

Gestion et configuration des systèmes de fichiers

1. Gestion des systèmes de fichiers

a. Les systèmes de fichiers courants

Un système d'exploitation est dans la plupart des cas installé sur un disque dur ou périphérique de stockage assimilé. Si on regarde de très près un disque dur neuf, on constate que son espace de stockage est constitué d'une suite d'octets sans aucune forme d'organisation. Pour exploiter convenablement tout ou partie de cet espace de stockage, il convient de le segmenter dans un premier temps, c'est le partitionnement, puis de créer sur les partitions à exploiter un système de fichiers.

Le système de fichiers sert à organiser un espace de stockage brut, comme une partition de disque pour y stocker des données. Si le terme courant est souvent le formatage de l'espace de stockage, on parle souvent en environnement Linux de création de filesystem.

Le terme filesystem (en anglais) fait l'objet d'une convention d'utilisation fréquente qui sera reprise dans cet ouvrage. On parle de « filesystem » lorsqu'il s'agit d'un système de fichiers attaché à un périphérique de stockage unique, et on parlera de « système de fichiers » pour désigner l'espace de stockage organisé, qu'il soit composé d'un ou de plusieurs filesystems.

Il existe plusieurs types de filesystems, dont les plus courants en environnement Linux sont ext, reiserfs et xfs.

ext

ext est le filesystem historique des systèmes Linux. Il existe actuellement trois versions de filesystem ext en production. ext2 est la version historique, ext3 est une évolution de ext2 qui lui ajoute un journal de transactions, et ext4 est une dernière évolution qui équipe les systèmes les plus récents et qui vise à pallier les limites de l'ext3 (taille maximum de fichiers portée de 2 téraoctets pour ext3 à 16 téraoctets pour ext4 par exemple).

Le journal de transactions présent en ext3 et ext4 permet d'accélérer notablement les vérifications sur les systèmes de fichiers et la récupération en cas de crash.

reiserfs

Reiserfs est un filesystem journalisé qui offre pour certaines opérations des performances un peu meilleures que ext3. Reiserfs se posait d'ailleurs en concurrent de ext à sa création. Ancien système de fichier par défaut des distributions Suse, reiserfs est aujourd'hui en voie de raréfaction. On lui reproche selon les conditions d'emplois une certaine fragilité ou un manque de performances globales.

xfs

xfs est le filesystem historique des serveurs unix IRIX. Il a été placé sous licence GPL en 2000. De bonnes performances ainsi que le support des très gros espaces de stockages (8 exaoctets de taille maximum pour le filesystem contre 16 et 32 téraoctets pour reiserfs et ext3) en font un filesystem intéressant.

b. Les systèmes de fichiers virtuels ou pseudo-filesystems

Un filesystem courant a pour objet de permettre l'exploitation d'un espace de stockage physique par un utilisateur ou des applications. Il existe toutefois sur les systèmes Linux des filesystems virtuels qui n'ont de réalité qu'en mémoire sans occupation d'espace sur le disque. Ils sont simplement visibles à l'utilisateur sans exploiter un quelconque espace disque.

proc

Le filesystem virtuel proc, généralement monté sous le répertoire /proc permet de visualiser des éléments systèmes liés à la gestion des processus par le noyau. proc montre aussi un certain nombre d'informations systèmes liées au matériel.

Visualisation des informations du processeur

On observe ici les informations techniques liées au microprocesseur employé. Notez par exemple la vitesse réelle de l'horloge au moment de l'exécution de la commande, qui atteste de la bonne gestion de l'énergie sur un système peu sollicité.

```
toto@serveur:~$ cat /proc/cpuinfo
```

```
processor : 0
vendor_id : AuthenticAMD
cpu family      : 15
model          : 75
model name     : AMD Athlon(tm) 64 X2 Dual Core Processor 4200+
stepping : 2
cpu MHz        : 1000.000
cache size     : 512 KB
physical id    : 0
siblings : 2
core id       : 0
cpu cores : 2
apicid        : 0
initial apicid : 0
fpu : yes
fpu_exception : yes
cpuid level   : 1
wp : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx
mmxext fxsr_opt rdtscp lm 3dnowext 3dnow rep_good extd_apicid pni
cx16 lahf_lm cmp_legacy svm extapic cr8_legacy
bogomips : 1999.89
TLB size : 1024 4K pages
clflush size      : 64
cache_alignment   : 64
address sizes     : 40 bits physical, 48 bits virtual
power management  : ts fid vid ttp tm stc
```

sys

Le filesystem virtuel **sys**, généralement monté sous le répertoire /sys permet de visualiser des éléments systèmes liés aux périphériques.

Visualisation de capacités hotplug d'un disque dur

De nombreux pseudo-fichiers de /proc et /sys ont un contenu limité à un seul caractère. Ici 0 pour indiquer que le disque sda n'est pas enfichable à chaud.

```
toto@serveur:~$ cat /sys/block/sda/removable
0
```

c. Création des filesystems

Les filesystems sont créés par l'administrateur sur des espaces de stockages bruts, historiquement des partitions de disque. Ils sont ensuite vérifiés, ponctuellement par l'administrateur ou à intervalles réguliers automatiquement par le système. La création de filesystems se fait historiquement avec la commande **mkfs**.

