Système de fichiers

Un système de fichiers est une façon d'organiser et de stocker une arborescence sur un support (disque, cd, ...). Chaque OS a sa propre organisation.
Linux possède son système appelé ext2 mais peut en gérer d'autres. La liste se trouve en général dans /proc/filesystems. L'utilisateur peut donc accéder sous Linux à d'autres systèmes de fichiers (DOS, Vfat, NTFS, ...) provenant d'un périphérique ou importé par le réseau.

Tout est fichier

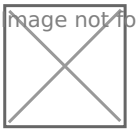
Comme nous l'avons vu précédemment, dans le système linux, tout est considéré comme un fichier :
- Les fichiers
- Les dossiers
- Un disque dur
- La mémoire
- Un port USB aussi .

Mais parmi ces fichiers, tous n'ont pas la même catégorie. On peut en effet retrouver :
- fichiers normaux (texte, exécutables, ...); symbole "-"
- fichiers répertoires, ce sont des fichiers conteneurs qui contiennent des références à d'autres fichiers. Véritable charpente de
l'arborescence.; symbole "d"
- fichiers spéciaux, ils sont situés dans /dev, ce sont les points d'accès préparés par le système aux périphériques. Le montage va
réaliser une correspondance de ces fichiers spéciaux vers leur répertoire "point de montage". Par exemple, le fichier /dev/hda permet l'accès et le chargement du 1er disque IDE.
- Accès caractère par caractère; symbole "c"
- Dispositif de communication ; symbole "p"
- Accès par bloc ; symbole "b"
- fichier lien symboliques, ce sont des fichiers qui ne contiennent qu'une référence (un pointeur) à un autre fichier. Cela permet d'utiliser un même fichier sous plusieurs noms sans avoir à le dupliquer sur le disque. ; symbole "l"

Le processus de montage que l'on évoque pour les fichiers spéciaux, avec sa commande mount est le moyen de faire correspondre les parties de l'arborescence et les partitions physiques de périphérique. Il permet de plus d'affecter tout système extérieur (cd, zip, réseau) à un répertoire créé pour cela dans l'arborescence. Il suffira ensuite de se déplacer à ce répertoire, appelé point de montage, qui est en fait un répertoire "d'accrochage", pour accéder à ses fichiers (bien sûr, conformément aux permissions que possède l'utilisateur).

FHS (Filesystem hierarchy standard)

C’est la standardisation des noms et des emplacements des dossiers, et de leurs contenus.



Voici la structure qui est recommandée d’après les spécifications

SI l’on détaille :
- /bin : Les fichiers binaires les plus importants
- /boot : le Kernel et le bootloader
- /dev : les périphériques
- /etc : les fichiers de configurations spécifiques à l’ordinateur
- /home : Emplacement contenant les profils dossiers utilisateur
- /mnt : Emplacement d’accès aux systèmes de fichiers montés dans l’OS
- /lib : Bibliothèques de fonctions utilisées par les commandes /bin et /sbin
- /proc : Le kernel place ici l’état en temps réel du fonctionnement du système et des processus
- /root : Dossier du profil “root”
- /sbin : Contient des exécutables
- /tmp : à utiliser pour la création de fichiers temporaires
- /usr : Applications installées par les utilisateurs (et souvent les données qui vont avec) souvent un des plus lourds!
- /var/log : Fichiers logs
- /var/spool : File d’attente d’impression
- /var/tmp : Fichiers temporaires

Gob et Jokers

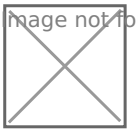
Le file globing, est l’opération qui transforme un caractère joker en une liste de chemins, et de fichiers correspondants qui formeront le résultat.
Le file globing est l’ancêtre des expressions régulières. Sous Bash, les jokers possibles sont :
- Le caractère *
- Le caractère ?
- Les caractères []
Le caractère * est utilisé pour obtenir la liste de tout le contenu
Ex : ls => ls *

Il est possible de préciser plus de caractéristiques :
ls D* => Tous les fichiers commençant par un D

ls d*v* => Tous les fichiers commençant par un d minuscule mais contenant aussi la lettre v
Le caractère ?, est utilisé pour préciser que le résultat doit contenir un seul ou plusieurs caractères (un par joker ?)
Exemple :
ls ? => liste tout ce qui ne contient qu'un seul caractère
ls ?? => liste tous les fichiers qui n'ont que 2 caractères
ls w?? => liste tous les fichiers qui ont 3 caractères et commençant par w
ls w??* => liste tous les fichiers qui commencent par w et contiennent au moins 3 caractères.
Le joker [], est utilisé pour spécifier une plage de caractères admissibles.
ls [a-f]?? => liste tous les fichiers dont la première lettre commence par a,b,c,d,e ou f et faisant 3 caractères.
ls [w]* => liste tous les fichiers commençant par w
ls [*] => ici l'étoile est son propre caractère, liste tous les fichiers commençant par *
ls *[0-9][0-9]* => liste tous les fichiers contenant 2 chiffres
ls [!a-y]* => ne liste que les fichiers commençant par autre chose que les lettres a à y. le ! inverse le résultat
ls *[^3-9] => liste tous les fichiers excluant ceux contenant les chiffres 3 à 9

