Reprenez le contrôle à l'aide de Linux !

30 heures Facile

Mis à jour le 29/06/2021

La structure des dossiers et fichiers

Ahhh, les fichiers sous Linux, tout un programme.
Vous croyez savoir ce que sont les fichiers et dossiers ? Vous croyez que votre disque dur s'appelle `C:` ? Que le lecteur CD c'est `D:` ou peut-être `E:` ?

Les choses ne fonctionnent pas du tout de la même manière sous Linux et sous Windows. Or, savoir comment se déplacer de dossier en dossier et savoir lister les fichiers, c'est quand même sacrément important ! C'est pour cela que nous allons voir ensemble le fonctionnement des fichiers sous Linux dès maintenant.

Organisation des dossiers

Le système qui gère les fichiers sous Linux est un peu déroutant au début, surtout quand on est habitué à celui de Windows. En effet, ici vous ne trouverez pas de `C:\` , `D:\` ou que sais-je encore. Les fichiers sont organisés d'une manière complètement différente.

Au lieu de séparer chaque disque dur, lecteur CD, lecteur de disquettes, lecteur de carte mémoire... Linux place en gros tout au même endroit.

> Mais comment on fait pour savoir si le dossier dans lequel on est appartient au premier disque dur, au second disque dur, au lecteur CD... ? C'est le bazar, non ?

C'est ce qu'on pourrait croire au premier abord, mais en fait c'est juste une autre façon de penser la chose. ;-)

Deux types de fichiers

Pour faire simple, il existe deux grands types de fichiers sous Linux :

- **les fichiers classiques** : ce sont les fichiers que vous connaissez, ça comprend les fichiers texte (`.txt` , `.doc` , `.odt` ...), les sons (`.wav` , `.mp3` , `.ogg`), mais aussi les programmes. Bref, tout ça, ce sont des fichiers que vous connaissez et que vous retrouvez dans Windows ;
- **les fichiers spéciaux** : certains autres fichiers sont spéciaux car ils **représentent** quelque chose. Par exemple, votre lecteur CD est un fichier pour Linux. Là où Windows fait la distinction entre ce qui est un fichier et ce qui ne l'est pas, Linux, lui, dit que **tout est un fichier**. C'est une conception très différente, un peu déroutante comme je vous l'ai dit, mais pas de panique, vous allez vous y faire.

La racine

Dans un système de fichiers, il y a toujours ce qu'on appelle une racine, c'est-à-dire un « **gros dossier de base qui contient tous les autres dossiers et fichiers** ».

Sous Windows, il y a en fait plusieurs racines. `C:\` est la racine de votre disque dur, `D:\` est la racine de votre lecteur CD (par exemple).
Sous Linux, **il n'y a qu'une et une seule racine** : « `/` ». Comme vous le voyez, il n'y a pas de lettre de lecteur car justement, Linux ne donne pas de nom aux lecteurs comme le fait Windows. Il dit juste « **La base, c'est `/`** ».

Il n'y a pas de dossier de plus haut niveau que `/` , c'est-à-dire qu'il n'existe pas de dossier qui contienne le dossier `/` . Quand on est à la racine, on ne peut pas remonter en arrière car... on est déjà tout au début.

Architecture des dossiers

Sous Windows, un dossier peut être représenté de la manière suivante : `C:\Program Files\Winzip` . On dit que `Winzip` est un sous-dossier du dossier `Program Files` , lui-même situé à la racine.
Vous noterez que c'est l'antislash `\` (aussi appelé *backslash*) qui sert de séparateur aux noms de dossiers.

Sous Linux, c'est au contraire le `/` qui sert de séparateur.
Comme je vous l'ai dit, il n'y a pas de `C:` sous Linux, la racine (le début) s'appelant juste `/` .

Le dossier de notre superprogramme ressemblerait plutôt à quelque chose comme cela : `/usr/bin/` . On dit que `bin` est un sous-dossier du dossier `usr` , lui-même situé à la racine.

Linux gère sans problème les noms de fichiers et dossiers contenant des espaces, des accents et des majuscules. Toutefois, vous remarquerez que la plupart du temps on préfère les éviter. On trouve ainsi plutôt des noms tout en minuscules sans accents ni espaces, comme `usr` , `bin` , `apache` , etc.
Souvenez-vous qu'il n'est pas obligatoire de nommer vos fichiers en suivant la même règle, mais la plupart des programmes que vous installerez préfèrent utiliser des noms tout en minuscules sans espaces ni accents, ne soyez donc pas surpris.

Les dossiers de la racine

Sous Windows, on a l'habitude de trouver souvent les mêmes dossiers à la racine :
`Documents and Settings` , `Program Files` , `Windows` ...
Sous Linux, vous vous en doutez, les dossiers sont complètement différents. Et l'on ne risque pas de trouver de dossier qui s'appelle Windows !

Je vais vous faire ici la liste des dossiers les plus courants que l'on retrouve à chaque fois à la racine de Linux.
La description de chaque dossier sera rapide, mais c'est juste pour que vous puissiez vous repérer au début. ;-)

Il n'est PAS nécessaire de retenir cette liste par cœur. D'ailleurs je n'ai mis que les dossiers principaux, et elle est quand même longue.
Servez-vous-en juste si vous avez besoin de savoir à quoi correspond grosso-modo tel ou tel dossier, mais ne vous en faites pas si vous ne maîtrisez pas à fond le sens de chacun de ces dossiers.

