**COURS 04  
Opérations sur les fichiers et répertoires   
(Création, Modification, recherche, archives tar, Compression, …)**

**Partitions et systèmes de fichiers**

Une partition est une partie (morceau) logique d’un périphérique de stockage.  
Un système de fichiers est une méthode de stockage et de recherche de fichiers sur un périphérique de stockage (habituellement une partition).

**Comparaison entre les systèmes de fichiers sous Windows et Linux**

|  |  |  |
| --- | --- | --- |
|  | Windows | Linux |
| Partition | Disk1 | /dev/sda1 |
| Type de système de fichiers | NTFS, FAT32, … | ext4, btrfs, XFS, ZFS, … |
| Paramètres de montage | Lettre de lecteur | Point de montage |
| Répertoire de base | Lecteur C | / |

**Hiérarchie standard d’un système de fichiers**

Bien que chaque distribution Linux possède sa propre façon de faire les choses, la plupart des distributions respecte une série de normes pour l’arborescence (disposition des répertoires).

Toutefois, un administrateur système devrait être en mesure de facilement retrouver les principaux fichiers de configuration.  
Ces fichiers devraient être classés de manière cohérente, aux mêmes endroits, et ce sur toutes les distributions.

Un système Linux stocke ses fichiers importants selon un schéma standard appelé **Hiérarchie Standard d’un Système de Fichiers** (*Filesystem Hierarchy Standard* ou FHS).

Cette norme offre la garantie que les utilisateurs peuvent travailler avec différentes distributions sans avoir à apprendre à chaque fois comment le système de fichiers est structuré.

FHS fait une distinction importante entre les fichiers qui sont partageables et les fichiers ne le sont pas …

* Les **fichiers dits partageables** peuvent être partagés entre hôtes, tels que les fichiers de données utilisateur et les fichiers binaires (exécutables) ;
* Les **fichiers non partageables** contiennent des informations spécifiques au système, tels que les fichiers de configuration.

FHS établit également une **seconde distinction importante** entre les fichiers/répertoires statiques et les fichiers/répertoires variables …

* Les **fichiers/répertoires statiques** ne sont pas généralement modifiés, sauf par intervention directe d’un administrateur système.  
  La plupart des binaires sont des exemples de fichiers statiques ;
* Les **fichiers/répertoires variables** sont ceux qui peuvent être modifiés par les utilisateurs, les scripts automatisés, les services, ...   
  Les répertoires de base des utilisateurs et les courriels des utilisateurs sont composés de fichiers variables et sont des exemples de ce type de fichiers.

Le FHS cherche à isoler chaque répertoire dans une cellule d’une matrice 2 × 2   
(partageable / non partageable X statique / variable).

|  |  |  |
| --- | --- | --- |
|  | **Partageable** | **Non partageable** |
| **Local** | **/usr**  **/opt** | **/etc**  **/boot** |
| **À distance** | **/home**  **/var/mail** | **/var/run**  **/var/lock** |

|  |  |  |
| --- | --- | --- |
|  | **Statique** | **Dynamique** |
| **Statique** | **/etc**  **/bin**  **/sbin**  **/lib** | **/dev**  **/var**  **/log** |
| **Variable** | **/usr**  **/opt** | **/home**  **/var/mail** |

**Répertoires importants selon le FHS**

|  |  |
| --- | --- |
| Répertoire | Objet |
| / | Répertoire racine  Tous les fichiers apparaissent dans ce répertoire ou dans un de ses sous-répertoires |
| /etc | Répertoire qui contient les fichiers de configuration du système |
| /boot | Contient les fichiers de démarrage importants, tels que le noyau Linux, le disque RAM initial et normalement des fichiers de configuration du chargeur de démarrage GRUB |
| /bin | Répertoire qui contient les fichiers de programme essentiels au fonctionnement normal et aux fichiers ordinaires que les utilisateurs peuvent utiliser |
| /sbin | Répertoire qui contient les fichiers de programme critiques pour un fonctionnement normal  Les utilitaires de ce répertoire sont surtout utiles à l’administrateur système |
| /lib  /lib64 | Répertoire qui contient les bibliothèques - le code utilisé par plusieurs binaires - qui sont critiques pour le fonctionnement de base du système |
| /usr | Répertoire qui contient les binaires et les données utilisés dans le fonctionnement normal du système  Ils ne sont toutefois pas critique pour le démarrage du système |
| /home | Répertoire qui contient les données personnels des utilisateurs  La séparation de ce répertoire dans son propre système de fichiers isole efficacement la plupart des données utilisateur du système d'exploitation. Cela peut s’avérer utile en cas de réinstallation du système d'exploitation (sans toutefois perdre les données de l'utilisateur) |
| /root | Répertoire qui contient les données de l'utilisateur root |
| /var | Répertoire qui contient divers fichiers transitoires, tels que les fichiers journaux et les fichiers de la queue d'impression  Le sous-répertoire de */var*, */var/tmp*, mérite une attention particulière Comme */tmp*, */var/tmp* contient des fichiers temporaires, ces fichiers ne doivent pas être supprimé lorsque l’hôte redémarre |
| /tmp | Répertoire qui contient les fichiers temporaires, y compris les fichiers temporaires créés par les programmes utilisateur  De tels fichiers peuvent théoriquement être supprimés lorsque l’hôte redémarre (en pratique de nombreuses distributions ne le font pas) |
| /mnt | Répertoire qui contient les points de montage pour les supports amovibles  Il est parfois divisé en sous-répertoires pour chaque système de fichiers monté |
| /media | Répertoire qui est le nouveau point de montage pour les supports amovibles; typiquement séparés dans des sous-répertoires pour chaque système de fichiers monté |
| /dev | Répertoire qui contient les fichiers de périphérique, qui fournissent un accès de bas niveau au matériel |
| /run | Répertoire qui contient des informations sur le système en cours d'exécution |