Droit d'accès et permissions

Droits
Un fichier est caractérisé par un certain nombre de droits d'utilisation. Lorsque que l'on fait un ls -l, on remarque les éléments suivants :



le premiers bloc est définie comme suit :
_ Type de fichier (d, -, p, l, ...)
_ _ _ Droits du propriétaire du fichier (owner), dans l'ordre (lecture, écriture, exécution)
_ _ _ Droits du groupe auquel appartient le fichier (group), dans l'ordre (lecture, écriture, exécution)
_ _ _ Droits d'un utilisateur quelconque (user), dans l'ordre (lecture, écriture, exécution)
Pour rappel :

Permission	Effet sur le fichier	Effet sur le dossier
r (read)	Autorise la lecture ou la copie du fichier et son contenu	Sans droit d'exécution : autorisation de liste les fichiers dans le dossier Avec droit d'exécution : autorisation de lister les fichiers de façon détaillée dans le dossier
w (write)	Autorise à supprimer ou modifier le contenu du fichier. Permet de supprimer le fichier	Sur un dossier, il faut également l'attribut "execute" afin d'opérer des modifications

x (execute)	Autorise un fichier d'être lancé comme un processus	Autorise un utilisateur à entrer dans ce dossier

L'attribut suivant (1 par exemple) est le nombre de lien symbolique vers ce fichier
Les 2 attributs d'après : root root définissent le propriétaire suivis par le groupe
L'utilisateur qui crée le fichier est considéré comme son propriétaire. Il faut savoir que seul root est habilité à modifier le propriétaire d'un fichier, avec la commande "chown"
Le groupe du fichier sera celui du groupe principal auquel appartient l'utilisateur qui crée ce fichier. la commande "id" indique l'identité de l'utilisateur actuel, son group principal, et tous les groupes auxquels il appartient. Pour changer le groupe d'un fichier, on utilise la commande "chgrp" seuls root et le propriétaire du fichier peuvent changer le groupe d'un fichier.

Permissions

Seul root et le propriétaire du fichier disposent de la possibilité de modifier les permissions d'un fichier au moyen de la commande chmod
Exemples :
Ajouter les droits d'exécution au propriétaire du fichier
`chmod u+x monfichier`
Supprimer les droits d'écriture pour le groupe :
`chmod g-w monfichier`
Assigner des droits à plusieurs sections des attributs ;
`chmod o=r,g-w,u+x monfichier`
Supprimer toutes les permissions
`chmod a=- monfichier`

La méthode octale reste la plus rapide pour l'attribution, avec
r=4; w=2 et x=1
Pour donner les permissions rwx r-x r-x à un fichier, on va additionner les valeurs :
Owner: 4+2+1 => 7
Group : 4+1 => 5
Other : 4+1 => 5
=> `chmod 755 monfichier`

Attribut spéciaux -setuid
Cet attribut spécial permet d'appliquer à un fichier exécutable, d'autoriser d'autres utilisateurs à lancer l'exécutable, comme s'il étaient l'utilisateur root.
`chmod 4000`
Exemple avec la commande passwd :
s -> permissions x+setuids (s)
S -> seule la permission setuid existe (pasx)

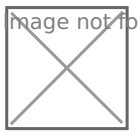
Image not found or type unknown

Attribut spéciaux -setgid
Même chose qu’avec setuid, mais pour le groupe utilisateur
`chmod 2000`
- Appliqué à un fichier exécutable, afin de s’exécuter via le group du propriétaire, au lieu de celui de l'utilisateur qui lance l’exécutable
- Appliqué à un dossier, fait en sorte que le contenu créé dans ce dossier appartient au groupe qui possède le dossier (et pas celui de l’utilisateur qui crée le contenu dans ce dossier)

Attribut spéciaux -sticky
chmod 1000
L’utilité de cet attribut est d’empêcher d’autres utilisateurs de supprimer le contenu d’un autre utilisateur.
Pour supprimer un fichier, il faut les autorisations d’écriture sur le dossier parent. Ainsi, un admin crée un dossier accessible et modifiable par tous les utilisateurs. Tout le monde pourra supprimer le contenu de ce dossier ...
L’utilisation de cette permission sticky permet de définir pour un dossier que :
- L’utilisateur propriétaire d’un fichier, dans ce dossier, pourra supprimer le fichier
- root et l’utilisateur propriétaire du dossier parent pourra aussi

Héritage vs umask

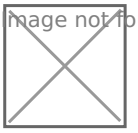
Lorsque l’on crée un sous-dossier sous windows, ce dernier hérite des droits d’accès du dossier parent (par défaut). Il faut savoir que sous linux : NON ! ce n’est pas le cas! En effet le sous-dossier aura les droits créés selon la valeur umask de l’utilisateur créant le sous dossier.



