

3

Systèmes d'exploitation I

Extrait du programme



THÈME : ARCHITECTURES MATÉRIELLES ET SYSTÈMES D'EXPLOITATION

Contenus : Systèmes d'exploitation

Capacités attendus : Identifier les fonctions d'un système d'exploitation.

Utiliser les commandes de base en ligne de commande.

Gérer les droits et permissions d'accès aux fichiers.

Commentaires : Les différences entre systèmes d'exploitation libres et propriétaires sont évoquées.

Les élèves utilisent un système d'exploitation libre.

Il ne s'agit pas d'une étude théorique des systèmes d'exploitation.



Vous pouvez retrouver le cours complet à l'adresse suivante :

<https://github.com/NaturelEtChaud/NSI-Premiere/tree/main/03%20Syst%C3%A8me%20d'exploitation%20I>

I. Les exposés en lien avec ce cours



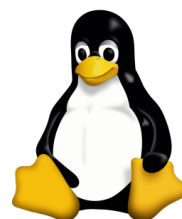
Richard Stallman (16 mars 1953-...)



Linus Torvalds (28 décembre 1969-...)



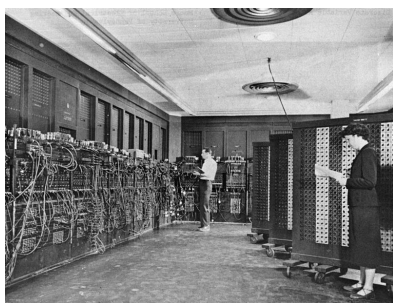
GNU



Tux

Sources Wikipédia.

II. Une petite histoire du système d'exploitation



AVANT 1950 : Les premiers calculateurs, tels que l'ENIAC (1945) n'ont pas de système d'exploitation. Ils n'exécutent qu'un seul programme à la fois. Ce dernier est saisi en re-cablant physiquement l'ordinateur.



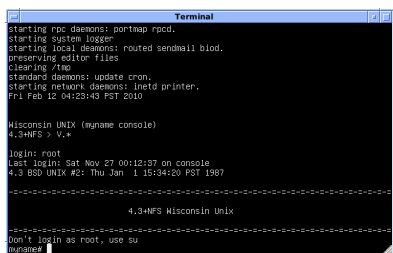
1956 : GM-NAA I/O est l'un des premiers systèmes d'exploitation. C'est un programme qui fonctionne sur un ordinateur IBM 704 et dont le rôle est d'exécuter en séquence des programmes utilisateurs stockés sur des cartes perforées. Il propose aussi des routines (ou fonctions) pour accéder simplement aux périphériques d'entrée/sortie.



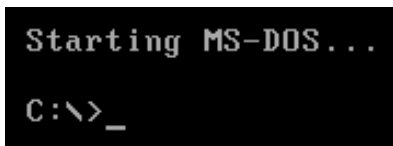
1967 : MultiCS (*Multiplexed Information and Computing Service*) développé à Bell Labs et au MIT est l'un des premiers systèmes d'exploitation à temps partagé : plusieurs programmes pouvaient s'exécuter « en même temps ». Il a largement influencé les systèmes d'exploitation modernes.



1970-1990 : Unix, développé à Bell Labs, est l'un des premiers systèmes d'exploitation multi-tâches et multi-utilisateurs. Les premières versions du système sont écrites en assembleurs, puis dans le langage C, nouvellement créé pour cela.



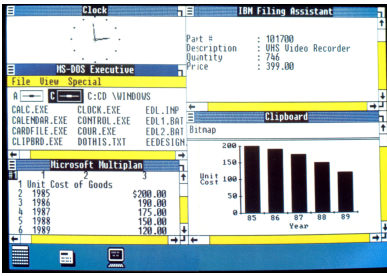
1978-1990 : Une variante d'Unix voit le jour et prospère : la Berkley Software Distribution ou BSD.



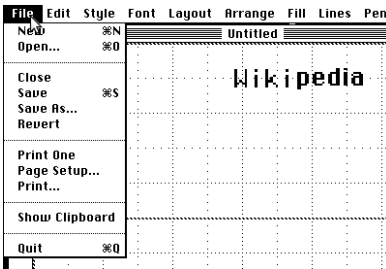
1960-1980 : Les ordinateurs du constructeur IBM utilisent diverses variantes de systèmes d'exploitation, différents Unix. Ils utilisent en particulier à partir des années 1980 le système MS-DOS de Microsoft. Ce système est conçu pour fonctionner uniquement sur processeur x86 d'Intel.