Linux utilise le **caractère /** (racine ou *root*) afin de représenter le répertoire principal.   
Il n’utilise pas des lettres de lecteur comme le système d’exploitation Windows.  
Les partitions sont montées (liés) comme des répertoires dans un seul système de fichiers.   
On les retrouve habituellement sous les répertoires **/mnt** et **/media**.

Ainsi, les fichiers d’un cédérom nommé debian pourrait se trouver dans le répertoire **/media/debian** et un fichier tux.texte sur ce disque serait **/media/debian/tux.texte**.

**Remarque** ...  
Tous les noms de fichiers et de répertoires pour un système Linux sont sensibles à la casse (**MAJUSCULES**/**minuscules**) …

* Les répertoires /data, /Data et /DATA représentent trois différents répertoires (ou dossiers) ;
* Les fichiers tux.texte, Tux.texte et TUX.TEXTE représentent trois différents fichiers.

De nombreuses distributions font une distinction entre les services de base partagés nécessaires pour un fonctionnement correct du système et d'autres programmes.   
Ils placent alors ces services de base dans des répertoires sous /usr (utilisateur).

**Remarque** …  
Les commandes Linux sont également sensibles à la casse (*case sensitive*).  
La commande **uname** est en minuscule.   
Les commandes **Uname** et **UNAME** n’existent pas pour un système Linux.

**Accès aux répertoires**

À la suite d’une connexion à un système ou à une ouverture d'un terminal, le répertoire est, par défaut, le répertoire personnel de l’utilisateur qui se connecte.

Pour connaître le chemin exact de ce répertoire, il suffit d’entrer la commande ..  
**>>** **echo $HOME**

Une image contenant texte

Description générée automatiquement

Les commandes suivantes sont utiles pour se situer ou se déplacer dans l’arborescence d’un système de fichiers Linux …

|  |  |
| --- | --- |
| Commande | Résultat |
| pwd | Affichage du répertoire de travail actuel |
| cd ~ ou cd | Déplacement vers le répertoire personnel (**~** (tilde)) |
| cd .. | Déplacement vers le répertoire parent (**..**) |
| cd - | Déplacement vers le répertoire précédent (**-** (moins)) |

**Chemins absolus et relatifs**

Il existe deux façons d'identifier les chemins pour accéder à des répertoires ou des fichiers …

* **Chemin d'accès absolu**Un chemin absolu prend toujours son point de départ au répertoire racine (/ ou *root*) et suit l'arbre, branche par branche, jusqu'à ce qu'il atteigne le répertoire ou le fichier souhaité.   
  Les chemins absolus débutent donc **toujours** par le caractère **/** ;
* **Chemin d’accès relatif**Un chemin relatif trouve sa source (son point de départ) dans le répertoire de travail actuel.  
  Il est possible d’utiliser la commande **pwd** ou la commande **echo $HOME** pour se situer dans l’arborescence.   
  Les chemins relatifs ne débutent **jamais** par le caractère **/**.

Les barres obliques multiples (/) entre les répertoires et les fichiers sont autorisés.   
Toutefois, seulement une barre oblique entre les éléments dans le chemin d'accès est traitée par le système.   
Le répertoire **////usr//bin** est valide, mais sera interprétés par le système comme **/usr/bin**.

**Exemples** …

Le répertoire actuel est le répertoire personnel de l’usager.  
On souhaite se déplacer vers le répertoire **/usr/bin**.

Les deux commandes seront alors …

* Méthode de **chemin absolu** …  
  **>> cd /usr/sbin**Il est important de noter que la position actuelle n’est pas importante ;
* Méthode de **chemin relatif** …  
  **>> cd ../../usr/sbin**On doit remonter au répertoire racine (/ ou root) et redescendre par la suite dans le répertoire /usr/sbin.  
  Dans ce cas, la position actuelle (le point de départ) est importante.

**Exploration du système de fichiers**

Se déplacer de haut et en bas de l'arbre du système de fichiers peut s’avérer fastidieux si on ne connait pas bien l’arborescence.

La commande **tree** est un utilitaire intéressant afin d'obtenir une vue d'ensemble de l'arborescence du système de fichiers.   
**>> tree**Une image contenant texte

Description générée automatiquement

**Remarque** …  
La commande **tree** affiche le répertoire comme le point de départ.

L’ajout du commutateur **-d** (aucune forme longue pour cette option) à la commande **tree** permet d’afficher uniquement les répertoires sans la liste des noms de fichiers.