Syntaxe de la commande mkfs

```
mkfs -t type device
```

mkfs : options et paramètres	
-t type	Précision du type de filesystem à créer : ext2, ext3, reiserfs, xfs...
device	Fichier spécial en mode bloc qui désigne le périphérique sur lequel créer le filesystem.

d. Vérification des filesystems

La vérification d'un filesystem consiste essentiellement en la vérification de cohérence entre la table des inodes du filesystem et les blocs de données correspondants. C'est-à-dire que pour chaque inode, on vérifiera que les blocs de données référencés par cet inode sont bien présents, en nombre et quantité annoncés. Pour les filesystems

journalisés, une option -f oblige une vérification complète d'un filesystem semblant propre, comme par exemple un filesystem qui n'aurait pas subi d'opération d'écriture depuis sa dernière vérification réussie.

La vérification de filesystems se fait avec la commande **fsck**.


Syntaxe de la commande fsck

```
fsck -t type device
```

fsck : options et paramètres	
-t type	Type du filesystem à vérifier.
device	Fichier spécial en mode bloc qui désigne le périphérique sur lequel se trouve le filesystem à vérifier.

e. Commandes spécialisées des filesystems ext

Les syntaxes indiquées ci-dessus pour les commandes **mkfs** et **fsck** sont universelles et doivent fonctionner. Toutefois, il faut savoir que ces commandes appellent en réalité des sous-programmes (mkfs.ext2 par exemple pour mkfs -t ext2), et qu'il existe par ailleurs des commandes spécialisées qui produiront le même résultat (**mke2fs** est un autre équivalent de **mkfs** -t ext2).

 Contrairement à la commande fsck, e2fsck fonctionne par défaut en mode interactif. Pour un fonctionnement en mode non interactif, elle doit être utilisée avec l'option -p. Elle vérifie alors automatiquement le filesystem sans nécessiter d'intervention de l'utilisateur.

f. Création de filesystem ext2 ou ext3

La commande mke2fs permet de créer directement des filesystems ext. Le format ext2 est utilisé par défaut, mais l'option -j (journal) permet de créer des structures de filesystem ext3.

Création d'un filesystem ext2

```
mke2fs device
```

Création d'un filesystem ext3

```
mke2fs -j device
```

Où *device* représente le fichier spécial en mode bloc qui désigne le périphérique sur lequel se trouve le filesystem à créer.

g. Affichage et modification des filesystems ext

La commande tune2fs permet d'afficher les paramètres d'un filesystem ext et éventuellement d'en modifier certains.

Affichage des paramètres d'un filesystem avec tune2fs

```
tune2fs -l device
```

Où *device* représente le fichier spécial en mode bloc qui désigne le périphérique sur lequel se trouve le filesystem à vérifier.

La différence entre le format ext2 et ext3 est la présence ou non d'un journal des transactions. La commande tune2fs permet d'ajouter un journal à un filesystem ext2, et donc de le convertir en ext3.

Affichage des paramètres d'un filesystem en ext3

Notez que la mention *has_journal* figure dans la section *Filesystem features*, ce qui indique un filesystem de type ext3.

```

alpha:/dev# tune2fs -l /dev/hda1
tune2fs 1.41.3 (12-Oct-2008)
Filesystem volume name:      <none>
Last mounted on:             <not available>
Filesystem UUID:             ff700a3b-b430-49a7-ae73-bd23f363a3fc
Filesystem magic number:     0xEF53
Filesystem revision #:       1 (dynamic)
Filesystem features:         has_journal ext_attr resize_inode
dir_index filetype needs_recovery sparse_super large_file
Filesystem flags:            signed_directory_hash
Default mount options:       (none)
Filesystem state:            clean
Errors behavior:             Continue
Filesystem OS type:          Linux
Inode count:                 501952
Block count:                 2004100
Reserved block count:        100205
Free blocks:                 1656481
Free inodes:                 477768
First block:                 0
Block size:                  4096
Fragment size:               4096
Reserved GDT blocks:         489
Blocks per group:            32768
Fragments per group:         32768
Inodes per group:            8096
Inode blocks per group:      506
Filesystem created:          Tue Aug 31 16:35:26 2010
Last mount time:             Wed Sep  1 13:26:17 2010
Last write time:             Wed Sep  1 13:26:17 2010
Mount count:                 4
Maximum mount count:         34
Last checked:                Tue Aug 31 16:35:26 2010
Check interval:              15552000 (6 months)
Next check after:            Sun Feb 27 15:35:26 2011
Reserved blocks uid:         0 (user root)
Reserved blocks gid:         0 (group root)
First inode:                 11
Inode size:                  256
Required extra isize:        28
Desired extra isize:         28
Journal inode:               8
Default directory hash:      half_md4
Directory Hash Seed:         74c4ea07-489a-4b95-b6e7-94440eeb208f
Journal backup:              inode blocks
alpha:/dev#

```



Les utilitaires `dumpe2fs`, `debugfs` ou `debugreiserfs` permettent d'obtenir davantage d'informations de bas niveau sur les filesystems.

Conversion d'un filesystem ext2 en ext3 avec tune2fs

```
tune2fs -j device
```

Où *device* représente le fichier spécial en mode bloc qui désigne le périphérique sur lequel se trouve le filesystem à modifier.

h. Dénomination des systèmes de fichiers

Certains paramètres des systèmes de fichiers sont modifiables après leur création. Parmi ces paramètres, certains prennent de plus en plus d'importance dans les systèmes Linux modernes, et simplifieront (peut-être) les opérations de montage. Ces paramètres sont le **label** et l'**uuid**. Ils permettent de monter les filesystems locaux sans avoir à les désigner par leur fichier de bloc spécial comme `/dev/sdb1`. Si cette évolution n'est pas forcément vécue comme un progrès ni une simplification par tous, sa généralisation rend sa connaissance nécessaire.