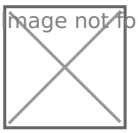
Read vaut 4, write 2, execute 1, rien vaut 0.
Dans le cas de umask, ces valeurs précisent les permissions à supprimer des permissions “maximum” à la création !
Sur un fichier, les permissions maximales sont : rw- rw- rw- soit 666 en octal. La valeur umask de l’utilisateur créant le fichier va donc
permettre de supprimer certaines de ces valeurs! Cette valeur se code avec trois nombres de base en octale.
Ex, Si l’on souhaite que par défaut l’utilisateur sisr, crée des fichiers avec les permissions rw- r-- --- :
- Le premier nombre octal sera 0 (on ne change rien à rw-)
- Le second nombre (groupe) sera 2 (on supprime la permission w, donc 2, il reste r--)
- Le troisième nombre (others) sera 6 (on supprime r soit 4 et w soit 2, il reste ---)
Le umask de l’utilisateur sisr doit donc être 026

Sur un dossier, sans droits d’exécution, il est impossible d’agir dans le dossier, et les éventuels droits en écriture dans ce dossier ne seront pas appliqués.
De ce fait, les permissions “maximales” par défaut accordées à la création d’un dossier sont : rwx rwx rwx soit 777 en octal
Exemple: Le propriétaire doit disposer des droits complets sur le dossier, seulement des droits d’exécution et lecture pour le groupe, et aucun droits pour les autres. Ceci se traduit par : rwx r-x ---
Le umask de l’utilisateur devra retirer les droits en écriture sur le groupe (-2) et les droit rwx à others (-7). Le umask sera donc 027

Umask propose un comportement différent sur les fichiers et sur les dossiers. Mais cette valeur umask est la même pour les fichiers et les dossiers ! On utilise des valeurs “classiques” pour régler umask



Si l’on fait :



Jusqu’ici, nous avons précisé des valeurs sur trois chiffres en base octale. La commande umask affiche le umask de l’utilisateur actuel, mais sur 4 chiffres !
Le premier représente en faite les attributs spéciaux (setuid, setgid, sticky). Cependant, ces attributs spéciaux ne sont pas spécifiables par défaut (lors de la création d’un fichier ou dossier). Il n’est donc pas nécessaire de la préciser ou la calculer lors de l’utilisation de umask.
/!\ Umask n’est utilisé qu’à la création, les fichier existant ne sont pas touchés!

Gestion des archives

Une archive est un fichier contenant un ou plusieurs fichiers ou dossiers, la plupart du temps “compressés” ce afin de diminuer la
consommation d’espace disque. Il existe de nombreux algorithmes de compression et d’archivage, vous connaissez probablement le zip et le rar connu sous windows. Voyons certains des plus utilisés sous les système linux :

Tar

Tar est une très ancienne commande (Tape Archive), utilisée autrefois pour la sauvegarde de contenu sur des bandes magnétiques. Les
premières versions permettaient uniquement l’archivage, mais depuis l'arrivée de gzip (-z) et bzip2(-j) c’est maintenant possible.
Principaux attributs de tar (extension de fichier .tar en général):

- tar -c : Créer une archive
- tar -cf : Créer une archive et spécifier son nom
- tar -cvf : Créer une archive et afficher un diagnostic à l'écran
- tar -cvzf : Créer une archive, compresser avec gzip et afficher le diagnostic
- tar -tf : Lister (option t) le contenu de l'archive en argument (option f)
- tar -vtf : Lister (t) de façon détaillée (v) l'archive (f)
- tar -xf : Extraire (option x) le contenu de l'archive (f)

Gzip et gunzip

Par défaut l'utilisation de gzip remplace un fichier par sa version compressée ! qui aura l'extension ".gz". Pour éviter ce comportement si l'on
veut conserver le fichier d'origine, on utilise l'option -c qui va rediriger le contenu extrait vers stdout !) il faudra donc faire soi-même la
redirection vers un fichier :
gzip -c monfichieraziper > monfichieraziper.gz
gzip -cr /etc > etc.gz
Gunzip quand à lui effectue l'opération inverse de gzip, il décompresse une archive gz. Et par défaut comme gzip, il remplace le fichier
d'origine ".gz" par sa version décompressé.

bzip2 et bunzip2

Le fonctionnement est similaire au gzip et gunzip, l'algorithme de compression utilisé est différent cependant. L'extension par défaut que l'on
rencontrera sera : bz2. L'option de récursion -r n'est pas disponible non plus, on utilisera le * à la place.