- **bin** : contient des programmes (exécutables) susceptibles d'être utilisés par tous les utilisateurs de la machine.
- **boot** : fichiers permettant le démarrage de Linux.
- **dev** : fichiers contenant les périphériques. En fait – on en reparlera plus tard – ce dossier contient des sous-dossiers qui « représentent » chacun un périphérique. On y retrouve ainsi par exemple le fichier qui représente le lecteur CD.
- **etc** : fichiers de configuration.
- **home** : répertoires personnels des utilisateurs. On en a déjà parlé un peu avant : c'est dans ce dossier que vous placerez vos fichiers personnels, à la manière du dossier `Mes documents` de Windows.

Chaque utilisateur de l'ordinateur possède son dossier personnel. Par exemple, dans mon cas mon dossier personnel se trouve dans `/home/mateo21/` . S'il y avait un autre utilisateur (appelons-le Patrick) sur mon ordinateur, il aurait eu droit lui aussi à son propre dossier : `/home/patrick/` .

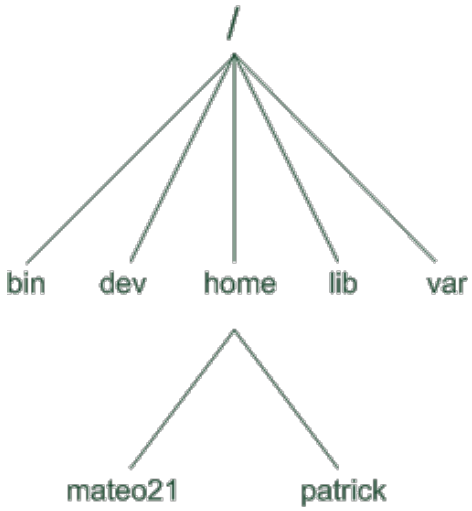
- **lib** : dossier contenant les bibliothèques partagées (généralement des fichiers `.so`) utilisées par les programmes. C'est en fait là qu'on trouve l'équivalent des `.dll` de Windows.
- **media** : lorsqu'un périphérique amovible (comme une carte mémoire SD ou une clé USB) est inséré dans votre ordinateur, Linux vous permet d'y accéder à partir d'un sous-dossier de `media` . On parle de **montage**.
- **mnt** : c'est un peu pareil que `media` , mais pour un usage plus temporaire.
- **opt** : répertoire utilisé pour les *add-ons* de programmes.
- **proc** : contient des informations système.
- **root** : c'est le dossier personnel de l'utilisateur « root ». Normalement, les dossiers personnels sont placés dans `home` , mais celui de « root » fait exception. En effet, comme je vous l'ai dit dans le chapitre précédent, « root » est le superutilisateur, le « chef » de la machine en quelque sorte. Il a droit à un espace spécial.
- **sbin** : contient des programmes système importants.

- **tmp** : dossier temporaire utilisé par les programmes pour stocker des fichiers.
- **usr** : c'est un des plus gros dossiers, dans lequel vont s'installer la plupart des programmes demandés par l'utilisateur.
- **var** : ce dossier contient des données « variables », souvent des *logs* (traces écrites de ce qui s'est passé récemment sur l'ordinateur).

Cette liste de dossiers est en fait présente sur tous les OS de type Unix, et pas seulement sous Linux.
Encore une fois, ne retenez pas tout ça. C'est juste pour vous donner une idée de ce que contiennent les dossiers à la racine de Linux, car je sais que c'est une question qu'on se pose souvent quand on débute.

Schéma résumé de l'architecture

Pour que vous vous y repériez correctement, sachez qu'on peut présenter l'organisation des dossiers de Linux comme le suggère la figure suivante.



Organisation des dossiers de Linux

La racine tout en haut est `/` ; elle contient plusieurs dossiers, qui contiennent chacun eux-mêmes plusieurs dossiers, qui contiennent des dossiers et fichiers, etc.

pwd & which : où... où suis-je ?

Le nombre de dossiers et de fichiers présents après l'installation d'Ubuntu est tellement grand qu'il serait facile de s'y perdre. Un grand nombre de programmes sont en effet préinstallés pour que vous puissiez profiter rapidement des possibilités de Linux.
Ne comptez donc pas sur moi pour vous faire la liste complète des dossiers et fichiers que vous possédez, ce n'est pas réaliste.

En revanche, je vais vous apprendre maintenant à vous repérer dans l'arborescence des dossiers. Vous saurez alors à tout moment où vous êtes sur votre disque. C'est un peu comme avoir une carte routière, en quelque sorte !

`pwd` : afficher le dossier actuel

Lorsque vous ouvrez la console pour la première fois, Linux vous place dans votre dossier personnel, votre `home`. En l'occurrence dans mon cas, le dossier dans lequel je serai placé sera `/home/mateo21`.

Normalement, l'invite de commandes vous indique le nom du dossier dans lequel vous vous trouvez :

```
mateo21@mateo21-desktop:~$
```

Si vous vous souvenez bien, le nom du dossier est situé entre le « : » et le « $ ». Donc ici, on se trouve dans le dossier « `~` ».

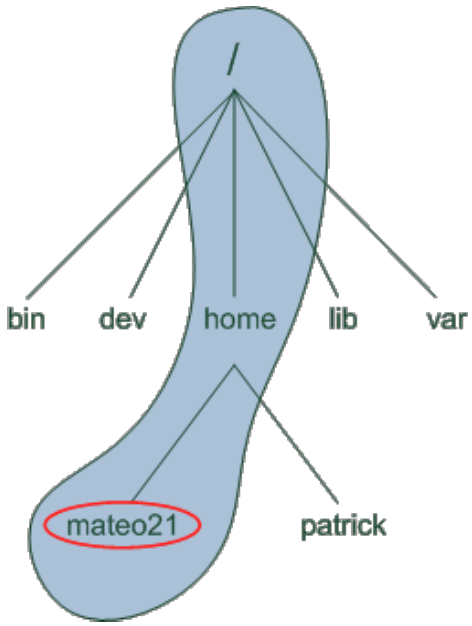
Rappel : je l'ai dit dans le chapitre précédent mais ça ne fait pas de mal de le répéter, sous Linux le symbole `~` est un synonyme de votre dossier personnel. Chez moi cela signifie donc `/home/mateo21`.