Le label des filesystems

Comme son nom l'indique, le label est une étiquette qu'on attribue au filesystem pour le désigner de façon confortable. Le label doit être précisé par l'administrateur soit à la création du filesystem, soit après coup avec une commande de tuning. Les systèmes d'inspiration Red Hat sont les principaux utilisateurs du label.

Ajout d'un label sur un filesystem existant	
<code>tune2fs -L label device</code>	Ajoute un label au périphérique de stockage <i>device</i> .
<code>reiserfstune -l label device</code>	Ajoute un label au périphérique de stockage <i>device</i> .
<code>xfs_admin -L label device</code>	Ajoute un label au périphérique de stockage <i>device</i> .

L'UUID des filesystems

L'UUID (*Universally Unique Identifier*), comme le label, permet de désigner un périphérique de stockage par un identifiant plutôt que par son fichier de bloc spécial (`/dev/sdb1` par exemple). La différence avec le label est que l'affectation de l'uuid est automatique à la création du filesystem. Il peut néanmoins être réaffecté après-coup par les commandes de tuning des filesystems. De plus en plus de systèmes généralisent l'exploitation de l'uuid. C'est le cas notamment des distributions ubuntu.

Si vous ne savez pas comment déterminer l'UUID d'un nouveau système, n'ayez pas d'inquiétude, il est généralement créé de façon aléatoire, et sa taille (128 bits) est le garant de son unicité (probable).

Modification d'un uuid sur un filesystem existant	
<code>tune2fs -U uuid device</code>	Affectation de l'UUID <i>uuid</i> au périphérique de stockage <i>device</i>
<code>tune2fs -U random device</code>	Affectation d'un UUID aléatoire au périphérique de stockage <i>device</i>
<code>tune2fs -U time device</code>	Affectation d'un UUID basé sur l'heure de création au périphérique de stockage <i>device</i>
<code>reiserfstune -u uuid device</code>	Affectation de l'UUID <i>uuid</i> au périphérique de stockage <i>device</i>
<code>xfs_admin -U uuid device</code>	Affectation de l'UUID <i>uuid</i> au périphérique de stockage <i>device</i>

Dans le tableau ci-dessus, *device* représente le fichier spécial en mode bloc qui représente le périphérique hébergeant le filesystem sur lequel on intervient. Par exemple `/dev/sda3`.

2. Gestion du swap

a. Pourquoi le swap et en quelle quantité ?

Le swap ou mémoire virtuelle est un espace de stockage exploité pour palier à un manque de mémoire physique sur le système. Quand la mémoire physique vient à manquer pour les applications, une partie des informations stockées en mémoire et n'ayant pas fait l'objet d'une utilisation récente est déplacée sur l'espace de swap, libérant ainsi de l'espace pour les applications qui ont un besoin immédiat de mémoire. Si des applications ont besoin des informations qui ont été basculées sur l'espace de swap, le mécanisme de swap est à nouveau engagé pour libérer encore de l'espace en mémoire physique, espace dans lequel les données swappées à nouveau nécessaires seront restituées pour exploitation par les applications.

Il ne faut pas se tromper sur l'utilisation qui doit être faite du swap. En fonctionnement ordinaire, un serveur ou une station de travail Linux ne devrait pas avoir à swapper. La grande époque du swap était celle où la mémoire coûtait si cher qu'il fallait trouver pour l'équipement d'un serveur un compromis entre le coût d'un système et les performances qu'il pouvait offrir. Aujourd'hui, les coûts relativement bas de la mémoire font qu'un système ne devrait avoir à swapper qu'en cas de surconsommation accidentelle de mémoire. Le swap est donc en quelque sorte une mémoire de secours qui évite de planter un serveur en attendant la mise en adéquation entre les besoins en mémoire et les ressources disponibles.

La quantité de swap est souvent sujette à caution et selon les auteurs, les sources et les époques. Il est difficile au moment de l'installation non automatique d'un système de faire un choix serein. Un consensus semble se déterminer autour de valeurs comprises entre une et deux fois la quantité de RAM. De toute façon, les installations par défaut des distributions proposent généralement la création d'un espace de swap automatiquement. Pour une installation sur mesure, les valeurs courantes (une à deux fois la RAM) sont parfaitement acceptables, et dans le doute, l'espace

disque étant aussi très bon marché, il vaut mieux surdimensionner.

b. Optimisation du swap

Le swap est optimisable en quantité et en qualité. Il peut arriver que le swap ait été sous-dimensionné à l'installation : par exemple, on installe sur un serveur existant une application qui exige une certaine quantité de RAM et un swap dix fois supérieur à l'existant.

Par ailleurs, le swap peut être déplacé vers un espace disque plus rapide : un SAN ou une baie de disque récente et donc plus rapide que le système disque initial est installée, et l'exploitation du swap pourrait être plus rapide sur ces systèmes de stockage.

Pour ces raisons, il peut être utile de créer un nouvel espace de swap, qui s'ajoutera ou se substituera à l'espace initial.

Nature de l'espace de swap

Le swap peut être constitué de plusieurs espaces de stockage qui sont des partitions ou des fichiers. Dans la mesure où le noyau accèdera directement et exclusivement aux partitions de swap, les performances seront meilleures qu'avec un fichier de swap où le filesystem représente un intermédiaire supplémentaire vers le stockage physique.