Les commandes suivantes aident à consulter l’arborescence du système de fichiers …

|  |  |
| --- | --- |
| Commande | Utilisation |
| cd / | Déplacement du répertoire actuel vers le répertoire racine (/ ou *root*) |
| ls | Affichage du contenu (liste) du répertoire de travail actuel |
| ls -a | Affichage de tous les fichiers, y compris les fichiers et répertoires cachés (fichier qui débutent par un point) |
| tree | Affichage d’une partie ou de la totalité de l’arborescence du système de fichiers |

**Gestion des fichiers et répertoires**

Linux possède 3 types de fichiers …

* **Fichiers réguliers**Les fichiers réguliers sont le type de fichier le plus courant sous Linux.   
  Ces fichiers peuvent être de toutes natures …  
  texte, images, binaires, ...   
  Ils constituent la majorité des fichiers présents sur un système Linux.
* **Répertoires**  
  Les répertoires sont des fichiers qui stockent d’autres fichiers.   
  Ils sont une sorte de conteneurs.
* **Fichiers spéciaux**  
  Les fichiers spéciaux représentent un périphérique physique, tel qu’un périphérique de stockage.  
  ils sont principalement utilisés pour les opérations d’entrées/sorties (E/S ou *I/O*).

**Création de répertoires**

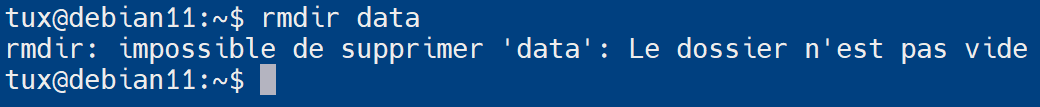
La commande **mkdir** est utilisée pour la création de répertoires ...  
**>> sudo mkdir <Option(s)> <Répertoire(s)>  
>> mkdir --verbose data  
>> mkdir --verbose data01 data02 data03**

**Une image contenant texte

Description générée automatiquement  
  
Suppression de répertoires**

La commande **rmdir** est utilisée pour la suppression de répertoires ...  
**>> sudo rmdir <Répertoire(s)>  
>> rmdir data  
>> rmdir data01 data02 data03**

La commande **rmdir** fonctionne seulement sur des répertoires vides.  
Si un répertoire n’est pas vide (s’il contient des fichiers ou d’autres répertoires) la commande **rmdir** retourne une erreur …



Pour supprimer un répertoire et son contenu, on utilise la commande **rm --recursive** (options courtes **-r** ou **-R**).

**Création de fichiers**

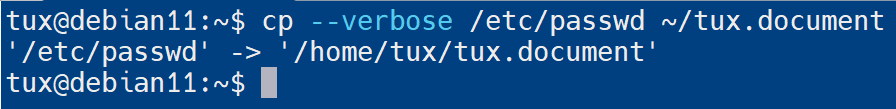
La commande **touch** peut être utilisée afin de créer un nouveau fichier au format texte ou ASCII.   
Elle crée et ouvre un nouveau fichier vide si le fichier si le nom du fichier demandé n’existe pas …  
**>> touch <Fichier(s)>  
>> touch tux.texte  
>> touch tux.texte squiddly.txt alcibiade.document  
>> touch fichier0{1..5}.txt  
  
Remarque** …  
Si le fichier existe déjà, le fichier ne sera pas affecté.  
La commande **touch** permet également de modifier l’horodatage (date de création, date de modification, …) des fichiers.

**Affichage du contenu d’un fichier**

La commande **cat** est utilisée pour afficher le contenu d’un fichier texte.   
Toutefois, si la sortie est très importante, l’ensemble du fichier est affiché en une seule fois.

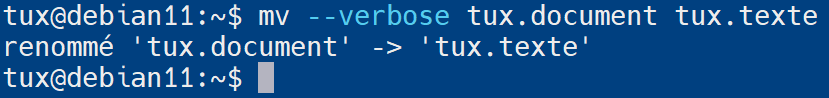
C’est pourquoi, pour les longs fichiers texte ou ASCII, il est préférable d’utiliser la commande **less**, qui affiche un fichier page par page.  
Les touches **Entrée**, **PgUP** et **PgDown** et les **flèches directionnelles** permettent de se déplacer dans le document.  
La touche **q** (pour quitter) permet de sortir de la commande **less**.

**Copie de fichier(s)**

La commande **cp** est utilisée pour créer une copie d’un fichier.   
Elle crée un nouveau fichier vers la destination (répertoire courant ou autre répertoire) avec le même nom (ou non) …  
**>> cp <Fichier source> <Fichier cible>  
>> cp <Chemin>/<Fichier source> <Chemin>/<Fichier cible>  
>> cp tux.texte tux.document  
>> cp /etc/passwd ~/tux.document  
>> cp --verbose /etc/passwd ~/tux.document  
  
**

**Remarque** …  
il est possible d’ajouter le commutateur **--verbose** (forme courte **-v**) afin d’afficher la réussite de la commande.

**Déplacement d’un fichier**

La commande **mv** est utilisée pour déplacer un fichier de sa source à sa destination.   
Elle est aussi utilisée pour renommer un fichier …  
**>> mv <Fichier source> <Fichier cible>  
>> mv <Chemin>/<Fichier source> <Chemin>/<Fichier cible>  
>> mv tux.document tux.texte  
>> mv /home/squiddly/squiddly.texte ~/tux.document  
>> mv --verbose /home/squiddly/squiddly.texte ~/tux.document**

**Remarque** …  
Il est également possible d’ajouter le commutateur **--verbose** (forme courte **-v**) afin d’afficher la réussite de la commande.

