

I – Systèmes de fichiers :

1 – Présentation :

Un système de fichier (FS pour File System) définit la façon dont seront stockées les informations sur les mémoires de masse (périphériques de stockage type disque dur) ainsi que l'organisation des fichiers. Il permet aussi à l'utilisateur d'accéder aux fichiers de manière transparente.

Chaque système de fichier est plus ou moins lié à un système d'exploitation particulier :

- Windows :
 - o FS d'installation : FAT, NTFS
 - o FS reconnus en natif : FAT, NTFS, ExFAT
- Linux :
 - o FS d'installation : ext, ReiserFS, etc...
 - o FS reconnus en natif : tous (ou presque)

Le FS utilisé influera notamment sur :

- la taille maximale des fichiers
- le nombre maximal de fichiers
- la taille des partitions

2 – Principes de base :

a – Le bloc :

C'est l'unité de base, indivisible, du système de fichier. Par conséquent, un fichier sera composé d'un nombre fini de blocs, et d'au moins 1 bloc.

Le choix de la taille d'un bloc est donc primordiale sur les performances d'un système de fichiers :

- Si on choisit un bloc de petite taille, lors de l'allocation de l'espace pour un grand fichier, celui-ci risque d'être morcelé à plusieurs endroits du disque. L'accès au fichier sera donc long à cause des déplacements sur le disque.
- Si on choisit un bloc de grande taille, on gaspillera de l'espace disque pour les petits fichiers. En effet, soit un disque de 500Go avec des blocs de 8ko. On aura donc $536\,870\,912\,000 / 8\,192 = 65\,536\,000$ blocs. On pourra donc remplir entièrement le disque avec 65 536 000 fichiers de 1 octet, en n'utilisant effectivement que 62,5 Mo du disque.

Or, statistiquement, la majorité des fichiers sont des fichiers de petite taille.

Sous Linux, la taille de blocs par défaut est de 4096 octets (4ko).

b – Le super bloc :

Les systèmes de fichiers sous Linux possède tous au moins un superbloc. Celui-ci contient des informations sur le système de fichier lui-même :

- Son type
- Sa taille
- Son statut
- La localisation d'autres zones de méta-données (superblocs secondaires, tables d'inodes)

Le superbloc est une zone critique du disque : sa corruption peut entraîner une impossibilité de lire les données stockées sur le disque. C'est pour cela qu'il en existe des copies synchrones sur le disque : si le superbloc primaire ne peut être lu, alors c'est le secondaire qui sera utilisé.

c - Les inodes :

Un inode (ou i-nœud) est une structure de données associée à un fichier physique sur le disque, et un seul, contenant des informations sur ce fichier et son contenu. Les inodes sont stockés dans la table d'inode située après le superbloc.

On peut connaître le numéro de l'inode associé à un fichier avec l'option `-li` de la commande `ls`.

Exemple :

```
user1@debian7:~/rep1$ ls -li
total 4
132344 -rw-r--r-- 3 user1 user1 84 oct. 26 18:20 fic1
132336 -rw-r--r-- 2 user1 user1 168 oct. 26 17:02 fic2
392635 -rw-r--r-- 1 user1 user1 0 nov. 16 21:31 fic3
```

NB : les fichiers du FS qui sont des liens physiques vers le même fichier physique ont le même numéro d'inode.
--

La norme POSIX définit les attributs qu'un inode doit au minimum posséder pour un fichier :

- taille du fichier
- identifiant du périphérique où se trouve le fichier
- propriétaire du fichier
- groupe du fichier
- numéro d'inode du fichier
- droits du fichier
- date de dernière modification de l'inode (change time)
- date de dernière modification du contenu du fichier (modification time)
- date de dernier accès au fichier (access time)
- nombre de liens physiques sur le fichier

On remarque que l'inode ne contient pas le nom du fichier.

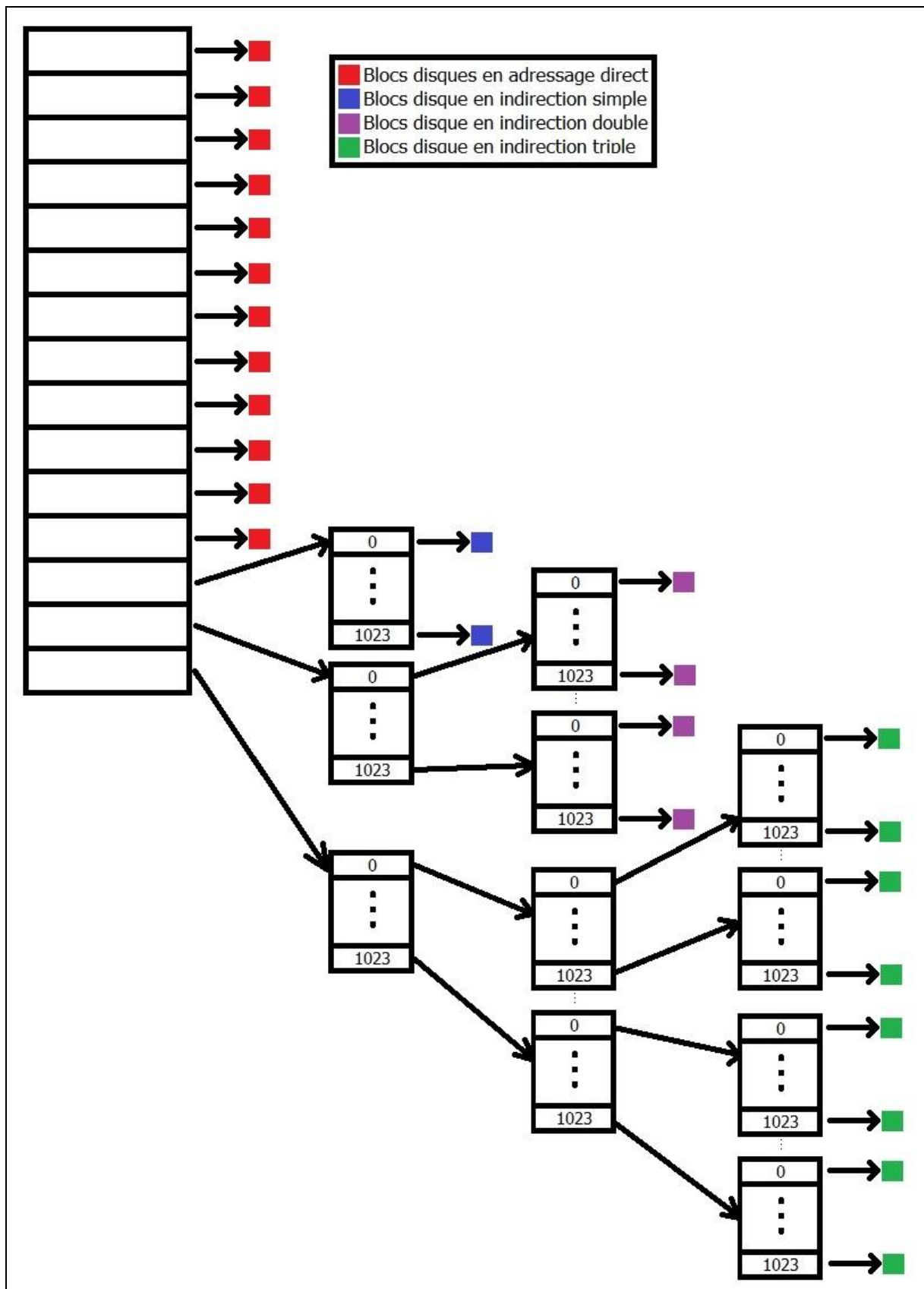
On peut obtenir les informations données ci-dessus avec la commande `stat`.