Si le swap est placé sur une partition, elle doit avoir été créée de type 82 avec un outil de partitionnement adéquat (fdisk Linux par exemple). Si c'est un fichier, il doit simplement être accessible en permanence sur un filesystem toujours monté.

Création de l'espace de swap

Pour pouvoir être exploité, l'espace de swap doit être préparé, un peu comme on créerait un filesystem sur un espace de stockage brut. Cette préparation se fait avec la commande **mkswap**, et elle peut être appliquée aussi bien à une partition qu'à un fichier de taille déterminée.

Syntaxe de la commande mkswap

```
mkswap espace_stockage
```

Où *espace_stockage* représente l'emplacement physique de l'espace de swap dont la dénomination peut se faire de différentes façons :

Désignations possibles des espaces de stockage pour la commande mkswap	
/chemin/fichier	Structure le fichier afin qu'il puisse être exploité en tant qu'espace de swap.
/dev/device	Structure l'espace de stockage désigné par le fichier spécial en mode bloc afin qu'il puisse être exploité en tant qu'espace de swap.
-L LABEL	Structure l'espace de stockage désigné par le label LABEL afin qu'il puisse être exploité en tant qu'espace de swap.
-U UUID	Structure l'espace de stockage désigné par l'uuid UUID afin qu'il puisse être exploité en tant qu'espace de swap.

Exploitation du swap

Une fois l'espace de swap créé, il doit être rendu accessible au noyau par la commande **swapon**. Le système sera alors capable de swapper à partir du nouvel espace créé.

Syntaxe de la commande swapon pour activer un espace de swap

```
swapon espace_stockage
```

Où *espace_stockage* représente l'emplacement physique de l'espace de swap dont la dénomination peut se faire de différentes façons :

Désignations possibles des espaces de stockage pour la commande swapon	
/chemin/fichier	Rend le fichier utilisable pour le swap par le noyau.

/dev/device	Rend l'espace de stockage désigné par le fichier spécial en mode bloc utilisable pour le swap par le noyau.
-L LABEL	Rend le stockage dont le label est LABEL utilisable pour le swap par le noyau.
-U UUID	Rend le stockage dont l'uuid est UUID utilisable pour le swap par le noyau.

Désactivation d'un espace de swap

Si on souhaite que le système arrête d'exploiter un espace de swap, il faut le lui signifier avec la commande **swapoff**.

Syntaxe de la commande swapoff pour désactiver un espace de swap

```
swapoff espace_stockage
```

Où *espace_stockage* représente l'emplacement physique de l'espace de swap dont la dénomination peut se faire de différentes façons :

Désignations possibles des espaces de stockage pour la commande swapoff	
/chemin/fichier	Arrête l'exploitation de l'espace de swap sur le fichier.
/dev/device	Arrête l'exploitation de l'espace de swap sur le device.
-L LABEL	Arrête l'exploitation de l'espace de swap sur le stockage dont le label est LABEL.
-U UUID	Arrête l'exploitation de l'espace de swap sur le stockage dont l'uuid est UUID.

Visualisation des espaces de swap

L'ensemble des espaces de swap exploités, ainsi que leur nature (fichier ou partition) peuvent être affichés avec les commandes **swapon** et **swapoff** évoquées précédemment.

Syntaxe de la commande swapon pour visualiser la configuration du swap

```
swapon -s
```

Exemple d'utilisation de la commande swapon

La commande indique la partition ou le fichier utilisé, la taille réservée et la quantité de swap utilisée.

```
A:~# swapon -s
Filename      Type      Size      Used      Priority
/dev/hda5     partition 409616    608      -1
```

Autre visualisation du swap

Il est également possible de visualiser la configuration du swap en consultant le contenu du fichier swap du filesystem virtuel /proc.

```
toto@cuicui:~$ cat /proc/swaps
Filename      Type      Size      Used      Priority
/dev/sda3     partition 10112908 2024
-1
```

3. Montage des filesystems

a. Montage et démontage

La commande **mount** permet de monter le filesystem d'un périphérique de stockage sous un répertoire local, généralement vide. Au minimum, il faut fournir comme argument à la commande **mount** le périphérique hébergeant le filesystem, et le répertoire qui constituera son point de montage.


La commande **umount** réalise l'opération inverse. Elle accepte comme argument le point de montage, ou le périphérique physique à démonter.

Montage d'un filesystem

```
mount -t type_fs -o options device point_montage
```

commande mount : options et paramètres	
<i>type_fs</i>	Facultatif : type de filesystem à monter.
<i>options</i>	Facultatif : options de montage.
<i>device</i>	Le périphérique hébergeant un filesystem à monter, sous forme de fichier spécial bloc.
<i>point_montage</i>	Le répertoire qui servira de point d'ancrage au filesystem monté.

Les options les plus courantes sont **ro** (lecture seule), **sync** (écritures synchrones sans passer par un cache mémoire), et **loop** (montage de données de fichiers plutôt que de filesystems).

 Le montage d'un filesystem avec l'option **sync** permet de s'affranchir de toute forme de cache en écriture sur le disque, et ainsi de fiabiliser les opérations d'écriture. La commande **sync** permet de vider ponctuellement le cache sur un filesystem qui ne bénéficie pas de cette option de montage.

Démontage d'un filesystem

```
umount -O options device point_montage
```

commande umount : options et paramètres	
<i>options</i>	Facultatif : options de démontage.
<i>device</i>	Facultatif si le point de montage est précisé : le périphérique à démonter.
<i>point_montage</i>	Facultatif si le périphérique est précisé : le répertoire servant de point de montage à libérer.