**Suppression d’un fichier**

La commande **rm** peut être utilisée pour supprimer un fichier …  
**>> rm <Fichier(s)>   
>> mv tux.document  
>> mv tux.document tux.texte  
>> mv --verbose tux.document**  
Une image contenant texte

Description générée automatiquement

**Liens**

La commande **ln** peut être utilisé afin de créer deux types de liens …

* **lien physique** (*hard link*)  
  À l’aide de la commande **ln** …  
  **>> ln --verbose <Fichier existant> <Nom du lien>** ;
* **lien symbolique** (*soft link*)  
  À l’aide de la commande **ln** suivie du commutateur **--symbolic** (forme courte **-s**) …  
  **>> ln –verbose --symbolic <Fichier existant> <Nom du lien>**.

**Lien physique**

Ces deux types de liens sont très utiles pour Linux et on les retrouve partout dans l’arborescence du système de fichiers.

**Exemple** …  
Le fichier **squiddly.texte** existe déjà.   
Un lien, appelé **alcibiade.texte** peut-être créé avec la commande …  
**>> echo "Je suis Squiddly" > squiddly.texte   
>> ln --verbose squiddly.texte alcibiade.texte**ou  
**>> ln -v squiddly.texte alcibiade.texte**

**Une image contenant texte

Description générée automatiquement**

**Remarque** …  
Encore une fois, le commutateur **--verbose** (forme courte **-v**) permet de confirmer la création du lien.

Au premier regard, deux fichiers semblent alors exister … **squiddly.texte** et **alcibiade.texte**.

Une image contenant texte

Description générée automatiquement

Toutefois, un examen plus approfondi de la liste des fichiers obtenue indique, qu’en réalité, il s’agit du même fichier.

Une image contenant table

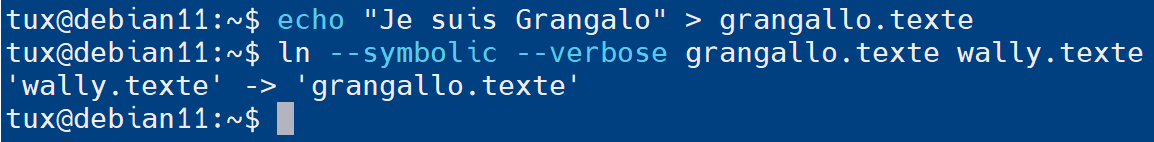
Description générée automatiquement

L'option **--inode** (forme courte -i) de la commande **ls** affiche, dans la première colonne, le numéro d'inode, qui est une valeur unique pour chaque fichier.   
On peut remarquer que ce champ est le même pour les fichiers **squiddly.texte** et **alcibiade.texte**.  
Ces deux entrées pointent cependant vers un seul et même fichier.  
Toutefois, pour ce fichier, plus d'un nom lui est associé.

On peut également remarquer que la valeur **2** apparaît dans la sortie de la commande ls  
Cela indique qu’il y a un autre objet (ou fichier) lié à **squiddly.texte**.  
En fait, ce fichier possède deux références.

**Lien symbolique**

Les liens symboliques sont en fait des raccourcis vers un autre fichier.  
On peut les créer avec la commande **ln** suivie du commutateur **--symbolic** (forme courte **-s**) …  
**>> echo "Je suis Grangalo" > grangallo.texte   
>> ln --symbolic --verbose grangallo.texte wally.texte**ou **>> ln -s -v grangallo.texte wally.texte**

****

Voici le résultat de la commande ls -l …

Une image contenant texte

Description générée automatiquement

**wally.texte** ne semble pas être un fichier régulier car il pointe clairement vers **grangallo.texte** et possède un numéro d'inode différent.

Les liens symboliques ne prennent pas d'espace supplémentaire au niveau du système de fichiers.   
Ils sont extrêmement pratiques car ils peuvent facilement être modifiés pour pointer vers des endroits différents.

Un moyen facile d’accéder à un répertoire avec de long chemins est de créer un lien symbolique.

Contrairement aux liens physiques, les liens symboliques peuvent pointer vers des objets, même si ces derniers sont situés sur un système de fichiers (ou une partition) différent.   
Ceci n’est pas le cas avec un lien physique.

Ces objets (fichiers ou répertoires) peuvent ou non être disponibles ou même existants au moment de la création du lien.   
Si le lien ne pointe pas vers un objet disponible ou existant, la commande ln suivie de l’option **--symbolic** crée un lien ballant (*dawnling link*).

Les liens symboliques sont très utiles.   
Toutefois, on doit les utiliser avec soin.   
Si on supprime soit **squiddly.texte** ou **alcibiade.texte**, l'objet inode (et le nom du fichier restant) demeure.

Ceci peut entrainer une situation non souhaitable car ce lien peut ultérieurement conduire à des erreurs subtiles lors de la création (à nouveau) d’un fichier avec un de ces noms.

**Remarque** …  
Si l’on modifie un des fichiers avec certains éditeurs, ces derniers conservent le lien par défaut.   
Cependant, il est possible que la modification de l'un des noms puisse briser (casser) le lien.  
Ce bris donne lieu à la création de deux objets.