Adresses des blocs disques :

Un inode contient aussi les adresses des blocs disque composant le fichier physique.

En ext3, un inode contient un tableau de 15 pointeurs :

- les 12 premiers pointent directement sur un bloc disque.
- Le 13^{ième} pointe sur un tableau de 1024 cases pointant chacune sur un bloc disque (simple indirection).
- Le 14^{ième} pointe sur un tableau de 1024 cases pointant chacune sur un tableau de 1024 cases pointant chacune sur un bloc disque (double indirection).
- Le 15^{ième} pointe sur un tableau de 1024 cases pointant chacune sur un tableau de 1024 cases pointant chacune sur un bloc disque (triple indirection).



Cette méthode a de nombreux avantages :

- les fichiers de petite taille (les plus nombreux) sont accessibles rapidement en adressage direct (avec des blocs de 4K, jusqu'à une taille de $4 \times 12 = 48\text{ko}$) ou en indirection simple (jusqu'à une taille de $(1024 + 12) \times 4 = 4144\text{ko} \approx 4\text{Mo}$).
- Le système peut gérer des fichiers de grande taille (jusqu'à une taille de $(12 + 1024 + 1024^2 + 1024^3) \times 4 = 4\text{To}$).

d – Les répertoires :

Comme on l'a vu, le nom du fichier ne se trouve pas dans l'inode. Il se trouve dans le répertoire que l'on peut voir comme un tableau à 2 colonnes qui contient les inodes et les noms des fichiers associés qui se trouvent dans le répertoire.

3 – Les différents systèmes de fichiers sous Linux :

ext4 est le système de fichiers par défaut des systèmes Linux récents. Il est considéré comme stable depuis le noyau 2.6.28.

ext4 :

- supporte les types de fichiers standards Unix suivants : fichiers réguliers, répertoires, fichiers périphériques spéciaux et liens symboliques.
- gère les noms de fichiers longs.
- réserve certains blocs pour le super utilisateur (root).
- permet à l'administrateur de choisir la taille des blocs logiques lors de la création du système de fichier.
- implémente des liens symboliques rapides. Un lien symbolique rapide n'utilise aucun bloc de donnée sur le système de fichiers.
- permet la journalisation.
- supporte les volumes de très grande capacité (jusqu'à 1 Eo (2^{60} octets)) et les fichiers jusqu'à 16 To).
- Supporte les extents : l'allocation des blocs est contigüe et permet donc de réduire la fragmentation tout en augmentant les performances sur les gros fichiers. De plus, il faut moins de métadonnées pour stocker les informations des blocs grâce à leur contigüité.
- est rétro-compatible avec ext2 et ext3 qui peuvent être montés en ext4.

ReiserFS est un autre système de fichiers utilisé sous Linux, même s'il l'est beaucoup moins que les ext.

ReiserFS :

- est très efficace sur les petits fichiers.
- permet l'augmentation à chaud des tailles de partitions.
- permet une journalisation rapide.
- N'a pas de gestion de quotas.

4 – Dénomination des disques :

Les fichiers désignant les périphériques se trouvent dans le répertoire /dev. La nomenclature pour désigner les disques (ou les partitions) est très précise sous linux et a la forme suivante :

/dev/sdXY

sd indique que c'est un périphérique de stockage de masse (mass-**s**torage **d**river).

X désigne le contrôleur physique du disque : a pour le premier disque du connecteur primaire SATA, b pour le second disque du connecteur primaire SATA, c pour le premier disque du connecteur secondaire SATA, d pour le second disque du connecteur secondaire SATA, etc...

Y indique le numéro de la partition dans le disque. De 1 à 4, ce seront les partitions primaires et l'éventuelle partition étendue (qui pourra être divisée). A partir de 5, ce seront les partitions logiques contenues dans la partition étendue.