Les options les plus courantes sont **-f** (force : forcer le démontage) et **-l** (lazy : démontage paresseux qui sera effectif quand toutes les ressources utilisées pour le montage auront pu être libérées).

Le démontage d'un filesystem est indispensable pour en effectuer la vérification avec la commande **e2fsck**. Le filesystem monté sur **/** est par définition indémontable puisque toujours occupé. Il est possible de forcer la vérification avant le montage lors du démarrage depuis la commande **shutdown**.

Vérification du filesystem racine avant montage

```
shutdown -F -r now
```

b. Visualisation des filesystems montés

La commande **mount** sans argument permet de visualiser les filesystems montés.

Par ailleurs, chaque montage réussi provoque l'écriture d'une ligne correspondante dans le fichier **/etc/mtab**. L'affichage du fichier **/proc/mounts** renvoie la même information.

c. Fichier fstab

Le fichier **/etc/fstab** permet de désigner des filesystems à monter ou des espaces de swap à activer automatiquement au démarrage. Accessoirement, il permet aussi de désigner des filesystems éventuellement montables, comme pour les périphériques amovibles par exemple. La syntaxe de la commande **mount** appelée ponctuellement en sera alors fortement simplifiée.

Le fichier **/etc/fstab** doit comporter sur chaque ligne l'ensemble des éléments nécessaires au montage d'un filesystem, à savoir le point de montage, la désignation de l'espace de stockage, et les options de montage. Pour les espaces de swap, la désignation du point de montage sera sans objet.

Le fichier **/etc/fstab** est composé d'une ligne par filesystem à monter, chaque ligne étant composée de six champs obligatoires.

Format type d'une ligne de déclaration de montage dans /etc/fstab

```
fs pointmontage type options dump fsck
```

Les champs sont séparés par des espaces ou des tabulations.

Fichier /etc/fstab : format des lignes de définition des montages		
Numéro de champ	Champ	Désignation
1	<i>fs</i>	Filesystem, désigné par son fichier de bloc spécial, son label ou son uuid.
2	<i>pointmontage</i>	Point de montage.
3	<i>type</i>	Type de filesystem. Obligatoirement swap pour le swap, auto ou type effectif de filesystem dans le cas contraire.
4	<i>options</i>	Options de montage. En fait, les options admises par la commande mount.
5	<i>dump</i>	Facultatif. Si la commande dump est utilisée pour la sauvegarde du système, ce champ doit être à 1 pour assurer la sauvegarde. Sinon, sa valeur par défaut est 0.
6	<i>fsck</i>	Facultatif. En cas de vérification automatique des filesystems au démarrage, indique dans quel ordre cette vérification doit se faire. Valeur obligatoire de 1 pour le filesystem monté sur /, 2 pour les autres. 0 pour que la vérification ne soit jamais effectuée.

Exemple de fichier /etc/fstab sur Ubuntu

Notez que les disques sont identifiés par leur uid.

```
# /etc/fstab: static file system information.
#
# <file system> <mount point> <type> <options> <dump>
# <pass>
proc /proc proc defaults 0 0
# /dev/sda2
UUID=52200c0b-ae8-4ae0-9492-1f488051e4a3 / ext3
relatime,errors=remount-ro 0 1
# /dev/sdb1
UUID=b0891c0e-1812-4d23-b77d-b861f7fd2713 /home ext3
relatime,errors=remount-ro 0 2
# /dev/sda3
UUID=ee7890fb-c312-406f-b100-669c97ee8d07 none swap
sw 0 0
/dev/scd0 /media/cdrom0 udf,iso9660 user,noauto,exec,utf8
0 0
```

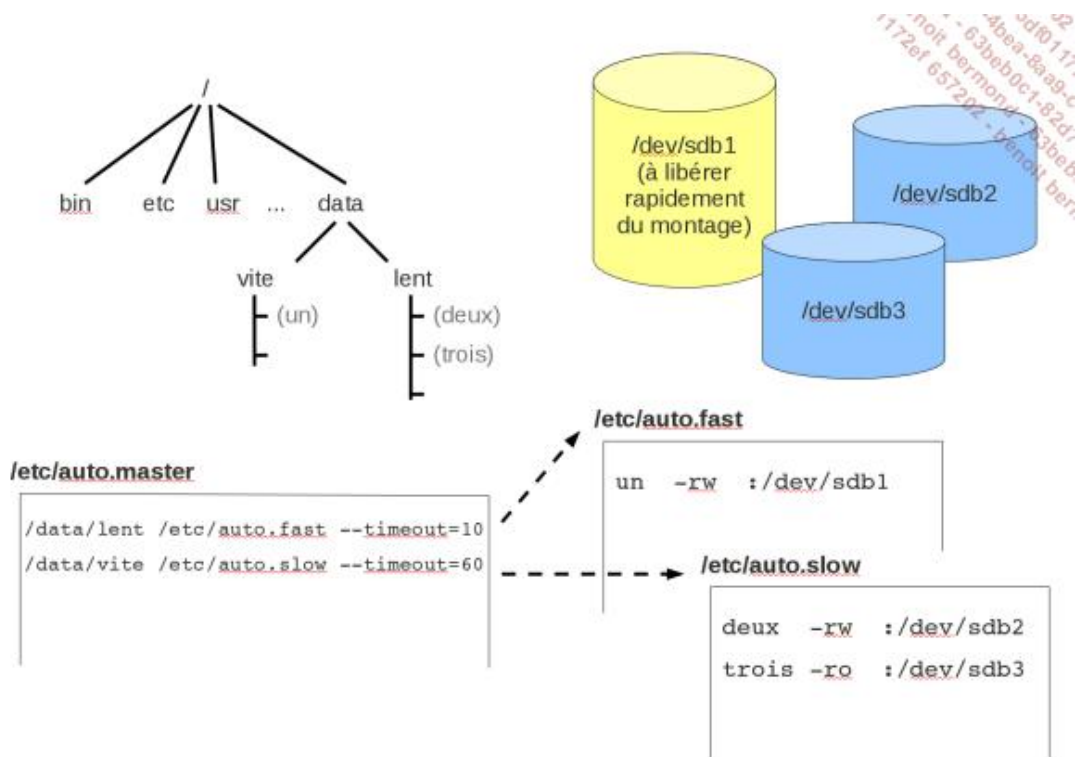
Les disques sont identifiés par leur label.

```
LABEL=/          /          ext3    defaults
1 1
LABEL=/boot      /boot      ext3    defaults
1 2
/dev/hda3        swap       swap    defaults
0 0
/dev/cdrom       /mnt/cdrom udf,iso9660
noauto,owner,kudzu,ro 0 0
/dev/fd0         /mnt/floppy auto
noauto,owner,kudzu 0 0
```