Cette indication de l'invite de commandes est pratique mais il faut savoir qu'il y a un autre moyen de connaître le nom du dossier actuel. C'est la commande **`pwd`**.
`pwd` est l'abréviation de « Print Working Directory », c'est-à-dire « Afficher le dossier actuel ».

C'est une commande très simple qui ne prend aucun paramètre (on commence doucement, hein !), vous pouvez la tester :

```
mateo21@mateo21-desktop:~$ pwd
/home/mateo21
```

Cela confirme bien ce que je vous disais : je me trouve en ce moment dans le dossier `/home/mateo21` (figure suivante).



Dossier /home/mateo21

À tout moment, si vous vous sentez perdus dans les méandres des dossiers, souvenez-vous de la commande `pwd`, elle vous dira où vous êtes ! ;-)

`which` : connaître l'emplacement d'une commande

Même si cette commande ne nous est pas indispensable, j'ai pensé que c'était une bonne idée de vous la montrer dès le début afin que vous puissiez vous familiariser un peu plus encore avec le système de fichiers de Linux.

Alors, que fait cette commande ? Elle vous permet de localiser la position du programme correspondant à une commande.
Je m'explique : chaque commande sous Linux correspond à un programme. Ainsi, `pwd` qu'on vient de voir **est** un programme.

Une commande n'est rien d'autre qu'un programme qu'on peut appeler n'importe quand et n'importe où dans la console.

La commande `which` prend un paramètre : le nom de la commande dont vous voulez connaître l'emplacement.
Testons sur `pwd` :

```
mateo21@mateo21-desktop:~$ which pwd
/bin/pwd
```

`pwd` se trouve donc dans le dossier `/bin/` ! Le « pwd » à la fin n'est pas un dossier mais le nom du programme lui-même.

Vous noterez que les programmes sous Linux ne possèdent en général pas d'extension (contrairement à Windows où l'extension utilisée est en général `.exe`).

Tous les programmes ne sont pas situés dans un même dossier. Pour vous en rendre compte, testez l'emplacement d'une autre commande... tenez, par exemple la commande `which` !

On va donc devoir écrire `which which` dans la console (oui, je sais, je suis un gros tordu.) :

```
mateo21@mateo21-desktop:~$ which which
/usr/bin/which
```

Cette fois, le programme ne se trouve pas dans `/bin/` mais dans `/usr/bin/` !

ls : lister les fichiers et dossiers

`ls` est une des toutes premières commandes que nous avons essayées dans le chapitre précédent. Nous allons rentrer ici plus dans le détail de son fonctionnement (et de ses nombreux paramètres...).

Commençons par taper « ls » sans paramètre depuis notre dossier personnel (oui : j'ai créé quelques dossiers pour mon usage personnel, ne vous étonnez pas si vous n'avez pas les mêmes.) :

```
mateo21@mateo21-desktop:~$ ls
Desktop  Examples  images  log  tutos
```

Ubuntu active la coloration des fichiers et dossiers par défaut, vous devriez donc voir des couleurs chez vous. Les dossiers apparaissent en bleu foncé. Vous remarquerez que le dossier `Examples` est en bleu clair : cela signifie que c'est un raccourci vers un dossier qui se trouve en fait ailleurs sur le disque.

Si la couleur ne s'affiche pas, vous pouvez rajouter le paramètre `--color=auto`, comme ceci : `ls --color=auto`.
Si vous ne voulez pas de la couleur au contraire, essayez le paramètre `--color=none`.
Pour éviter d'avoir à taper à chaque fois ce long paramètre, il faut modifier un fichier de configuration, mais on verra cela plus tard.

La commande `ls` accepte un grand nombre de paramètres. Ça ne sert à rien que je vous fasse la liste complète ici (ce serait bien trop long) ; par contre, je vais vous faire découvrir les paramètres les plus utiles. Ça vous permettra de vous entraîner à utiliser et combiner des paramètres !

`-a` : afficher tous les fichiers et dossiers cachés

Sous Linux, on peut « cacher » des fichiers et dossiers. Ce n'est pas une protection, car on peut toujours les réafficher si on veut, mais ça évite d'encombrer l'affichage de la commande `ls`.

Votre dossier `home` est un très bon exemple car il est rempli de fichiers et dossiers cachés. En ajoutant le paramètre `-a`, on peut voir tous ces fichiers et dossiers cachés :

```
mateo21@mateo21-desktop:~$ ls -a
.                               .gnome                .nano_history
..                              .gnome2               .nautilus
.armagetron                     .gnome2_private       .openoffice.org2
.bash_history                   .gnome_private        .pgadmin3
.bash_logout                    .gstreamer-0.10       .pgpass
.bashrc                         .gtkrc-1.2-gnome2     .profile
.blender                        .gweled               .qt
.config                         .ICEauthority         .recently-used
.DCOPserver_mateo21-desktop__0  .icons                .recently-used.xbel
.DCOPserver_mateo21-desktop_:0  images                .ssh
Desktop                         .inkscape             .sudo_as_admin_success
.dmrc                           .java                 .themes
.emilia                         .jedit                .thumbnails
.esd_auth                       .kde                  .Trash
.evolution                      .lesshst              .tsclient
Examples                        .lgames               tutos
.face                           .local                .update-manager-core
```

```
.fontconfig                  log                  .update-notifier
.gaim                        .macromedia          .vlc
.gconf                       .mcop                .wormux
.gconfd                      .mcoprc              .Xauthority
.geany                       .metacity            .xine
.gimp-2.2                    .mozilla             .xsession-errors
.gksu.lock                   .mozilla-thunderbird
```

Vous comprenez peut-être mieux maintenant pourquoi tous ces fichiers et dossiers sont cachés : c'est encombrant.
Certains éléments commençant par un point « . » sont des dossiers, d'autres sont des fichiers. La meilleure façon de faire la distinction est de comparer les couleurs : les dossiers en bleu, le reste dans la couleur par défaut (par exemple, le blanc ou le noir).