On peut voir la liste des partitions d'un système linux avec la commande :

fdisk -l

On a alors sur chaque ligne du résultat, dans l'ordre :

- la dénomination du périphérique
- une indication sur la possibilité de démarrer sur le périphérique (* si oui)
- le secteur de début de la partition
- le secteur de fin de la partition
- le nombre de blocs de la partition
- l'identifiant du type de la partition (83=Linux, 7=NTFS, 82=swap...)

5 – Création d'un système de fichiers :

C'est la commande mkfs (**make file system**) qui permet de créer un système de fichier sur un disque.

Il n'y a pas de formatage à proprement parler sous Linux, l'appel à mkfs écrit le système de fichier sans la phase préalable de remplissage du disque par des zéros et la vérification des secteurs du formatage sous windows.

Syntaxe :

mkfs -t typeFS options périphérique

En fait, mkfs est un alias : grâce au paramètre typeFS, ce sont les commandes mkfs.typeFS qui seront appelées : mkfs.ext4 sera exécutée quand on exécutera mkfs -t ext4.

Les options dépendent du système de fichiers à créer.

Pour ext3 et ext4 :

- b *n* : *n* sera la taille des blocs en octets (multiple de 512)
- c : vérifie s'il y a des mauvais blocs avant de créer le système de fichiers
- i *n* : *n* donnera la quantité d'octets par inode. Plus la valeur est grande, moins il y aura d'inodes, donc on pourra créer moins de fichiers, mais on économise de l'espace.
- m *n* : *n* sera le pourcentage d'espace disque réservé au super-utilisateur
- L *nom* : affecte le label *nom* au disque.

Le périphérique sera donné sous la forme /dev/sdXY vue précédemment.

II – Montage d'un périphérique :

1 – Définition :

Avant de pouvoir accéder à un périphérique de stockage, il faut le monter. C'est-à-dire qu'il faut lui attribuer un point d'entrée dans l'arborescence du système de fichiers.

Le point d'entrée sera un répertoire déjà existant du système de fichiers, qui sera appelé point de montage.

Si le point de montage n'est pas vide, alors son contenu sera inaccessible le temps que le périphérique y sera monté. On pourra de nouveau y accéder une fois le périphérique démonté.

2 – La commande mount :

Sans argument, ni option, elle affiche la liste des partitions montées sur le système, avec les options de montage associées, et les points de montage.

Syntaxe :

`mount -t typeFS options_de_mount -o options_de_montage périphérique point_de_montage`

Une option très utile de mount est :

`-L label`

qui permet de monter un périphérique sans connaître sa dénomination, uniquement avec le label défini lors de la création du système de fichiers. Cela est notamment très pratique si l'on change l'ordre des disques.

Options de montage :

`sync/async` : les écritures sur le périphérique seront effectuées en temps réel (`sync` : plus sûr mais plus lent) ou différées via un tampon (`async` : plus performant mais moins sûr).

`exec/noexec` : il sera possible de lancer un exécutable depuis le périphérique (`exec`) ou pas (`noexec`).

`auto/noauto` : le système de fichier sera monté automatiquement (`auto`) ou devra l'être explicitement (`noauto`)

`user/nouser` : n'importe qui peut monter le système de fichiers (`user`) ou seul le root le peut (`nouser`)

`ro/rw` : montage en lecture seule (`ro`) ou en lecture/écriture (`rw`)

Pour modifier une option d'un système déjà monté, on utilise l'option `remount` en plus de l'option concernée.

Démontage d'un système de fichiers :

La commande `umount` détache un périphérique d'un point de montage.

Syntaxe :

`umount point_de_montage`

`umount` ne fonctionnera pas si au moins un fichier du périphérique concerné est utilisé par un processus.

La commande `lsuf` permet de savoir quel(s) processus utilise(nt) le système de fichier :

Syntaxe :

`lsuf point_de_montage`

La commande `fuser` avec l'option `-k` permet de tuer tout processus accédant à un point de montage particulier :

`fuser -k point_de_montage`

3 – Le fichier `/etc/fstab` :

Il contient une configuration des différents systèmes de fichiers à monter au démarrage du système, avec toutes les informations nécessaires.

Chaque ligne contient 6 champs :

périphérique	point_de_montage	typeFS	options_de_montage	dump	pass
--------------	------------------	--------	--------------------	------	------

où - périphérique :

- dénomination du périphérique à monter (`/dev/sdXY`)
- label du périphérique à monter (`LABEL=nom_du_label`)
- UUID du périphérique à monter (`UUID=valeur_de_l'uuid`)

- `point_de_montage` : répertoire où sera monté le périphérique.
- `typeFS` : type du système de fichier du périphérique à monter.
- `options_de_montage` : options à appliquer lors du montage du périphérique.
- `dump` : exécution de l'outil de sauvegarde *dump* (0 : non, 1 : oui).
- `pass` : priorité d'exécution de l'outil de vérification `fsck` (0 : pas de vérification, 1 : plus grande priorité pour `/`, 2 : priorité moindre).

NB : on trouve les labels et UUIDs des différents disques dans les répertoires `/dev/disk/by-label` et `/dev/disk/by-uuid`.

Tous les systèmes de fichiers contenus dans `/etc/fstab` ne contenant pas l'option `noauto` seront montés automatiquement lors du boot : `/` est monté en premier, puis le swap, les autres systèmes de fichiers et enfin les systèmes de fichiers virtuels (`/proc`, `/sys`, etc...).

Un périphérique déclaré dans `/etc/fstab` pourra être monté manuellement avec `mount` en lui passant seulement l'identifiant de périphérique utilisé dans le fichier ou le point de montage.

La commande `mount -a` permet de monter tous les systèmes de fichiers déclarés dans `/etc/fstab`.

II – GRUB :

1 – Définition :

GRUB (GRand Unified Bootloader) est un chargeur de système. Il est exécuté après la phase de démarrage matériel de la machine (BIOS ou UEFI) et va permettre à l'utilisateur de choisir (le cas échéant) puis de lancer l'amorce du système d'exploitation.