**Englobement simple**

L’englobement est une fonctionnalité très importante d’un interpréteur de commandes (*shell*) Linux.

L'englobement sous Linux consiste à utiliser des métacaractères (tels que **\***, **?** ou **[ ]**) afin de faire correspondre une expression à plusieurs fichiers simultanément.   
Certains administrateurs système utilisent les termes *joker* ou *wildcard* (en anglais).  
Ces deux termes signifient à peu près la même chose.

Sous Linux, on parle toutefois de *globbing*.

Pour consulter l'aide sur l'englobement, on peut consulter la page de manuel **glob**, à la section 7 (section divers et conventions) ...  
**>> man 7 glob**

L’englobement peut être utilisé n'importe où dans un nom de fichier pour remplacer un ou plusieurs caractères.

**Astérisque \***

L'astérisque **\*** peut être utilisé pour correspondre à n'importe quelle chaîne, incluant la chaîne vide.

Par exemple, pour lister tous les fichiers du répertoire **/etc** dont le nom se termine par **.conf**...  
**>> ls /etc/\*.conf**

Pour lister tous les fichiers du répertoire **/var/log** débutant par **auth** ...  
**>> ls /var/\*.conf**

Les fichiers **auth.log\*** contiennent l'historique des connexions des utilisateurs sur le système d'exploitation.   
Dans ce fichier, on peut notamment voir les tentatives d'authentification qui ont échoué.   
Ce fichier est très utile lorsqu'on enquête sur une tentative de piratage d'un système Linux.

**Point d'interrogation ?**

Avant d'aller plus loin, on crée un jeu de fichiers pour réaliser des tests ...>>  
**>> mkdir --verbose glob  
>> cd glob  
>> touch fichier\_{A..D}{1..5}.txt  
>> touch fichier\_{a..d}{1..5}.txt**Le point d'interrogation **?** est utilisé pour remplacer un seul caractère dans un nom de fichier.   
Le caractère remplacé ne peut pas être nul comme c'est le cas avec **\***.

Pour afficher tous les fichiers du jeu précédent qui contiennent le chiffre **5** ...  
**>> ls fichier\_?5.txt**

On pourrait également utiliser le caractère **?** pour afficher tous les fichiers ayant l'identifiant a suivi de n'importe quelle valeur ...  
**>> ls fichier\_a?.txt**

**Crochets [ ]**

Les crochets sont utilisés pour définir un ensemble spécifique de caractères souhaités.

Voici quelques exemples d'ensembles de caractères ...

|  |  |
| --- | --- |
| Glob | Description |
| [agh] | Correspond à une lettre, soit a, g ou h |
| [0-9] | Correspond à n'importe quel chiffre entre 0 et 9 |
| [a-z] | Correspond à n'importe quelle lettre de l'alphabet |
| [[:lower:]] | Correspond à n'importe quelle lettre minuscule – Est équivalent à [a-z] |
| [[:upper:]] | Correspond à n'importe quelle lettre majuscule – Est équivalent à [A-Z] |

Par exemple, pour afficher tous les fichiers portant un identifiant avec une lettre majuscule suivi de n'importe quel caractère ...  
**>> ls fichier\_[[:upper:]]?.txt**

**Remarque** ...  
Deux englobements différents ont été utilisés dans la commande précédente, soit **[[:upper:]]** et **?**.

Pour afficher tous les fichiers portant la lettre a minuscule et un chiffre entre 1 et 3 ...  
**>> ls fichier\_a[1-3].txt**

**Point d'exclamation !**

Le point d'exclamation est utilisé en complément des crochets pour signifier une négation.

Par exemple, pour afficher tous les fichiers identifiés par la lettre **a** suivie d'un chiffre qui n'est pas **1**, **2** ou **3** ...  
**>> ls fichier\_a[!1-3].txt**

Quelques exemples d'englobement

|  |  |  |  |
| --- | --- | --- | --- |
| Expression | Signification | | Exemple |
| \* | Correspond à tous les fichiers et répertoires | **bob.txt** et **1-fichier.doc** | |
| a\* | Correspond à tous les fichiers et répertoires dont le nom commence par un a | **animal** et **anna.pdf** | |
| f\*.txt | Correspond à tous les fichiers et répertoires dont le nom commence par f et se termine par .txt | **fichier.txt** et **freddy.txt** | |
| p?? | Correspond à tous les fichiers et répertoires dont le nom commence par p suivi d'exactement deux caractères | **pas et pi4** | |
| f[a-z][0-9] | Correspond à tout fichier dont le nom commence par f, suivi d'une lettre, suivie d'un chiffre | **fa0** et **fb8** | |
| g[!a-z] | Correspond à tout fichier dont le nom commence par g, suivi d'un caractère qui n'est pas une lettre | **g\_** et **g6** | |
| [egp]\*.txt | Correspond à tout fichier dont le nom commence par e,g ou p, suivi de n'importe quels caractères et de .txt | **eat.txt, gat.txt, excellent.txt** et **pat.txt** | |

**Recherche dans l’arborescence de fichiers  
Utilitaires locate et updatedb**

**Utilitaires locate et updatedb**

Les utilitaires **locate** et **updatedb** sont utilisés afin de localiser rapidement un fichier correspondant à un motif (*pattern*) donné.