Les deux premiers éléments sont assez intrigants : « . » et « .. ». Le premier représente en fait le dossier actuel, et « .. » représente le dossier parent, c'est-à-dire le dossier précédent dans l'arborescence. Par exemple, là je suis dans `/home/mateo21` , « .. » représente donc le dossier `/home` .

Le paramètre `-A` (un « A » majuscule au lieu d'un « a » minuscule) a pratiquement la même signification : cela affiche la même chose sauf ces éléments « . » et « .. ». Comme quoi il faut faire attention aux majuscules !

`-F` : indique le type d'élément

Ce paramètre est surtout utile pour ceux qui n'ont pas affiché la couleur dans la console (ou n'en veulent pas). Il rajoute à la fin des éléments un symbole pour qu'on puisse faire la distinction entre les dossiers, fichiers, raccourcis...

```
mateo21@mateo21-desktop:~$ ls -F
Desktop/  Examples@  images/  log/  tutos/
```

Grâce à ça on peut voir que tous les éléments sont des dossiers, sauf `Examples` qui est un raccourci (d'où la présence du `@`).

`-l` : liste détaillée

Le paramètre `-l` (la lettre « L » en minuscule) est un des plus utiles. Il affiche une liste détaillant chaque élément du dossier :

```
mateo21@mateo21-desktop:~$ ls -l
total 16
drwxr-xr-x 2 mateo21 mateo21 4096 2007-09-24 17:22 Desktop
lrwxrwxrwx 1 mateo21 mateo21   26 2007-09-19 18:31 Examples -> /usr/share/example-content
drwxr-xr-x 2 mateo21 mateo21 4096 2007-09-25 15:17 images
drwxr-xr-x 3 mateo21 mateo21 4096 2007-09-25 11:11 log
drwxr-xr-x 3 mateo21 mateo21 4096 2007-09-19 19:51 tutos
```

Il y a un élément par ligne.
Chaque colonne a sa propre signification. De gauche à droite :

1. droits sur le fichier (on fera un chapitre entier pour expliquer comment fonctionnent les droits sous Linux) ;
2. nombre de liens physiques (cela ne nous intéresse pas ici) ;
3. nom de la personne propriétaire du fichier (là, c'est moi !). Si le fichier avait été créé par quelqu'un d'autre, par exemple Patrick, on aurait vu son nom à la place ;
4. groupe auquel appartient le fichier (on en reparlera dans le chapitre sur les droits). Il se peut que le nom du groupe soit le même que celui du propriétaire ;
5. taille du fichier, en octets ;
6. date de dernière modification ;
7. nom du fichier (ou dossier).

Vous noterez aussi que dans le cas du raccourci (on parle de **lien symbolique**), la commande nous précise vers où pointe le raccourci (en l'occurrence `/usr/share/example-content`).

`-h` : afficher la taille en Ko, Mo, Go...

Quand on fait un `ls -l` , la taille est affichée en octets. Seulement, ce n'est parfois pas très lisible. Par exemple :

```
mateo21@mateo21-desktop:~/Examples$ ls -l
total 9500
-rw-r--r-- 1 root root 3576296 2007-04-03 17:05 Experience ubuntu.ogg
-rw-r--r-- 1 root root  229674 2007-04-03 17:05 fables_01_01_aesop.spx
-rw-r--r-- 1 root root  848013 2007-04-03 17:05 gimp-ubuntu-splash.xcf
-rw-r--r-- 1 root root 1186219 2007-04-03 17:05 kubuntu-leaflet.png
-rw-r--r-- 1 root root   47584 2007-04-03 17:05 logo-Edubuntu.png
```

Si vous rajoutez le paramètre `h` (« h » pour *Human Readable*, c'est-à-dire « lisible par un humain »), vous obtenez des tailles de fichiers beaucoup plus lisibles (normal, vous êtes des humains) :

```
mateo21@mateo21-desktop:~/Examples$ ls -lh
total 9,3M
-rw-r--r-- 1 root root 3,5M 2007-04-03 17:05 Experience ubuntu.ogg
-rw-r--r-- 1 root root 225K 2007-04-03 17:05 fables_01_01_aesop.spx
-rw-r--r-- 1 root root 829K 2007-04-03 17:05 gimp-ubuntu-splash.xcf
-rw-r--r-- 1 root root 1,2M 2007-04-03 17:05 kubuntu-leaflet.png
-rw-r--r-- 1 root root  47K 2007-04-03 17:05 logo-Edubuntu.png
```

Grâce à ça, on voit alors bien que le fichier `Experience ubuntu.ogg` fait 3,5 Mo, `logo-Edubuntu.png` fait 47 Ko, etc.

`-t` : trier par date de dernière modification

Voilà une option dont l'intérêt est sous-estimé ! `-t` permet en effet de trier par date de dernière modification, au lieu de trier par ordre alphabétique comme cela est fait par défaut.
On voit ainsi en premier le dernier fichier que l'on a modifié, et en dernier celui auquel on n'a pas touché depuis le plus longtemps :

```
mateo21@mateo21-desktop:~$ ls -lt
total 16
drwxr-xr-x 2 mateo21 mateo21 4096 2007-09-25 15:17 images
drwxr-xr-x 3 mateo21 mateo21 4096 2007-09-25 11:11 log
drwxr-xr-x 2 mateo21 mateo21 4096 2007-09-24 17:22 Desktop
drwxr-xr-x 3 mateo21 mateo21 4096 2007-09-19 19:51 tutos
lrwxrwxrwx 1 mateo21 mateo21   26 2007-09-19 18:31 Examples -> /usr/share/example-content
```

De toute évidence, le dernier fichier (ici, c'est un dossier) modifié est « images ». En revanche, comme je n'ai jamais touché à « Examples », il est normal qu'il apparaisse en dernier.