Il peut gérer de nombreux systèmes d'exploitation : il est capable de charger un autre chargeur de démarrage pour les systèmes qu'il ne gère pas en natif (par exemple pour Windows). Il peut même charger par le réseau des images de systèmes d'exploitation et démarrer dessus.

Il est compatible avec de nombreux systèmes de fichiers.

La version actuelle de GRUB est GRUB2.

2 – Fonctionnement :

GRUB est d'abord chargé en mémoire à partir du secteur de démarrage du disque dur.

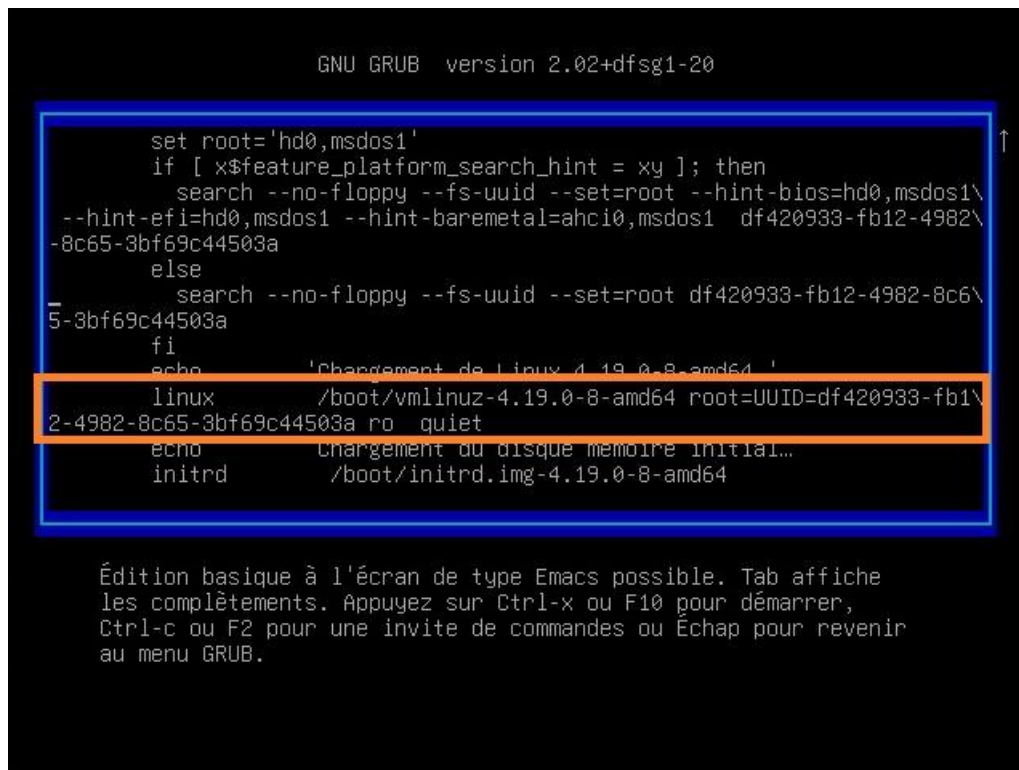
Ensuite, il affiche un menu de sélection du système à charger (en mode graphique ou texte selon la nature de la distribution qui l'a installé).

Une fois le système choisi ou le temps alloué à la sélection écoulé, soit GRUB laisse la main à un autre chargeur pour les systèmes qu'il ne peut gérer (Windows), soit il charge le noyau du système demandé et lui laisse la main (Linux et dérivés).

3 – Passage d'options au noyau :

Il est possible de passer au noyau Linux diverses options lors de la phase de démarrage via GRUB.

Lorsque GRUB propose le choix parmi les différents systèmes qu'il est possible de charger, il faut se mettre sur le système concerné et taper "e". S'ouvre alors le fichier de configuration de GRUB pour le chargement du système concerné :



```
GNU GRUB version 2.02+dfsg1-20

set root='hd0,msdos1'
if [ x$feature_platform_search_hint = xy ]; then
  search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1\
--hint-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 df420933-fb12-4982\
-8c65-3bf69c44503a
else
  _
  search --no-floppy --fs-uuid --set=root df420933-fb12-4982-8c6\
5-3bf69c44503a
fi
echo          'Chargement de Linux 4.19.0-8-amd64 '
linux        /boot/vmlinuz-4.19.0-8-amd64 root=UUID=df420933-fb1\
2-4982-8c65-3bf69c44503a ro quiet
echo          Chargement du disque memoire initial...
initrd       /boot/initrd.img-4.19.0-8-amd64

Édition basique à l'écran de type Emacs possible. Tab affiche
les compléments. Appuyez sur Ctrl-x ou F10 pour démarrer,
Ctrl-c ou F2 pour une invite de commandes ou Échap pour revenir
au menu GRUB.
```

Dans ce fichier, la commande "linux" permet de charger le noyau. On définit les options après la déclaration du noyau à charger (qui est un fichier de /boot dont le nom commence par vmlinuz).

Sur l'exemple, on voit qu'il y a 3 options déclarées :

```
root=UUID=66642ac1-3194-4ff2-87d5-c3d854238880
ro
quiet
```

root indique quel est le disque qu'il faudra monter en racine de l'arborescence (/). Ici, c'est l'UUID du disque qui est utilisé pour le dénommer.

ro indique que le disque sera monté en lecture seule (Read Only).

quiet indique que le démarrage se fera en mode silencieux.

4 – Quelques options de démarrage :

Les options sont souvent de la forme :

nom=valeur1, valeur2, ...

Où *nom* est le nom de l'option et *valeur1, valeur2*, etc... sont les valeurs éventuelles qu'on leur donne.