La commande **mount -a** est appelée au démarrage d'un système. Cette commande provoque le montage de tous les périphériques référencés dans le fichier **/etc/fstab**, à l'exception de ceux qui présentent l'option **noauto** dans le quatrième champ.

d. Automontage

Le montage peut être une opération pénible à réaliser pour l'opérateur. Certaines fonctionnalités optionnelles permettent de s'affranchir dans une certaine mesure de la connaissance des fonctions et commandes de montage. Les bureaux graphiques Gnome ou KDE par exemple gèrent depuis longtemps le montage automatique des périphériques amovibles insérés. L'automontage est une technique qui permet de monter à la volée un filesystem en fonction de l'accès qui y est fait par l'utilisateur. L'automontage est en voie de raréfaction sur les postes de travail autonomes, mais très efficace sur les systèmes de fichiers distribués.



Configuration de l'automontage

L'automontage s'appuie sur deux fichiers de paramétrage : les tables d'automontage, et sur un service qui vérifie en permanence s'il est besoin de réaliser des opérations de montage.

La première table d'automontage, la table maîtresse est configurée dans le fichier **/etc/auto.master**. Elle précise l'ensemble des répertoires soumis à automontage, c'est-à-dire les répertoires dans lesquels pourront avoir lieu des montages automatiques si une application fait appel à un contenu de ce répertoire. On créera dans ce fichier autant de lignes qu'il y aura de répertoires à surveiller.

Format de la table maîtresse d'automontage (`/etc/auto.master`)

`repertoire` `fic_sec` `--option=valeur`

Fichier <code>/etc/auto.master</code> : directives et variables utilisées	
<i>repertoire</i>	Le répertoire dans lequel les accès seront surveillés pour voir s'il y a lieu de procéder au montage.
<i>fic_sec</i>	Le fichier de table secondaire qui précise les montages à réaliser pour le répertoire.
<i>option</i>	Option liée à la gestion de l'autofs. Option courante à connaître : <code>timeout</code> . La valeur est alors le nombre de secondes avant le démontage en cas d'inactivité.

Le nom des fichiers de table secondaire est libre, même s'il porte généralement le préfixe « `auto.` » et se situe dans `/etc`. Il faudra autant de fichiers secondaires qu'on en aura décrit dans la table maîtresse. Dans bien des cas, une table secondaire unique est suffisante. Chaque table secondaire correspond au chargement d'un démon indépendant.

Format de fichier d'une table secondaire

`pmv` `options` `device`

Fichier de table secondaire d'automontage : directives et variables	
<i>pmv</i>	Point de montage virtuel : le répertoire virtuel dont l'accès par une application provoquera le montage.
<i>options</i>	Les options de montage, précédées par un tiret et séparées par des virgules.
<i>device</i>	Le périphérique à monter.

Gestion du service d'automontage

Pour prendre en compte la nouvelle configuration, il faudra redémarrer le service. Le script de lancement du service s'appelle généralement **autofs**, et il est situé dans **`/etc/init.d`**.



Tout changement de table maîtresse doit s'accompagner d'un redémarrage du service.

Redémarrage du service

```
/etc/init.d autofs restart
```

Visualisation de la configuration

```
/etc/init.d autofs status
```

Fonctionnement de l'automontage

Pour tester la configuration de votre automontage, suivez les étapes suivantes :

- Créez le répertoire de travail. Ne créez pas les points de montage, ils ne doivent pas exister.
- Renseignez les fichiers de configuration.
- Redémarrez le service.
- Depuis le shell, positionnez-vous en aveugle dans le point de montage virtuel, celui qui est décrit dans le fichier de table secondaire.
- Vérifiez que le montage a bien eu lieu de façon transparente.

4. Gestion des disques durs

Dans la plupart des situations, les connaissances courantes sur la dénomination courante des disques durs (hda, sda) sont suffisantes, et on se concentre surtout sur la façon de les exploiter sous forme de partitions ou volumes logiques. Il arrive toutefois qu'il soit nécessaire de paramétrer les disques durs du point de vue matériel, pour optimiser les performances ou pour détecter des défaillances.

a. Détermination des fichiers spéciaux

Il y a quelque temps encore, les systèmes Linux contenaient dans leur répertoire /dev l'ensemble des fichiers spéciaux pour tous les périphériques gérables par le noyau. Avec le noyau 2.6 est arrivé le service **udev**, qui a pour tâche de gérer dynamiquement la création de fichiers spéciaux à la découverte d'un périphérique.