En pratique, je combine `-t` avec `-r` qui renverse l'ordre d'affichage des fichiers. Je préfère en effet avoir le dernier fichier modifié en bas de la liste, c'est plus pratique à l'usage dans la console.
Et comme je suis un gros bourrin, je combine un peu tous les paramètres que l'on vient de voir, ce qui donne un beau `ls -larth` qui contient toutes les options que j'aime. ;-)

```
mateo21@mateo21-desktop:~$ ls -larth
total 380K
-rw-------  1 mateo21 mateo21   26 2007-09-19 16:40 .dmrc
-rw-r--r--  1 mateo21 mateo21   89 2007-09-19 16:40 .gtkrc-1.2-gnome2
-rw-------  1 mateo21 mateo21   16 2007-09-19 16:40 .esd_auth
drwx------  2 mateo21 mateo21 4,0K 2007-09-19 16:40 .update-notifier
lrwxrwxrwx  1 mateo21 mateo21   26 2007-09-19 18:31 Examples -> /usr/share/example-content
-rw-r--r--  1 mateo21 mateo21  220 2007-09-19 18:31 .bash_logout
drwxr-xr-x  4 root    root    4,0K 2007-09-19 18:31 ..
drwxr-xr-x 10 mateo21 mateo21 4,0K 2007-09-25 16:03 .jedit
-rw-r--r--  1 mateo21 mateo21 1,1K 2007-09-25 16:03 .pgadmin3
drwxr-xr-x 47 mateo21 mateo21 4,0K 2007-09-25 16:03 .
-rw-------  1 mateo21 mateo21 1,8K 2007-09-25 16:38 .bash_history
-rw-------  1 mateo21 mateo21  17K 2007-09-25 16:52 .recently-used
drwx------  2 mateo21 mateo21 4,0K 2007-09-25 16:54 .gconfd
-rw-------  1 mateo21 mateo21   39 2007-09-25 17:18 .lesshst
-rw-r--r--  1 mateo21 mateo21  53K 2007-09-25 17:21 .xsession-errors
```

Note : j'ai volontairement réduit cette liste car il y a beaucoup de fichiers dans mon `home` . En pratique la liste est beaucoup plus grande.

Le fichier caché « .xsession-errors » est donc le dernier qui a été modifié dans ce dossier sur mon ordinateur.

Plutôt que d'avoir à réécrire `ls -larth` à chaque fois (c'est un peu long), on peut créer un alias, c'est-à-dire une commande synonyme. Par exemple, j'ai créé l'alias `ll` (deux fois « L ») qui est automatiquement transformé par Linux en `ls -larth` .
On verra comment créer des alias lorsqu'on saura se servir d'un éditeur de fichiers.

cd: changer de dossier

Bon : mine de rien, depuis tout à l'heure on est coincé dans notre dossier `home` et on aimerait bien bouger de là.
Le moment est venu de déplacer le navire, moussaillon !

La commande que nous allons étudier ici s'appelle `cd` , abréviation de *Change Directory* (changer de dossier). C'est une commande très importante que vous allez utiliser quelques milliers de fois dans votre vie (au moins).

Contrairement à `ls` , la commande `cd` ne prend pas plein de paramètres mais juste un seul : le nom du dossier dans lequel vous souhaitez aller.

Si on veut aller à la racine, il suffit de taper `cd /` :

```
mateo21@mateo21-desktop:~$ cd /
mateo21@mateo21-desktop:/$ pwd
/
```

Après avoir tapé `cd /` , on se retrouve à la racine. L'invite de commandes a changé et le `~` a été remplacé par un `/` . Si vous êtes sceptiques, un petit coup de `pwd` devrait vous confirmer que vous êtes bien dans `/` .

Bien ! Listons les fichiers et dossiers contenus dans `/` :

```
mateo21@mateo21-desktop:/$ ls -F
bin/    dev/   initrd/          lib/         mnt/   root/  sys/  var/
boot/   etc/   initrd.img@      lost+found/  opt/   sbin/  tmp/  vmlinuz@
cdrom@  home/  initrd.img.old@  media/       proc/  srv/   usr/  vmlinuz.old@
```

Vous y retrouvez un grand nombre de dossiers que je vous ai décrits au début du chapitre.
Allons dans le sous-dossier `usr` :

```
mateo21@mateo21-desktop:/$ cd usr
```

Voyons voir ce qu'il y a là-dedans...

```
mateo21@mateo21-desktop:/usr$ ls -F
bin/  games/  include/  lib/  local/  sbin/  share/  src/  X11R6/
```

Chez moi, il n'y a que des dossiers. Hummm, le dossier `games` m'intrigue, voyons voir ce que j'ai comme

jeux :

```
mateo21@mateo21-desktop:/usr$ cd games
mateo21@mateo21-desktop:/usr/games$
```

Schématiquement, on vient de faire ce qui est illustré dans la figure suivante.