- *root=périphérique* : permet de spécifier quel périphérique sera monté en tant que système de fichier racine par le noyau. Le périphérique peut être donné sous diverses formes : leur dénomination (par exemple, */dev/sda1*), leur label (par exemple, *LABEL="donnees"*) ou leur identifiant unique (l'UUID, voir exemple dans la capture d'écran ci-dessus).
- *init=executable* : permet de préciser quelle sera la commande initiale exécutée par le noyau. Par défaut, c'est */sbin/init* (qui est un lien symbolique vers */lib/systemd/systemd* pour le système d'initialisation *systemd*). On peut, par exemple, mettre un shell en exécutable (*/bin/bash*) pour accéder au système en super-utilisateur sans mot de passe et ainsi récupérer un mot de passe perdu (voir TP)
- *single* : permet de démarrer en mode mono-utilisateur (le root) en chargeant le minimum de services.
- *ro* : le système de fichier racine sera monté en lecture lors de la phase de démarrage (pour permettre la vérification avec le programme *fsck*). C'est le mode conseillé.
- *rw* : le système de fichier racine sera monté en lecture et écriture lors de la phase de démarrage (ne pas utiliser le programme *fsck* dans ce cas).
- *quiet* : le minimum de messages sera affiché pendant la phase de chargement du noyau.

5 - Configuration de GRUB2 :

La configuration de GRUB2 se trouve dans `/boot/grub/grub.cfg`, fichier généré à partir de `/etc/default/grub` et des scripts contenus dans `/etc/grub.d`

On ne modifie jamais directement `/boot/grub/grub.cfg` : on exécute `update-grub2` après toute modification de `/etc/default/grub` ou des scripts de `/etc/grub.d`.

Les scripts de `/etc/grub.d` sont numérotés pour être exécutés dans un ordre précis :

- `00_header` : génère l'en-tête de `grub.cfg`
- `05_debian_theme` : génère le thème graphique de grub (pour une distribution avec interface graphique)
- `10_linux` : génère les entrées de grub relatives au système Linux
- `20_memtest86+` : génère les entrées memtest pour les tests mémoire
- `30_os-prober` : détecte les OS présents sur la machine et génère les entrées correspondantes
- `40_custom` : permet d'introduire ses propres entrées

Dans `/etc/default/grub`, on définit l'aspect du menu de démarrage :

- `GRUB_DEFAULT=val` : indique quel OS sera chargé par défaut (0 : première entrée de la liste)
- `GRUB_TIMEOUT=val` : temps en secondes avant que le système par défaut ne soit chargé si l'utilisateur n'intervient pas.

6 – Réparation de GRUB :

Il peut arriver que GRUB ne fonctionne plus et donc ne donne plus accès au système d'exploitation. Cela peut arriver après une mauvaise configuration de grub.cfg, après l'installation d'un OS qui a écrasé GRUB (Windows par exemple) ou tout simplement si on s'est trompé lors de l'installation de Linux et qu'on n'a pas demandé l'installation de GRUB.

a – GRUB écrasé ou manquant :

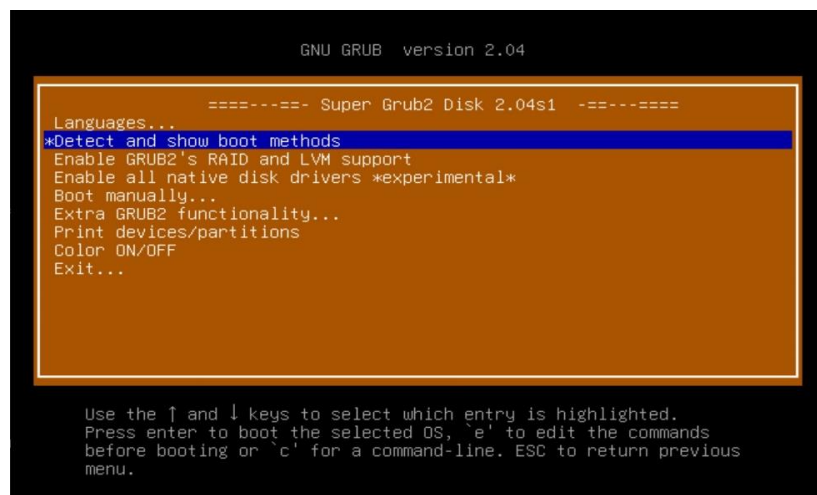
Cela se manifeste par un démarrage direct de Windows ou par un écran noir au démarrage, sans aucune invite de commande.

Dans ces cas-là, il faut accéder au système pour (ré)installer GRUB avec la commande grub-install.

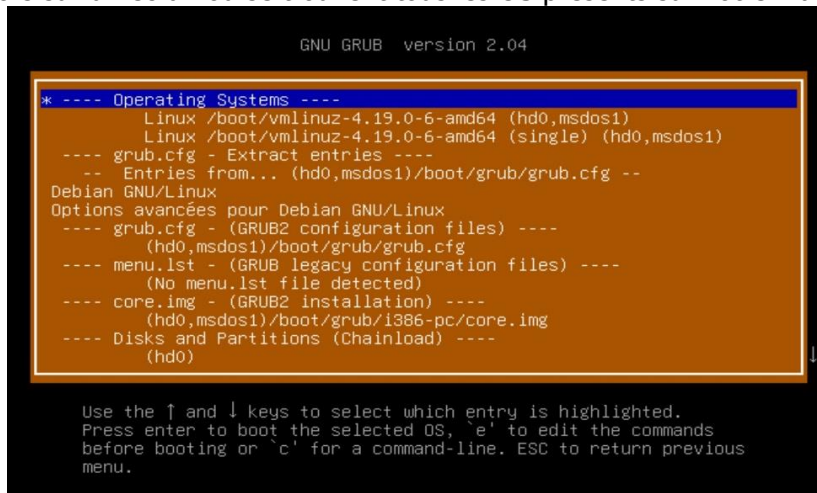
La façon la plus simple pour cela est d'utiliser un périphérique amorceable qui contient une distribution dédiée à ce genre de problèmes, telle que SuperGRUB.