La commande **locate** effectue ses recherches dans une base de données indexée, base de données créée à la suite de l’exécution de la commande **updatedb**.   
Elle permet d’afficher très rapidement un résultat.

Certains résultats peuvent toutefois être imprécis selon la date de la dernière mise à jour de la base de données.

La façon la plus simple d'utiliser **locate** est de lui donner un motif à rechercher …  
**>> locate <Motif>   
>> locate jpg**

Le résultat de la commande **locate** affiche tous les objets qui contiennent le motif, peu importe ce qui vient avant ou après.

Par défaut, le modèle est sensible à la casse.   
Cela signifie que les fichiers contenants **.JPG** ne seront pas affichés car le motif est en minuscules.

Pour éviter un résultat incomplet, on utilise le commutateur **--ignore-case** (forme courte -i) …  
**>> locate --ignore-case <Motif(s)>  
>> locate --ignore-case jpg zip**ou  
**>> locate -i jpg zip**

**Remarque** …  
Il est possible de passer plusieurs motifs à la commande **locate**.

Si on souhaite afficher le compte des fichiers correspondant à un motif donné au lieu d'afficher leur chemin complet, on utilise le commutateur **--count** (forme courte **-c**) …  
**>> locate --count <Motif>  
>> locate --count .jpg**ou  
**>> locate -c .jpg**

Il existe toutefois une limite avec la commande **locate**.  
Le résultat de la commande ne montre que les entrées présentes dans la base de données générée par l’utilitaire **updatedb** (base de données située dans le fichier **/var/lib/mlocate.db**).   
Si cette base de données est obsolète (qu’elle n’a pas été mise à jour depuis un moment), la sortie peut afficher les fichiers qui ont été créés ou supprimés depuis la dernière mise à jour.   
Pour mettre la base de données **/var/lib/mlocate.db** à jour ...   
**>> sudo updatedb**

**Commande find**

La commande **find** est l'un des utilitaires de ligne de commande les plus importants et les plus utilisés avec les systèmes d'exploitation Linux.   
La commande **find** est un outil permettant de rechercher et de localiser fichiers et répertoires.   
Elle lance une recherche récursive par laquelle un ou plusieurs répertoires sont scrutés selon certains critères, tel que …

* les autorisations ;
* les utilisateurs ;
* les groupes ;
* le type de fichier ;
* les dates relatives (date de création, de dernière modification, … ;
* la taille ;
* ...

Lorsqu’aucun argument n’est précisé, l’utilitaire **find** affiche la liste de tous les fichiers dans le répertoire courant (et ses sous répertoires).

# Synopsis

La commande **find** permet la recherche des fichiers dans une hiérarchie de répertoires …  
**>> find <Chemin de recherche> <Option(s)> <Critère(s) de recherche>**

La commande est suivie du chemin d’accès au répertoire et des options de recherche.   
Une option de recherche est précédée d’un trait d’union suivi de l’option elle-même.

Le répertoire utilisé par défaut est le **répertoire courant** et l'option par défaut est **-print** (affichage du résultat à l’écran).

Une expression peut être constituée d'opérateurs, d'options, de tests et d'actions.

|  |  |
| --- | --- |
| Option | Signification |
| -name, -iname | Recherche par nom de fichier La sortie retourne seulement les fichiers de la liste avec un certain motif dans leur nom |
| -type | Recherche par type de fichier Cette option limite les résultats aux fichiers d'un certain type spécifié … **d** pour un répertoire ; **l** pour un lien symbolique ; **f** pour un fichier régulier ; … |
| -size, -empty | Recherche par taille de fichier Cette option limite les résultats aux fichiers avec une taille spécifiée (**=**, **>** ou **<**) … un nombre (**n**) signifie exactement une valeur ;  … **+n** spécifie une taille supérieure à une valeur ou  … **-n** spécifie une taille inférieure à une valeur |
| -ctime, -mtime, -atime | Recherche par date de création, date de dernière modification ou date de dernier accès |
| -user, -group | Recherche par propriétaire ou par groupe |
| -perm | Recherche par autorisations d’accès |
| -inum | Recherche par numéro d’inode |
| -links | Recherche sur le nombre de liens d’un fichier |

Plusieurs options de recherche peuvent être combinées.   
La commande **find** utilise une opération **and** (et) qui est implicite lorsque plusieurs options sont utilisées.   
De plus, une opération **or** (ou) peut être utilisé ou une condition peut être annulée (**not**) …

|  |  |
| --- | --- |
| Options | Significations |
| -and | Les résultats de la recherche doivent remplir les deux conditions |
| -or | Les résultats de la recherche doivent correspondre à au moins l’une des deux conditions |
| -not | La recherche ignore la condition |

La commande **find** est récursive.  
Peu importe le répertoire spécifié, elle effectue une recherche dans le répertoire spécifié, et les sous répertoires qu'il contient, et ainsi de suite …

**Rappel** …  
Si le répertoire de recherche n’est pas spécifié, la commande **find** effectue sa recherche dans le répertoire courant.