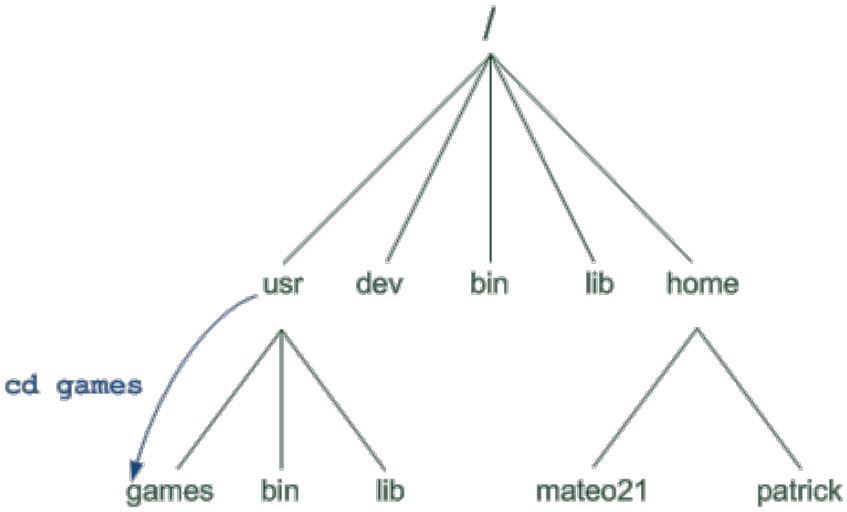


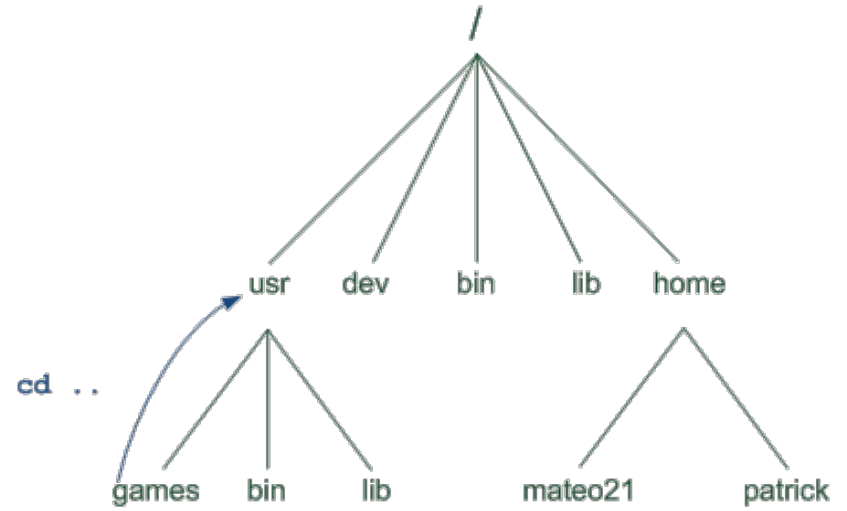
Illustration de la commande cd

Supposons maintenant que j'aie envie de revenir au dossier précédent, aussi appelé dossier parent, c'est-à-dire `/usr` . Comment je fais ?

Il faut utiliser les deux points comme ceci :

```
mateo21@mateo21-desktop:/usr/games$ cd ..
mateo21@mateo21-desktop:/usr$
```

Et hop là, on est revenu au dossier parent ! (figure suivante.)



Retour au dossier parent

Si on avait voulu reculer de deux dossiers parents, on aurait écrit `../..` (« reviens en arrière, puis reviens en arrière »). Cela nous aurait ramené à la racine :

```
mateo21@mateo21-desktop:/usr/games$ cd ../..
mateo21@mateo21-desktop:/$
```

Eh mais en fait, il y a plusieurs façons d'aller dans un dossier, non ? Tout à l'heure, on est allé à la racine en tapant `cd /` , et maintenant en tapant `cd ../..`
C'est quoi cette affaire ?

Il y a en fait deux façons de changer de dossier : en indiquant un **chemin relatif**, ou en indiquant un **chemin absolu**.

Les chemins relatifs

Un chemin relatif est un chemin qui dépend du dossier dans lequel vous vous trouvez. Tout à l'heure, on est allé dans le sous-dossier `games` de `/usr` en tapant juste son nom :

```
mateo21@mateo21-desktop:/usr$ cd games
```

En faisant cela, on utilise un chemin relatif, c'est-à-dire relatif au dossier actuel. Quand on met juste le nom d'un dossier comme ici, cela indique que l'on veut aller dans un sous-dossier.

Si on fait `cd games` depuis la racine, ça va planter :

```
mateo21@mateo21-desktop:/$ cd games
bash: cd: games: Aucun fichier ou répertoire de ce type
```

Je crois que le message d'erreur est assez clair : il n'y a aucun dossier `games` dans `/` .

Pour se rendre dans `games` , il faut d'abord indiquer le dossier qui le contient (`usr`) :

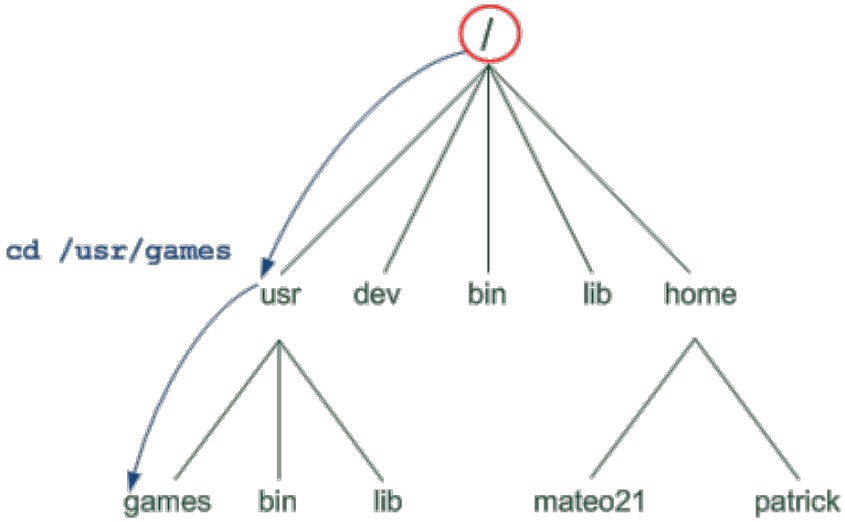
```
mateo21@mateo21-desktop:/$ cd usr/games
mateo21@mateo21-desktop:/usr/games$
```

Les chemins absolus

Contrairement aux chemins relatifs, les chemins absolus fonctionnent quel que soit le dossier dans lequel on se trouve.