NB : Pour une machine virtuelle, il suffit de mettre l'ISO de Supergrub dans le lecteur optique virtuel et de démarrer. Supergrub va alors se charger.

Une fois SuperGRUB installé sur le périphérique amorceable, il faut démarrer la machine dessus (via le BIOS ou l'UEFI), et sélectionner la détection de systèmes d'exploitation :



Vous arrivez alors sur un écran où se trouvent tous les OS présents sur votre machine :



Il faut alors choisir le système Linux et le lancer.
Si tout se passe bien, il est alors possible de se connecter au système.

Il faut maintenant (ré)installer GRUB sur le disque système.
Pour cela, il faut repérer l'identifiant du disque /dev/sdX où est montée la partition racine.

On peut alors exécuter :

```
root@debian:~#grub-install /dev/sdX
```

Si tout se passe bien on met GRUB à jour avec la commande update-grub2.

On redémarre alors la machine avec la commande reboot. GRUB doit s'afficher.

NB : Pour résoudre ce genre de problème, il est donc fortement conseillé à un administrateur Linux de toujours posséder un périphérique amorceable contenant une distribution Linux (Live-CD ou Live-USB)

b – Erreur dans GRUB :

Il se peut que le fichier grub.cfg soit mal configuré et provoque une erreur. On se retrouve alors avec une invite de commande grub> :

```
GNU GRUB version 2.02+dfsg1-20

Minimal BASH-like line editing is supported. For the first word, TAB
lists possible command completions. Anywhere else TAB lists possible
device or file completions.

grub>
```

Il va falloir charger et lancer le système à la main afin de récupérer un terminal et de pouvoir réparer GRUB.

NB : Attention : le clavier est en configuration QWERTY :

Pour un clavier de PC AZERTY :

Le caractère **(** s'obtient avec la combinaison de touches **MAJ+9**

Le caractère **)** s'obtient avec la combinaison de touches **MAJ+0**

Le caractère **m** s'obtient avec la touche **,**

Le caractère **,** s'obtient avec la touche **;**

Le caractère **=** s'obtient avec la touche **=**

Le caractère **/** s'obtient avec la touche **!**

Le caractère **z** s'obtient avec la touche **w**

Le caractère **w** s'obtient avec la touche **z**

Le caractère **a** s'obtient avec la touche **q**

Le caractère **q** s'obtient avec la touche **a**

Les chiffres s'obtiennent directement, sans appuyer sur MAJ.

La commande 'ls' permet d'afficher les partitions du disque. Elles s'affichent sous la forme (hdX,msdosY) où X représente le numéro du disque et Y celui de la partition.

Il faut alors chercher la partition contenant le système d'exploitation. Pour la reconnaître, elle contient les répertoires de la FHS linux.

```
grub>ls (hdX,msdosY) / (ne pas oublier le / pour avoir le contenu de la partition)
```

On retient les valeurs de X et Y trouvées.

On va charger le noyau Linux à la main avec la commande linux en lui passant le nom du fichier noyau et l'identifiant du disque racine :

```
grub>linux /nom_du_fichier_du_noyau root=/dev/sdZT
```

où

- le nom du fichier noyau commence par vmlinuz (utiliser la tab-completion pour le nom exact)
- la valeur de Z dépend de la valeur de X (si X=0 alors Z=a, si X=1 alors Z=b, etc...) et T prend la valeur de Y.

On va ensuite charger une image minimale du système d'exploitation, sous la forme d'un disque RAM (en mémoire vive), qui permettra de charger le système de fichiers principal par la suite :

```
grub>initrd /nom_du_fichier_du_ramdisk
```

où le nom du ramdisk commence par initrd (utiliser la tab-completion pour obtenir le nom complet).

Le système est alors prêt à démarrer. Il suffit d'exécuter la commande 'boot'.

Une fois connecté au système, on doit réinstaller GRUB proprement (voir a-) ou corriger les erreurs dans grub.cfg.

III – Les quotas disques

1 - Définition :

Ils permettent de fixer des limites par utilisateur ou par groupe sur le nombre de fichiers (limite sur le nombre d'inodes) ou sur la taille du disque utilisée (limite sur le nombre de blocs).

On implémente les quotas par système de fichiers.

Il existe, pour chaque type de quotas (groupe, utilisateur, inodes, blocs) deux limites :

- Limite dure : la quantité de blocs ou d'inodes définis par cette limite ne pourra jamais être dépassée. Quand elle est atteinte, plus aucune création de fichiers ou augmentation de taille de fichier possible.
- Limite douce : elle pourra être temporairement dépassée. On définit alors un **délai de grâce** laissant le temps à l'utilisateur/groupe de revenir sous la limite douce. Au-delà de ce temps la limite douce devient dure.

2 - Mise en place :

Les paquets de gestion de quotas doivent être installés :
apt-get install quota quotatool

Il faut ensuite indiquer que les quotas seront activés lors du montage du système de fichiers où ils seront mis en place.

Dans /etc/fstab, il faut ajouter sur la ligne du système de fichiers concerné, dans les options :

- usrquota pour activer les quotas par utilisateurs.
- grpquota pour activer les quotas par groupes.

