi pour présenter des lignes déjà construites qu'il suffit de modifier. Il y a autant de lignes que de systèmes de fichiers avec des quotas. Les limites « soft» et « hard » pour le nouveau disque sont à zéro, ce qui signifie « pas de limite ». L'administrateur les modifie à sa guise.
L'administrateur peut aussi, via la commande edquota -t, modifier les valeurs par défaut des quotas et, en particulier, le nombre de jours où lalimite soft peut être encore dépassée.

```
# edquota -t
Grace periode before enforcing soft limit for users:
Time units may be: days, hours, minutes, or seconds
Filesystem Block grace period Inode grace period
/dev/hda9 7days 7days
```

Si un système de fichiers avec des quotas est partagé par NFS, le démon rquotad est démarré par l'exécution du script /etc/rc.d/init.d/nfs start. Les quotas des utilisateurs qui se connectent sur un poste client qui a réalisé le montage NFS sont définis par la commande edquota -n ou edquota -r, exécutée sur le client.

La commande setquota est une alternative à la commande edquota. On fournit les arguments sur la ligne de commande. Cela permet de pallier au défaut de certaines implémentations de edquota qui ne précisent pas le modèle des données à saisie Cette commande se révèle aussi très pratique dans un script La commande qui suit fixe ainsi les quotas de ali:
La limite « soft» pour les blocs est de 10000 et la limite « hard »de 20000.
La limite « soft» pour les inodes est de 100 la limite « hard » de 110

```
# setquota ali  10000 20000 100 110 /dev/hda9
```

4. Activer les quotas. La commande quotaon réalise cette opération. L'administrateur peut activer les quotas pour un disque ou activer les quotas de tous les disques qui ont les options des quotas dans le fichier */etc/fstab .*
quotaon /dev/hda9 # Active les quotas pour /dev/hda9

- Suivi des quotas

L'utilisateur qui dépasse une des deux limites voit un message d'avertissement ou d'erreur s'afficher sur son écran.
L'utilisateur ali appartient au groupe étude. Il a dépassé sa limite « soft» pour les fichiers et, ce faisant, a aussi entraîné un dépassement pour le groupe.

$ touch a z e r t y
ide0(3,9): warning, group file quota exceeded.
ide0(3,9): warning, user file quota exceeded.
Le dépassement de la limite « hard » génère un message nettement moins explicite.

$ touch f
touch: creating 'f ':débordement du quota d'espace disque

Pour assurer le suivi des quotas, l'administrateur dispose des commandes quota, warnquota, repquota et quotacheck

La commande quota affiche les valeurs courantes et les limites de l'utilisateur qui l'exécute, pas forcément l'utilisateur root.

quotaon -a # Active les quotas pour les systèmes de fichiers montés avec
les options « usrquota » ou « grpqucta » dans */etc/fstab.*

Remarque:
Rappelons que l'administrateur doit modifier le fichier */etclfstab* et s'assurer que la commande quoataon -a figure bien dans les scripts de démarrage pour que les quotas soient systématiquement activés à chaque démarrage du système.

$ quota
Disk quotas for user brahim (uid 502):

Filesystem	blocks	quota	limit	grace	files	quota	limit	grace
/dev/hda9	0	0	0		3	5	10	

L'administrateur peut visualiser les valeurs d'un autre utiisateur

quota -u ali
Disk quotas for user ali (uid 501):

Filesystem	blocks	quota	limit	grace	files	quota	limit	grace
/dev/hda9	10001*		10000	15000		16	0	0

Les valeurs des colonnes blocks et files représentent les valeurs actuelles et sont fournies à titre indicatif Elles ne peuvent pas être modifiées.
La commande warnquota permet d'envoyer un courrier aux utilisateurs qui ont dépassé leurs quotas.
La commande repquota affiche une synthèse de l'utilisation d'un disque et des quotas d'un système de fichiers.

repquota /dev/hda9
*** Report for user quotas on device /dev/hda9
Block grace time: 7days; lnode grace time: 7days

	Block limits				File limits			
User	used	soft	hard	grace	used	soft	hard	grace
ali	-+	0	0	0	10	5	10	6days
brahim		--	0	0	0	3	5	10
said	--	0	0	0	3	0	0	

repquota -a # Tous les systèmes de fichiers avec quotas

La commande quotacheck vérifie la cohérence des tables des quotas, contenues dans les fichiers *aquota.usr* et *aquota.group.* Pour cela, elle reconstruit de nouvelles tables, en examinant les systèmes de fichiers, et les compare à celles des fichiers quotas. La mise à jour des tables est effectuée, si nécessaire.

```
# quotacheck /dev/hda9
quotacheck: Quota for users is enabled on mountpoint /mnt/appli9 so
quotacheck might damage the file.
Please turn quotas off or use -f to force checking.
# quotaoff /dev/hda9
# quotacheck /dev/hda9
```

quotacheck -a