Du point de vue de l'utilisateur ordinaire, le service **udev** travaille dans l'ombre et le mieux est de ne pas s'en soucier : les fichiers spéciaux sont présents quand on en a besoin et il n'y a rien à vouloir de plus. En revanche, l'administrateur ou l'utilisateur avancé peut créer des règles comportementales qui permettent de déclencher des actions en fonction d'événements liés au stockage. L'emplacement de ces règles est précisé dans le fichier de configuration de udev : **/etc/udev/udev.conf**. En l'absence d'information, c'est la valeur par défaut qui est employée, à savoir **/etc/udev/rules.d**.

Exemple de règle udev

Dans cette configuration standard d'une distribution Ubuntu, on voit (après quelques efforts d'interprétation) que le système génère des liens symboliques pour les différentes appellations courantes du lecteur de media optique.

```
root@serveur # cat /etc/udev/rules.d/70-persistent-cd-rules
ENV{ID_CDROM}=="?*", ENV{ID_PATH}=="pci-0000:00:14.1-scsi-
0:0:0:0", SYMLINK+="cdrom", ENV{GENERATED}="1"
ENV{ID_CDROM}=="?*", ENV{ID_PATH}=="pci-0000:00:14.1-scsi-
0:0:0:0", SYMLINK+="cdrw", ENV{GENERATED}="1"
ENV{ID_CDROM}=="?*", ENV{ID_PATH}=="pci-0000:00:14.1-scsi-
0:0:0:0", SYMLINK+="dvd", ENV{GENERATED}="1"
ENV{ID_CDROM}=="?*", ENV{ID_PATH}=="pci-0000:00:14.1-scsi-
0:0:0:0", SYMLINK+="dvdrw", ENV{GENERATED}="1"
```

b. Informations sur les périphériques de stockage

Grâce au service udev, on ne trouve plus dans le répertoire /dev que les périphériques réellement présents sur le système. Cela constitue naturellement un premier niveau d'informations.

La solution la plus simple pour obtenir plus de détails est d'exploiter la commande **dmesg** qui consigne tous les messages renvoyés par le noyau depuis son démarrage. On dit que la commande **dmesg** affiche le **ring-buffer** du noyau.

Utilisation de dmesg pour identifier les disques durs

Il est vivement recommandé de filtrer la sortie de la commande dmesg, celle-ci étant par nature assez bavarde.

```
alpha:~# dmesg | grep [sh]d
[ 0.000000] Kernel command line: root=/dev/hda1 ro quiet
[ 3.136965] hda: VBOX HARDDISK, ATA DISK drive
[ 3.822425] hda: host max PIO4 wanted PIO255(auto-tune)
selected PIO4
[ 3.822677] hda: UDMA/33 mode selected
[ 4.575784] hdc: VBOX CD-ROM, ATAPI CD/DVD-ROM drive
[ 5.275977] hdc: host max PIO4 wanted PIO255(auto-tune)
selected PIO4
[ 5.275977] hdc: UDMA/33 mode selected
[ 7.203721] hda: max request size: 128KiB
[ 7.203728] hda: 16777216 sectors (8589 MB) w/256KiB Cache,
CHS=16644/16/63
[ 7.204020] hda: cache flushes supported
[ 7.204020] hda: hda1 hda2 < hda5 >
[ 7.234912] hdc: ATAPI 32X DVD-ROM drive, 128kB Cache
```

```
[ 7.257272] Driver 'sd' needs updating - please use bus_type methods
[ 7.257525] sd 0:0:0:0: [sda] 4194304 512-byte hardware sectors
(2147 MB)
[ 7.257620] sd 0:0:0:0: [sda] Write Protect is off
[ 7.257627] sd 0:0:0:0: [sda] Mode Sense: 00 3a 00 00
[ 7.257769] sd 0:0:0:0: [sda] Write cache: enabled, read cache:
enabled, doesn't support DPO or FUA
(...)
```

Récupération d'informations sur un périphérique par la commande udevadm

Le service **udev** peut aussi nous fournir des informations précieuses par le biais de sa commande d'administration **udevadm**.

```
alpha:~# udevadm info --query=all --name=/dev/hda
P: /block/hda
N: hda
S: block/3:0
S: disk/by-id/ata-VBOX_HARDDISK_VBf92d3e4d-7faf607b
S: disk/by-path/pci-0000:00:01.1-ide-0:0
E: ID_TYPE=disk
E: ID_MODEL=VBOX_HARDDISK
E: ID_SERIAL=VBf92d3e4d-7faf607b
E: ID_REVISION=1.0
E: ID_BUS=ata
E: ID_PATH=pci-0000:00:01.1-ide-0:0
```

Surveillance d'événements par la commande udevmonitor (ou udevadm monitor)

On peut surveiller quasiment en temps réel les événements système.

```
toto@ubuntu:~$ udevmonitor
monitor will print the received events for:
UDEV - the event which udev sends out after rule processing
KERNEL - the kernel uevent

KERNEL[1276268963.339194] change
/devices/pci0000:00/0000:00:14.1/host4/target4:0:0/4:0:0:0 (scsi)
KERNEL[1276268963.339804] change
/devices/pci0000:00/0000:00:14.1/host4/target4:0:0/4:0:0:0/block/sr0 (block)
(...)
```