On remonte le système de fichiers pour prendre en compte les nouvelles options :

mount -o remount point_de_montage

On crée ensuite les fichiers de quota à la racine du système de fichier concerné :

aquota.user et aquota.group

avec la commande :

```
quotacheck -cu dénomination_du_FS (pour aquota.user)
quotacheck -cg dénomination_du_FS (pour aquota.group)
quotacheck -cug dénomination_du_FS (pour aquota.user et aquota.group)
```


3 – Gestion des quotas :

On édite les quotas d'un utilisateur avec la commande `edquota` :
edquota login_utilisateur

NB : Par défaut, l'éditeur de texte est `vi`, on peut le changer avec la commande :
`update-alternatives -config editor`

On édite alors un fichier texte organisé en 7 colonnes :

Système de fichiers – Nombre de blocs utilisés - Limite douce sur les blocs - Limite dure sur les blocs - Nombre d'inodes utilisés - Limite douce sur les inodes - Limite dure sur les inodes

Une limite à 0 n'est pas activée.

On active/désactive les quotas avec **quotaon/quotaoff** suivi du nom du système de fichiers concerné ou de `-a` pour tous les systèmes de fichiers.

repquota point_de_montage permet d'afficher les quotas en cours sur le point de montage.

quota -v login : Affiche les quotas de l'utilisateur donné.

quotacheck -c point_de_montage : met à jour les fichiers de quotas (`quotaoff` conseillé avant)

4 – Délai de grâce :

On modifie le délai de grâce avec **edquota -t**.
On édite alors un fichier avec 3 colonnes :

Système de fichiers - Délai de grâce pour les blocs - Délai de grâce pour les inodes

Par défaut, les deux sont à une semaine : 7days.
Les unités de temps days, hours, minutes et seconds sont acceptées.

IV - Commandes de gestion des disques :

1 – Informations sur les périphériques :

blkid : affiche ou recherche les identifiants des périphériques blocs connectés au système (qu'ils soient montés ou non) :

- la dénomination système
- l'UUID
- le label s'il existe
- le type du système de fichiers

blkid permet aussi de rechercher des périphériques répondant à certaines caractéristiques avec l'option **-t** *NOM=valeur*.

Exemple :

```
root@debian7:~# blkid
/dev/sdb1: UUID="a3ed2fcf-8420-4c9e-94e4-3c5471ba5dd0" TYPE="ext4" PARTUUID="523ed54e-01"
/dev/sda1: UUID="8e3888f6-8b02-483e-ab3f-9ef66f34fbd8" TYPE="ext4" PARTUUID="8d9b7322-01"
/dev/sda5: UUID="53b58954-2957-4918-a065-91086432c194" TYPE="swap" PARTUUID="8d9b7322-05"
/dev/sr0: UUID="2016-01-19-16-46-09-00" LABEL="VBOXADDITIONS_5.0.14_105127" TYPE="iso9660"
root@debian7:~# blkid -t TYPE=ext4
/dev/sdb1: UUID="a3ed2fcf-8420-4c9e-94e4-3c5471ba5dd0" TYPE="ext4" PARTUUID="523ed54e-01"
/dev/sda1: UUID="8e3888f6-8b02-483e-ab3f-9ef66f34fbd8" TYPE="ext4" PARTUUID="8d9b7322-01"
root@debian7:~#
```

2 - Occupation du disque :

df : donne des informations sur chaque système de fichiers monté :

- Dénomination
- Nombre total de blocs de 1ko
- Nombre de blocs de 1ko occupés
- Nombre de blocs de 1ko disponibles
- Capacité (pourcentage d'occupation)
- Point de montage

L'option **-h** (human-readable) affiche les résultats dans l'unité la plus lisible.

du : donne des informations sur l'espace occupé par une arborescence.
Par défaut, affiche les informations sur le répertoire courant.

options :

- h : affichage en unité lisible.
- s : n'affiche que la taille totale de l'arborescence demandée.

3 – Maintenance :

fsck (File System Consistency check) : permet de vérifier la cohérence d'un système de fichiers.

Syntaxe : fsck -t typeFS périphérique

Le système de fichiers à vérifier ne doit pas être monté, ou en lecture seule.

C'est une commande interactive : en cas de problème, elle demandera à l'utilisateur quoi faire (option -p pour lancer une réparation automatique).

fsck est lancé automatiquement au démarrage du système, si le système de fichiers n'a pas été vérifié depuis un certain temps ou depuis un certain nombre de montages.

tune2fs : permet de modifier certaines options des systèmes de fichiers ext.

Syntaxe : tune2fs option(s) périphérique

options : (vérification au boot seulement)

-c X : nombre de montages après lesquels le système de fichiers sera automatiquement vérifié.

-i X : temps après lequel le système de fichiers sera automatiquement vérifié (d : jours ; w : semaines ; m : mois)

-L label : change le label du disque.

V - Droits d'accès étendus :

1 - SUID et SGID :

Ces droits permettent de lancer un exécutable en endossant les droits du propriétaire du fichier (SUID) ou du groupe auquel appartient le fichier (SGID).

Exemple :

La commande `passwd` modifie les fichiers d'utilisateurs normalement modifiables seulement par le root. Elle possède donc le droit SUID donnant les droits root sur cette commande à ceux qui l'utilisent.

Ce droit se matérialise par un `s` à la place du `x` dans les droits user (SUID) et dans les droits group (SGID) :

```
-rwsr-xr-x 1 root root 45396 mai 25 2012 passwd
```

Ce droit `s` se donne en l'ajoutant à `u` (SUID) ou à `g` (SGID)
En numérique, les valeurs à ajouter sont 4000 pour le SUID et 2000 pour le SGID

Exemple :

`chmod 4755 fic`
donnera les droits `rwsr-xr-x` à `fic`

Si le droit d'exécution n'est pas accordé, mais le SUID ou le SGID oui, ce qui est totalement illogique, le SUID et le SGID concernant l'exécution, on aura un `S` à la place du `s`.

Exemple :

`chmod 6744 fic`
donnera les droits `rwsr-Sr--` à `fic`

(`S` majuscule car le SGID est activé, mais il n'y a pas de droit en exécution pour le groupe)

2 - Le sticky bit :

Il permet de mettre en place une protection contre l'effacement de fichier dans un répertoire.