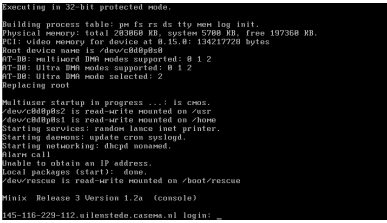
1980-1990 : Le succès de l'architecture x86 d'Intel et de l'ordinateur personnel (PC, par opposition aux calculateurs) installent MS-DOS comme le système d'exploitation principal pour les particuliers et de nombreux domaines d'activités.



1990-PRÉSENT : L'éditeur Microsoft développe un système graphique au dessus de MS-DOS, qui deviendra ensuite un système d'exploitation à part entière, le système Windows. Il devient, à la suite de MS-DOS, le système d'exploitation le plus utilisé sur PC.



1984-2001 : En parallèle de Microsoft, l'entreprise Apple commercialise des ordinateurs personnels basés sur l'architecture Motorola puis PowerPC. Les machines sont équipées du système d'exploitation Mac OS, un système graphique.



1991 : Linus Torvalds, alors étudiant à l'Université d'Helsinki, souhaite modifier le système d'exploitation MINIX (une variante d'Unix pour processeur Intel). Bien que les sources de MINIX soient disponibles, la licence logicielle ne permet pas la diffusion des modifications. Linus Torvalds décide donc de créer son propre système d'exploitation. Il le diffuse sous la licence libre GNU GPL. Le système d'exploitation GNU/Linux est né.



1991-PRÉSENT : Le système Linux connaît une adoption rapide, principalement due à sa diffusion sous licence libre. Il est particulièrement répandu comme système d'exploitation pour serveurs et machines de calculs ou de stockage.



2001-PRÉSENT : Apple crée un nouveau système d'exploitation, basé sur un système BSD, le système macOS (d'abord appelé OS X).



2007-PRÉSENT : Apple reprend le cœur du système macOS et en développe une version pour les téléphones portables qu'il commercialise.








2008-PRÉSENT : Google diffuse le système d'exploitation Android pour téléphones mobiles. Ce système utilise le noyau du système Linux, auquel sont ajoutés des programmes et bibliothèques non libres.

Source Texte : Numériques et sciences informatiques - 30 leçons avec exercices corrigés, Thibaut Balabonski, Sylvain Conchon, Jean-Christophe Filliâtre, Kim Nguyen, Edition Ellipses

Sources Images : Wikipédia

III. Les principaux systèmes d'exploitation

	Microsoft Windows 1.0 - 3.x - 95 - 98 - Me - NT - 2000 - XP - 2003 - Vista – 2008 – 7 – 8 – 10
	GNU/Linux Debian - Fedora - Gentoo - Mandriva - Red Hat – Slackware - SuSE - Ubuntu
	Mac OS Système 5 - 6 - 7 - 8 – 9 Mac OS X -.0 -.1 -.2 -.3 -.4 -.5- .6-.7- .8- .9- .10 macOS Sierra - High Sierra
	BSD FreeBSD - NetBSD - OpenBSD - DragonFly BSD - PC-BSD
	Autres Android, AmigaOS - BeOS - DOS - Inferno - LynxOS - Haiku - OS/2 - QNX - Solaris - UNIX MVS - OS/360 - OS/390 - OS/400 - Plan 9 - ReactOS - VMS - ZETA - FreeDOS

Source : Fonction d'un SE, Audrey Queudet, formation NSI 2018-2019 à Nantes

IV. Mais monsieur, un système d'exploitation, qu'est-ce que c'est ?

Commençons par voir ce qui se passe avec le système d'exploitation le plus utilisé au monde, Windows¹.

a) Un superlogiciel

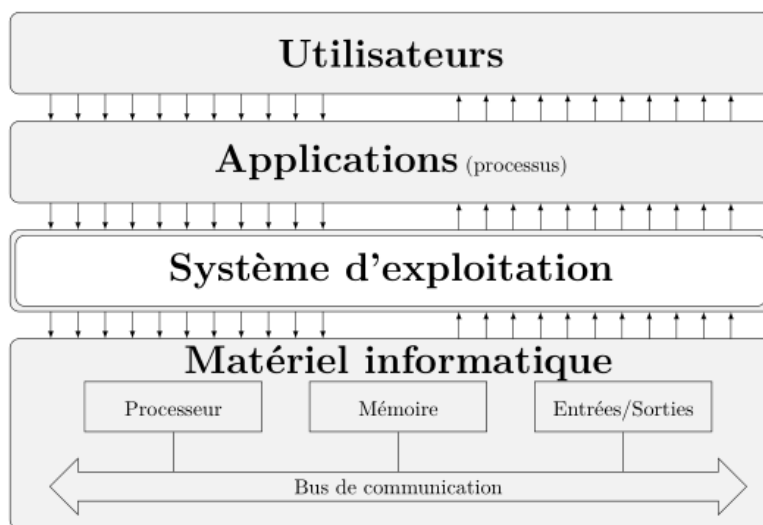
Lorsqu'on allume notre ordinateur, Windows se lance presque en premier. Si on regarde bien, on peut constater que c'est quelque chose d'autre qui s'affiche à l'écran au cours des toutes premières secondes. Cet « autre chose » est ce qu'on appelle l'écran de **boot**. C'est la **carte mère** qui affiche l'écran de boot. La carte mère est le composant fondamental de tout ordinateur, c'est elle qui fait travailler le processeur, les disques durs, le lecteur de CD-ROM, etc...