zip et unzip

Même si ce format est surtout utilisé sous windows, linux dispose de son équivalent :
zip ./test.zip ./test/* va compresser le contenu du dossier test dans le fichier test.zip
zip -r /etc/etc.zip /etc va compresser tout le contenu du dossier /etc

xz

xz -z * permet de compresser des fichiers individuellement (autant d'archives que de fichiers) avec l'extension .xz
xz -d permet de décompresser les fichier

cpio

cpio: copy-in copy-out
En mode copy-out (option -o), les fichiers dans un dossier seront recopiés vers une archive
En mode copy-in (option -i) le contenu d'une archive est listé ou copie les fichiers d'une archive vers un dossier.

dd

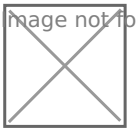
Permet d'effectuer la copie d'un fichier un d'une partition complète bit à bit. Cela permet de :
- Cloner un disque ou une partition
- Sauvegarder la MBR (master boot record)
- Création d'un fichier d'une taille précise rempli de 0

Recherche de fichier grâce aux commandes locate et find

locate et find sont les deux commandes principales pour chercher des fichiers dans la table d'inode de linux. Elles fonctionnent différemment
mais permettent le même résultat. Nous verrons aussi les commandes whereis / which / type

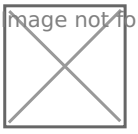
locate

Cette commande utilise une base de données mise à jour quotidiennement par l'OS. Si vous désirez mettre à jour cette base de données, vous
pouvez utiliser la commande updatedb
Les particularités de cette commande :
- N'affiche que les fichiers lisibles par l'utilisateur
- Très rapide pour la recherche, mais les fichiers récents seront absent à moins d'utiliser l'updatedb
- peut nécessité d'être installé apt-get install locate
Exemple : locate passwd



find

La commande a besoin de plusieurs choses pour travailler :
- Un nom de dossier comme point de départ de la recherche (sinon travail depuis le dossier courant)
- Pour rechercher un fichier on utilise l'option -name
- Plus lent que locate mais permet de se positionner où l'on veut et n'oubliera aucun fichier
exemple :



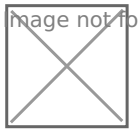
Vous remarquez qu'il y a beaucoup moins de fichiers, c'est qu'il faut utiliser le globing :
exemple : find / -name *passwd*
Voici différents options pour le find :

Options	Exemple	Description

-iname	-iname Hello	recherche les fichiers en ignorant la casse
-mtime	-mtime -7	Les fichiers modifiés il y a moins de 7 jours
-mmin	-mmin 5	Les fichiers modifiés il y a moins de 5 minutes
-size	-size +10M	Les fichiers pesant plus de 10Mo
-user	-user sisr	Les fichiers possédés par l'utilisateur sisr
-empty	-empty	liste les fichiers vides
-type	-type d	Liste les fichiers qui sont des dossiers
-maxdepth	-maxdepth 1	Profondeur de recherche limité à 1, ne parcourt pas les sous-dossiers

Si l'on veut tous les fichiers créés par l'utilisateur sisr qui s'appelle hosts :
find / -user sisr -name hosts
Ou une syntaxe plus compliquée :
find / -iname 'ifconfig*' -o \(-name hosts -user sisr \)
cette commande trouvera à partir du dossier racine :
- Tous les fichiers commençant par ifconfig
- OU Tous les fichiers intitulés hosts appartenant à l'utilisateur sisr
Pour définir sur quels attributs effectués un OU, il faut mettre les attributs entre parenthèses. Le problème c'est que le shell interprète la parenthèse comme un caractère spécial, il faut donc le protéger en utilisant \ devant chaque parenthèse pour dire au shell de ne pas l'interpréter mais de l'envoyer directement à la commande find. Le caractère \ est remplaçable par le ' find . -name host* -exec ls -l {} \;
- Trouver tous les fichiers commençant par host dans le dossier actuel et ses sous-dossiers
- Puis lancer la commande ls -l pour chaque fichier trouvé
- le {} permet de passer, fichier par fichier, la main à la commande ls
- Pour que ceci fonctionne, il faut rajouter un ; à la fin de chaque fichier passé en argument, donc \;

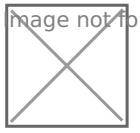
whereis et which



- whereis trouve deux binaire au nom de base
- which nous indique que seul bash est utilisé quand on utilise la commande base et non pas bash.bashrc. Ce dernier est un script
automatiquement lancé lorsqu'on se connecte à la ligne de commande en mode interactif.

type

Image not found or type unknown



Revision #1
Created 31 October 2019 13:06:07 by Cécile
Updated 31 October 2019 13:06:29 by Cécile