Un chemin absolu est facile à reconnaître : il commence toujours par la racine (`/`). Vous devez ensuite faire la liste des dossiers dans lesquels vous voulez entrer. Par exemple, supposons que je sois dans `/home/mateo21` et que je souhaite aller dans `/usr/games` . Avec un chemin absolu :

```
mateo21@mateo21-desktop:~$ cd /usr/games
mateo21@mateo21-desktop:/usr/games$
```



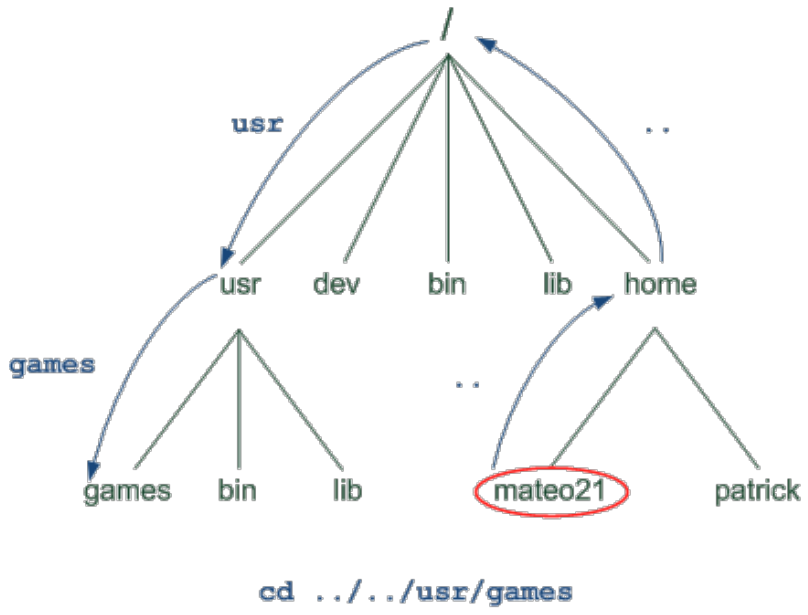
Chemin absolu

Le schéma suivante montre bien qu'on **part** de la racine `/` pour indiquer où on veut aller.

Si on avait voulu faire la même chose à coup de chemin relatif, il aurait fallu écrire :

```
mateo21@mateo21-desktop:~$ cd ../../usr/games/
mateo21@mateo21-desktop:/usr/games$
```

Ce qui signifie « **reviens en arrière (donc dans `/home`) puis reviens en arrière (donc dans `/`), puis va en avant dans `usr` , puis va en avant dans `games`** ». Voyez en figure suivante.



Chemin relatif

Ici, comme c'est un chemin relatif, on **part** du dossier dans lequel on se trouve (ici, c'est `/home/mateo21`) et on indique à la machine le chemin à suivre à partir de là pour aller dans le dossier qu'on veut.

Un chemin absolu est donc facile à reconnaître, car on part toujours de la racine `/` .
Un chemin relatif peut aussi s'avérer très pratique et plus court (ça dépend des cas).
Ce sera à vous de choisir à chaque fois comment vous voulez écrire votre chemin. Vous avez le choix.

Retour au répertoire `home`

Si vous voulez retourner dans votre répertoire `home` personnel, plusieurs solutions s'offrent à vous.

- **La brutale** : il suffit d'écrire le chemin absolu en entier. Cela donne :

```
mateo21@mateo21-desktop:/usr/games$ cd /home/mateo21/
mateo21@mateo21-desktop:~$
```

La maligne : plus court et plus pratique, vous pouvez utiliser l'alias `~` qui signifie la même chose. Cela donne :

```
mateo21@mateo21-desktop:/usr/games$ cd ~
mateo21@mateo21-desktop:~$
```

La super maligne : si vous ne mettez aucun paramètre à la commande `cd` , ça vous ramène aussi dans votre répertoire personnel.

```
mateo21@mateo21-desktop:/usr/games$ cd
mateo21@mateo21-desktop:~$
```

Autocomplétion du chemin

Cette astuce est vitale ; si vous ne vous en servez pas, vous passez à côté d'une des plus importantes astuces de la console.

L'idée est simple : taper `cd /usr/games/trucbidule` c'est bien, mais c'est parfois un peu long de tout écrire. On a la flemme. Vous avez la flemme. Oui, vous. Alors vous allez justement demander à l'ordinateur de compléter le chemin tout seul !