**Recherches avancées**

**Limitation de la recherche à un répertoire spécifique**

Pour rechercher dans le répertoire courant, on utilise le point (**.**) comme chemin d’accès relatif au répertoire ...  
**>> find . <Option(s)>**

Pour rechercher dans son répertoire personnel, on utilise le tilde (**~**) comme chemin d’accès relatif au répertoire ...  
**>> find ~ <Option(s)>**

Il est également possible de rechercher dans l’ensemble du système de fichiers.   
Toutefois, en raison du grand nombre de fichiers et éventuellement de répertoires, cela peut prendre un certain temps.

**Remarque** …  
Compte-tenu des permissions pour certains répertoires, il peut être utile de passer ces différentes commandes avec l’utilitaire **sudo**.

Pour rechercher dans tout le système, on utilise la barre oblique (**/**) comme chemin d’accès absolu au répertoire ...  
**>> sudo find / <Option(s)>**

**Personnalisation des résultats d’une recherche**

La commande **find** peut être très verbeuse.   
Il peut donc être utile de personnaliser afin de limiter la sortie de la commande.

Pour y arriver, on peut utiliser les pipelines (ou tubes).   
Les pipelines sont utilisés afin de rediriger la sortie de la commande **find** comme entrée pour une autre commande.

Pour afficher les résultats par page (à l’aide d’un *pager*), on redirige la sortie de la commande **find** à la commande less …  
**>> find <Chemin> <Option(s)> | less**

Pour afficher le nombre de correspondances, on redirige la sortie de la commande find à la commande wc avec le paramètre **--lines** (forme courte **‑l**) …  
**>> find <Chemin> <Option(s)> | wc --lines**  
ou  
**>> find <Chemin> <Option(s)> | wc -l**

Afin de n’afficher que les premiers ou les derniers résultats de recherche, on redirige la sortie de la commande find aux commandes **head** ou **tail**.   
On ajoute le paramètre **--lines** (forme courte **‑n**) suivi du nombre de lignes à afficher …  
**>> find <Chemin> <Option(s)> | head --lines <Nombre de ligne>**  
ou  
**>> find <Chemin> <Option(s)> | head -l <Nombre de ligne>**

**>> find <Chemin> <Option(s)> | tail --lines <Nombre de ligne>**  
ou  
**>> find <Chemin> <Option(s)> | tail -l <Nombre de ligne>**

**Exemples de recherche**

**Recherche sur le nom d’un objet**

Afin de rechercher les fichiers et ou répertoires nommés **bonjour.texte** du répertoire **/usr** …  
**>> find /usr -name bonjour.texte -print**

**Rappel** …  
L’option **-print** est par défaut.   
Elle peut donc être omise.  
**>> find /usr -name bonjour.texte** donne le même résultat.

Pour rechercher tous les fichiers se terminant par g dans le répertoire **/usr** …   
**>> find /usr -name "\*.sh" -print**

Pour rechercher une série de fichier ayant le même inode (**123456**) dans la totalité de l’arborescence de fichiers (**/**) …  
**>> find / -inode 123456**

Pour rechercher les objets en lecture seulement dans le répertoire courant …  
**>> find . -perm /u=r**

Pour rechercher les binaires (exécutables) dans le répertoire courant …  
**>>** **find . -perm /a=x**

**Recherche sur le type d’un objet**

Pour seulement rechercher les répertoires nommés **bonjour.texte** dans le **répertoire courant** …  
**>> find . -type d -name bonjour.texte**

Pour rechercher uniquement les fichiers réguliers nommés **bonjour.texte** du répertoire **/usr** …  
**>> find /usr -type f -name bonjour.texte**

Pour rechercher uniquement les fichiers réguliers nommés **nginx** du répertoire **/usr** …  
**>> find /usr -type f -name nginx**

**Recherche basée sur la taille d’un objet**

Il est également possible d’utiliser la commande **find** pour une recherche basée sur la taille ...  
**>>** **find / -size 0**

**Remarque** ...  
La taille s’affiche en blocs de 512 octets, par défaut.

Il également possible de spécifier octets (**c**), kilooctets (**k**), mégaoctets (**M**), gigaoctets (**G**), ...   
Tout comme les commutateurs de temps, la taille des fichiers peut également être des nombres exacts (**n**), **+n** ou **-n**.

Pour rechercher dans toute l'arborescence du système de fichiers, les fichiers dont la taille est supérieure à 1Mo (2000 blocs de 512o) …  
**>>** **find / -size 2000 -print**

**Recherche basée sur le temps**

Il est parfois utile de chercher des fichiers en fonction d'attributs tels que …

* la date de création ;
* la date de la dernière utilisation  
  ou encore
* la taille.

Pour rechercher des objets en fonction du temps …  
**>>** **find / -ctime 3**

Il est également possible de rechercher en fonction des accès/dernière lecture (**-atime**) ou encore des dernières modifications (**-mtime**).

Le nombre correspond au nombre de jours et peut être exprimé comme …

* un nombre (**n**) qui signifie exactement de cette valeur ;
* **+n** supérieur à ce nombre  
  ou
* **-n** inférieur à ce nombre.

**Remarque** …  
Il existe également des options similaires pour les temps en minutes (comme dans **-cmin**, **-amin** et **-mmin**).

**Archivage de fichiers  
Commande tar (Archivage et Compression)**

L’utilitaire **tar**, (abréviation de *tape archive(r))*, est utilisée pour créer des archives en transformant un groupe de fichiers en une archive.