Il est notamment utile dans les répertoires publics où tout le monde a le droit de lire et d'écrire. Comme c'est le droit en écriture du répertoire où il se trouve qui permet d'effacer un fichier, tout le monde peut supprimer tous les fichiers.

Le sticky bit, appliqué au répertoire, permet de supprimer la possibilité par un utilisateur d'effacer un fichier qui ne lui appartient pas. Il pourra toujours le lire et le modifier.

La valeur numérique du sticky bit est 1000 et son symbole est le `t`.

Exemple :

```
root@debian7:/usr/bin# chmod +t /public
root@debian7:/usr/bin# ls -dl /public
drwxr-xr-t 2 root root 4096 févr.  2 22:03 /public
root@debian7:/usr/bin#
```

3 - Changement de groupe/propriétaire :

Le super-utilisateur peut changer le propriétaire d'un fichier :

chown nouvel_utilisateur nom_fichier

Un utilisateur peut changer le groupe d'un fichier s'il fait partie du nouveau groupe et que le fichier lui appartient :

chgrp nouveau_groupe nom_fichier

Pour les 2 commandes, l'option `-R` effectue le changement de manière récursive

NB : `chown` peut modifier le propriétaire et le groupe simultanément:

chown nouvel_utilisateur : nouveau_groupe nom_fichier

VI - Localisation de fichiers :

1 - L'arborescence Linux :

Elle respecte le FHS (File Hierarchy Standard).

Les répertoires du système sont :

- /bin : exécutable des commandes systèmes
- /home : répertoires personnels des utilisateurs
- /init.d : procédures à lancer lors du démarrage
- /boot : fichiers de démarrage
- /etc : fichiers de configuration
- /root : répertoire personnel du superutilisateur
- /sys et /proc : images des processus en cours d'exécution
- /dev : points d'entrée des périphériques
- /lib : bibliothèques partagées du système
- /sbin : exécutable des commandes du système
- /usr : sous-arborescence utilisée par les applications utilisateurs
- /lost+found : contient les fichiers récupérés après un crash

2 - Recherche de fichiers :

a – Recherche statique :

Syntaxe : *locate motif*

La commande locate se base sur une base de données contenant l'emplacement de tous les fichiers du système. Elle est donc très rapide mais ne prend pas en compte les modifications depuis la dernière mise à jour de la base de données.

Cette base est mise à jour avec la commande updatedb.

Le fichier /etc/updatedb.conf permet de configurer updatedb, et notamment exclure des systèmes de fichiers ou des répertoires de la base.

b – Recherche dynamique :

Syntaxe : *find point_de_départ critère(s)_de_recherche action*

L'action par défaut est -print, qui affiche les résultats à l'écran

Critères de recherche :

Nom de fichier :

- name nom : recherche par nom de fichier, on peut utiliser les méta-caractères du shell (protégés par des guillemets ou des quotes), sensible à la casse.
- iname : idem -name mais non sensible à la casse.

Exemple :

```
user1@debian7:~$ find . -name "*.c"
./fopen.c
./fwrite.c
./signal_sigaction.c
./fread.c
./wait_multiple.c
./prog_reseau/serveur_concurrent_bloque.c
./prog_reseau/serveur_concurrent.c
./prog_reseau/serveur_simple_bloque.c
./signal_mapping.c
user1@debian7:~$
```

Taille de fichier :

- size +nb: retourne les fichiers dont la taille est supérieure à nb blocs.
- size -nb: retourne les fichiers dont la taille est inférieure à nb blocs.
- size nb: retourne les fichiers dont la taille est égale à nb blocs.

On peut préciser une taille en multiples d'octets en faisant suivre nb de :

- b pour des octets.
- k pour des kilo-octets.
- M pour des méga-octets.

Exemple :

```
user1@debian7:~$ find . -size +1M -print
./cache/tracker/meta.db-wal
./cache/tracker/meta.db
user1@debian7:~$
```

- empty : utilisé pour trouver les fichiers vides

Date de modification :

-mtime nb : spécifie le nombre de périodes de 24h
-daystart : le temps sera mesuré à partir de 0h00.

Exemples :

find /var/log -mtime 1
Retourne la liste des fichiers de /var/log qui ont été modifiés il y a exactement 24h

find /var/log -mtime -1
Retourne la liste des fichiers de /var/log qui ont été modifiés il y a moins de 24h

find /var/log -mtime +1
Retourne la liste des fichiers de /var/log qui ont été modifiés il y a plus de 24h

find /var/log -mtime 1 -daystart
Retourne la liste des fichiers de /var/log qui ont été modifiés hier.

Les critères -atime et -ctime fonctionnent comme mtime, mais concernent la date d'accès et la date de création.

On a aussi les critères -mmin, -amin et -cmin qui fonctionnent de la même façon, mais avec les minutes.

Utilisateur :

-user nom_utilisateur : recherche les fichiers appartenant à nom_utilisateur.
-group nom_groupe : recherche les fichiers du groupe nom_groupe.

Multi-critères :

On peut enchaîner les critères de recherche sur la ligne de commande :

find / -user esgi -size +2M

Exécution de commande :

Il est possible d'exécuter une commande sur les fichiers trouvés avec la directive -exec

-exec commande ';'

On peut passer à la commande le nom des fichiers trouvés via la chaîne '{ }'

Exemple :

find /home -size +100M -exec rm '{ }' ';'

c – Recherches sur les commandes :

- *which nomcommande*

Indique où se situe dans l'arborescence la commande passé en paramètre.

Cela permet notamment de savoir, pour les cas où plusieurs exécutables de même nom situés dans des répertoires différents existent, lequel sera exécuté par défaut.

- *whereis nomcommande*

Retourne la localisation de l'exécutable, des pages de manuel et des éventuelles sources de la commande passée en argument