Visualisation des paramètres de périphériques avec lsdev

La commande **lsdev** permet de récupérer des informations sur les périphériques reconnus, notamment les valeurs DMA, IRQ et I/O. Ces valeurs sont lues dans les fichiers **/proc/interrupts**, **/proc/ioports**, et **/proc/dma**.

```
toto@ubuntu:~$ lsdev
Device          DMA   IRQ  I/O Ports
-----
0000:00:01.1          0170-0177 01f0-01f7 0376-0376 03f6-03f6
d000-d00f
0000:00:03.0          d020-d03f
0000:00:04.0          d040-d05f
0000:00:05.0          d100-d1ff d200-d23f
82801AA-ICH          5
ACPI          4000-4003 4004-4005 4008-400b 4020-4021
ata_piix          14 15    0170-0177 01f0-01f7 0376-0376 03f6-03f6
d000-d00f
cascade          4    2
eth0              10
floppy            2    6    03f2-03f2 03f4-03f5 03f7-03f7
vga+              03c0-03df
toto@ubuntu:~$
```

Enfin, on trouve sous le répertoire **/dev/disk** les éléments de stockage apparaissant selon la façon dont ils sont reconnus et identifiés par le système.

```
root@serveur:/dev/disk$ ls
by-id by-label by-path by-uuid
root@serveur:/dev/disk$ cd by-uuid
root@serveur:/dev/disk/by-uuid$ ls
52200c0b-ae8-4ae0-9492-1f488051e4a3 B0F82CDF82CA318
b0891c0e-1812-4d23-b77d-b861f7fd2713 ee7890fb-c312-406f-b100-669c97ee8d07
root@serveur:/dev/disk/by-uuid$ file *
52200c0b-ae8-4ae0-9492-1f488051e4a3: symbolic link to `../../sda2'
b0891c0e-1812-4d23-b77d-b861f7fd2713: symbolic link to `../../sdb1'
B0F82CDF82CA318: symbolic link to `../../sda1'
ee7890fb-c312-406f-b100-669c97ee8d07: symbolic link to `../../sda3'
root@serveur:/dev/disk/by-uuid$
```

c. Gestion des performances avec hdparm

La commande **hdparm** permet de consulter et configurer de nombreux paramètres du disque dur, certains d'ailleurs dangereux pour le disque.

Visualisation des paramètres fonctionnels avec hdparm

```
alpha:~# hdparm /dev/hda
/dev/hda:
multcount      = 0 (off)
IO_support     = 0 (default)
unmaskirq     = 0 (off)
using_dma      = 1 (on)
keepsettings   = 0 (off)
readonly       = 0 (off)
readahead      = 256 (on)
geometry       = 16644/16/63, sectors = 16777216, start = 0
```

Si le matériel le supporte, les paramètres fonctionnels du disque peuvent être modifiés avec l'option appropriée suivie d'un paramètre numérique, en général 0 ou 1. Les options les plus courantes sont **c** (activation ou désactivation de l'accès 32 bits au disque) et **d** (activation ou désactivation de l'accès DMA). Une option demandée sans valeur numérique associée entraîne l'affichage de la valeur courante.

Consultation de l'accès 32 bits

```
alpha:~# hdparm -c /dev/hda
/dev/hda:
IO_support     = 0 (default)
```

Consultation puis suppression de la lecture anticipée

```
alpha:~# hdparm -a /dev/hda
/dev/hda:
readahead      = 256 (on)
alpha:~# hdparm -a 0 /dev/hda
/dev/hda:
setting fs readahead to 0
readahead      = 0 (off)
```

Visualisation, activation et désactivation de l'accès DMA

```
alpha:~# hdparm -d /dev/hda
/dev/hda:
using_dma      = 0 (off)
alpha:~# hdparm -d 1 /dev/hda
```

```
/dev/hda:
setting using_dma to 1 (on)
using_dma      = 1 (on)
alpha:~# hdparm -d 0 /dev/hda
/dev/hda:
setting using_dma to 0 (off)
using_dma      = 0 (off)
```



Une autre commande : **sdparm**, moins courante permet une communication de bas niveau avec les périphériques SCSI, par exemple pour réaliser leur désactivation pour retrait à chaud.

d. Gestion des défaillances matérielles

Nous avons vu que la commande **fsck** permettait de vérifier la cohérence d'un filesystem. Si une incohérence est due à un problème de gestion de l'écriture (arrêt électrique lors d'une opération d'écriture), **fsck** peut essayer de récupérer tant bien que mal la situation et une fois résolu, le problème peut être oublié. En revanche, la défaillance physique d'un disque, liée à un défaut de la surface magnétique par exemple, doit être traitée de façon adéquate pour éviter toute conséquence ultérieure. La commande **badblocks** référence les blocs physiquement défectueux sur un disque ou une partition. La liste des blocs défectueux est envoyée sur la sortie standard, mais il est courant d'utiliser un fichier qui sera exploitable par les programmes **e2fsck** ou **mke2fs**. Dans ce cas, il faut préciser la taille des blocs employés pour éviter tout ennui. Si le but est simplement de vérifier l'absence de défaut sur le disque, la commande **badblocks** peut être utilisée sans aucune option.

Détection des blocs en erreur avec badblocks

```
badblocks -b taille_blocks -o fichier_sortie
```

Où *taille_blocks* représente la taille des blocs du système de fichiers et *fichier_sortie* le fichier qui consignera l'ensemble des blocs altérés.