L’utilitaire tar est un utilitaire qui crée en ensemble dans lequel les fichiers peuvent être attachés, regroupés et facilement déplacés.

La commande tar possède également les capacités …

* d'extraire des archives tar ;
* d'afficher une liste des fichiers inclus dans l'archive   
  ainsi que
* d'ajouter des fichiers supplémentaires à une archive existante.

La syntaxe de l’utilitaire tar est la suivante …  
**>> tar [Opération et Option(s)] [Nom de l’archive] [Nom(s) de fichier]**

**Opération**  
Un seul argument d'opération est autorisé et obligatoire.  
Les opérations les plus fréquemment utilisées sont …

* **--create** (option courte **-c**)  
  Création d’une nouvelle archive tar ;
* **--extract** (option courte **-x**)  
  Extraction de l'intégralité de l'archive ou un ou plusieurs fichiers d'une archive ;
* **--list** (option courte **-t**)  
  Affichage d’une liste des fichiers inclus dans l'archive.

**Options**  
Les options les plus fréquemment utilisées sont …

* **--verbose** (option courte **-v**)  
  Affichage des fichiers en cours de traitement par la commande tar ;
* -**-file=<Nom de l’archive>** (option courte **-f**)  
  Spécification du nom du fichier d'archive ;
* **<Nom de l’archive>**  
  Spécification du nom de l'archive ;
* **<Nom des fichiers à archiver>**  
  Spécification d’une liste de noms de fichiers séparés par des espaces à extraire.   
  S'il n'est pas fourni, l'intégralité de l'archive est extraite.

**Création d’une archive**

Pour créer une archive …  
**>> tar --create --verbose --file archive.tar /home/tux  
>> tar -cvf archive.tar /home/tux**

Signification des commutateurs de la commande …

* **--create** (option courte **-c**)  
  Création d’une archive ;
* **--verbose** (option courte **-v**)  
  Affichage de la progression dans le terminal lors de la création de l'archive (mode verbeux).   
  Le commutateur --verbose est toujours facultatif ;
* **--file** (option courte **-f**)  
  Spécification du nom de fichier de l'archive.

En général pour archiver un seul répertoire ou un seul fichier …  
**>> tar --create --verbose --file archive.tar /home/tux  
>> tar -cvf archive.tar /home/tux**

**Remarque** …  
L’utilitaire tar fonctionne de manière récursive.   
Il effectuera l'action requise sur chaque répertoire suivant à l'intérieur du répertoire spécifié.

Pour archiver plusieurs répertoires à la fois …  
**>> tar --create --verbose --file archive.tar /home/tux /home/squiddly /home/wally  
>> tar -cvf archive.tar rep01 home/tux /home/squiddly /home/wally**

**Extraction d'une archive**

On peut extraire une archive en utilisant …  
**>> tar --extract --verbose --file archive.tar   
>> tar --get --verbose --file archive.tar   
>> tar -xvf archive.tar**

Cette commande extrait le contenu du fichier archive.tar dans le répertoire courant.

Pour extraire le contenu de l'archive dans un répertoire spécifique …  
**>> tar --extract --verbose --file archive.tar --directory /tmp  
>> tar --get --verbose --file archive.tar -- directory /tmp  
>> tar -xvf archive.tar -C /tmp**

Ces commandes extraient le contenu du fichier archive.tar dans le répertoire /tmp.

**Compression avec tar**

L’utilitaire tar peut créer une archive.tar et par la suite la compresser avec gzip ou bzip2 …  
**>> tar --create --gzip --verbose --file <Nom de l’archive.tar.gz>**ou **>> tar -czvf <Nom de l’archive.tar.gz>**

Cette commande crée un fichier compressé à l'aide de l'algorithme gzip (**--gzip**).

Bien que la compression gzip soit le plus souvent utilisée pour créer des fichiers **.tar.gz** ou **tgz**, tar prend également en charge la compression **bzip2**.   
On remplace **--gzip** (**-z**) par **--bzip2** (**-j**) ou **--xz** (**-J**) …  
**>> tar --create --bzip2 --verbose --file <Nom de l’archive.tar.bz>**ou **>> tar -cjvf <Nom de l’archive.tar.bz>**

Pour décompresser le fichier …  
**>> tar --extract --gzip --verbose --file archive.tar.gz --directory /tmp  
>> tar -xzvf archive.tar.gz --directory /tmp**,

On remplace -c par -x, où x signifie **extraire**.

L’utilitaire **gzip** est plus rapide mais il donne une compression généralement un peu moins importante.  
On obtient donc un fichier un peu plus volumineux.

L’utilitaire **bzip2** est plus lent, mais le résultat est compressé un peu plus  
On obtient donc un fichier un peu plus petit.

Alternativement, il est possible d’utiliser les commandes **gzip** et **bzip** …  
**>> gzip <Fichier à compresser>**

L’utilitaire **gzip** crée le fichier compressé avec le même nom mais avec le suffixe **.g**z.  
Il supprime le fichier d'origine après avoir créé le fichier compressé.

L’utilitaire bzip2 fonctionne de la même manière.

Pour décompresser les fichiers, on utilise soit **gunzip** ou **bunzip2** selon l'algorithme utilisé pour compresser un fichier.