L'autocomplétion de chemin fonctionne de la même manière que l'autocomplétion de commande qu'on a vue dans le chapitre précédent : avec la touche `Tab` (Tabulation). Faites le test avec moi. Commencez par vous placer dans `/usr` :

```
mateo21@mateo21-desktop:~$ cd /usr
mateo21@mateo21-desktop:/usr$
```

Tapez ensuite juste `cd ga` , puis appuyez sur `Tab` . C'est magique, le nom du dossier a été automatiquement complété !

```
mateo21@mateo21-desktop:/usr$ cd games/
```

Revenez maintenant dans `/usr` (en faisant `cd ..` par exemple) et essayez de taper juste `cd l`, puis faites `Tab`. Rien ne se passe : cela signifie que l'ordinateur n'a pas trouvé de dossier qui corresponde au début de votre recherche, ou alors qu'il y en plusieurs qui commencent par « l ». Faites à nouveau `Tab` :

```
mateo21@mateo21-desktop:/usr$ cd l
lib/   local/
mateo21@mateo21-desktop:/usr$ cd l
```

On vient de vous donner la liste des dossiers qui commencent par « l » ! Cela signifie qu'il faut préciser votre recherche parce que sinon, l'ordinateur ne peut pas deviner dans quel dossier vous voulez entrer. Ça tombe bien, la commande a été réécrite en dessous, vous n'avez plus qu'à ajouter une lettre plus précise : par exemple « o » pour que Linux devine que vous voulez aller dans le dossier `local`. Tapez donc « o », puis à nouveau `Tab`, et le nom sera complété !

```
mateo21@mateo21-desktop:/usr$ cd local/
```

Faites des tests pour vous entraîner à utiliser l'autocomplétion, c'est vraiment très important. Vous allez voir, c'est intuitif et vraiment pratique !

du: taille occupée par les dossiers

La commande « `du` », pour *Disk Usage* (utilisation du disque) vous donne des informations sur la taille qu'occupent les dossiers sur votre disque.

Placez-vous pour commencer dans `/usr/games`, et tapez `du` :

```
mateo21@mateo21-desktop:~$ cd /usr/games
mateo21@mateo21-desktop:/usr/games$ du
5732    .
```

Comme ce dossier ne contient pas de sous-dossier, la commande `du` nous renvoie la taille totale que font les fichiers contenus dans le dossier.

Si vous allez dans votre `home` en revanche, celui-ci contient beaucoup de sous-dossiers. Dans ce cas, la commande `du` va renvoyer la taille de chacun des sous-dossiers, puis la taille totale à la fin (« . ») :

```
mateo21@mateo21-desktop:/usr/games$ cd
mateo21@mateo21-desktop:~$ du
400     ./.Trash
4       ./.themes
32      ./.mozilla-thunderbird/8vyw6pqo.default/Mail/Local Folders
36      ./.mozilla-thunderbird/8vyw6pqo.default/Mail
12      ./.mozilla-thunderbird/8vyw6pqo.default/US
...
...
264     ./.jedit/jars
```

```
4       ./.jedit/macros
380     ./.jedit/settings-backup
856     ./.jedit
82484   .
```

J'ai volontairement coupé la liste car elle est très longue.

`-h` : la taille pour les humains

Ce qui est bien, c'est que les commandes reprennent souvent les mêmes paramètres. Ainsi, on avait vu `-h` pour `ls`, eh bien ce paramètre est le même pour avoir des tailles « humaines » avec `du` !

```
mateo21@mateo21-desktop:~$ du -h
400K    ./.Trash
4,0K    ./.themes
32K     ./.mozilla-thunderbird/8vyw6pqo.default/Mail/Local Folders
36K     ./.mozilla-thunderbird/8vyw6pqo.default/Mail
12K     ./.mozilla-thunderbird/8vyw6pqo.default/US
...
...
264K    ./.jedit/jars
4,0K    ./.jedit/macros
380K    ./.jedit/settings-backup
856K    ./.jedit
81M     .
```

Mon dossier `home` prend donc 81 Mo d'espace disque, son sous-dossier caché `.jedit` prend 856 Ko, etc.

`-a` : afficher la taille des dossiers ET des fichiers

Par défaut, `du` n'affiche que la taille des dossiers. Pour avoir aussi la taille des fichiers qu'ils contiennent, rajoutez l'option `-a` (*all*) :

```
mateo21@mateo21-desktop:~$ du -ah
...
8,0K    ./.jedit/settings-backup/abbrevs~5~
24K     ./.jedit/settings-backup/history~1~
8,0K    ./.jedit/settings-backup/abbrevs~4~
380K    ./.jedit/settings-backup
44K     ./.jedit/pluginMgr-Cached.xml.gz
856K    ./.jedit
81M     .
```

`-s` : avoir juste le grand total

Pour n'avoir que l'espace total occupé par le dossier et donc ne pas afficher le détail des sous-dossiers, utilisez `-s` (que je combine à `-h` pour plus de lisibilité) :

```
mateo21@mateo21-desktop:~$ du -sh
81M     .
```

Je vois ainsi que mon dossier `home` fait 81 Mo (rappel : le symbole point « . » signifie « **le dossier actuel** »).

En résumé

- Sous Linux, tout est organisé sous forme de fichiers. Il n'y a pas de lecteur du type `C:` comme sous Windows.
- Les dossiers sont imbriqués entre eux à partir du dossier parent principal `/` . On l'appelle la **racine**.
- Le dossier dans lequel les utilisateurs stockent leurs documents est `/home` . Si votre login est `patrick` , alors votre dossier personnel sera `/home/patrick` .
- La commande `pwd` permet de savoir en console dans quel dossier on se situe.
- `ls` affiche la liste des fichiers présents dans le dossier actuel.
- `cd` permet de changer de dossier.

Que pensez-vous de ce cours ?

INDIQUER QUE CE CHAPITRE N'EST PAS TERMINÉ

ENTRER UNE COMMANDE

MANIPULER LES FICHIERS

Le professeur

Mathieu Nebra

Entrepreneur à plein temps, auteur à plein temps et co-fondateur d'OpenClassrooms :o)

Découvrez aussi ce cours en...

Livre

PDF

Expérience de formation

Forum

Blog

Presse

OPPORTUNITÉS

Nous rejoindre

Devenir mentor

Devenir coach carrière

AIDE

FAQ

POUR LES ENTREPRISES

Former et recruter

EN PLUS

Boutique

Mentions légales

Conditions générales d'utilisation

Politique de protection des données personnelles

Cookies

Accessibilité