C'est seulement une fois que Windows est chargé que vous pouvez enfin utiliser vos programmes : jeux, Internet, logiciels de dessin, ...

Mais pourquoi faut-il que Windows se charge d'abord ? Pourquoi ne pourrait-on pas lancer des jeux dès le démarrage de l'ordinateur ?

Parce que l'ordinateur a besoin d'une sorte de « superlogiciel » qui soit le chef d'orchestre. C'est lui qui doit gérer la mémoire de votre ordinateur, la répartir entre tous les programmes. Il fait le lien entre votre matériel (carte graphique, mémoire, imprimante) et vos logiciels.

Ce « superlogiciel » s'appelle le **système d'exploitation**. Windows est donc un système d'exploitation que l'on abrège souvent en **OS** car un système d'exploitation se dit **Operating System** en anglais.



b) Le système d'exploitation Linux

Au début des années 1980, Microsoft vient de sortir son premier OS, MS-DOS, mais ce dernier est encore loin d'être abouti. Celui qui était considéré comme le meilleur s'appelait Unix. Il était beaucoup plus puissant mais aussi plus compliqué à utiliser, ce qui explique pourquoi seuls les informaticiens professionnels l'utilisaient. Unix était payant et devenait de plus en plus cher. En 1984, Richard Stallman a voulu réagir en proposant une alternative gratuite : le projet **GNU** était né. GNU ne devait pas seulement être un OS gratuit, il devait également être « libre ».

Un programme libre est la plupart du temps un programme gratuit. Mais c'est aussi un programme qu'on a le droit de copier, modifier, redistribuer, ... C'est une véritable idéologie en informatique : des gens pensent qu'il vaut mieux donner le code source des programmes que l'on crée car cela permet le partage des connaissances et aide l'informatique à évoluer plus vite. Le slogan du monde du Libre pourrait être : « **L'union fait la force** ».

En 1991, Linus Torvalds entreprend de créer sur son temps libre son propre système d'exploitation, Linux².

Ces deux projets étaient complémentaires : tandis que Richard créait les programmes de base (programme de copie de fichier, de suppression de fichier, éditeur de texte), Linus s'était lancé dans la création du cœur d'un système d'exploitation : le **noyau**.

1. En janvier 2007, il était installé sur plus de 95% des ordinateurs personnels. *Source Wikipédia*

2. Ce système a pris le nom de Linux, en référence au nom de son créateur (Linux est la contraction de Linus et Unix).

Le projet **GNU** (programmes libres) et **Linux** (noyau d'OS) ont fusionné pour créer **GNU/Linux**³.

Les programmes sous Linux ont d'énormes avantages :

- ils sont presque tous gratuits ;
- les logiciels sont souvent mis à jour et ce toujours gratuitement ;
- certains de ces logiciels sont meilleurs que ceux que l'on trouve sous Windows. D'ailleurs, certains n'existent même pas sous Windows ou ont d'abord été développés sous Linux avant de passer sous Windows.

Pour simplifier la vie des utilisateurs et leur permettre de faire un choix, différentes distributions de Linux ont été créées. Voici ce qui peut différer d'une distribution à l'autre :

- l'installation : elle peut être très simplifiée comme très compliquée ;
- la gestion de l'installation des programmes. Si elle est bien faite et centralisée, elle peut rendre l'installation de nouveaux logiciels plus simple que sous Windows ;
- les programmes préinstallés sur l'ordinateur.

En fait, une distribution est en quelque sorte l'emballage de Linux. Le cœur, lui, reste le même sur toutes les distributions. Quelle que soit celle que vous installez, vous obtenez un Linux compatible avec les autres. Certaines distributions sont juste plus ou moins faciles à prendre en main.

Voici quelques-unes des différentes distributions de Linux :

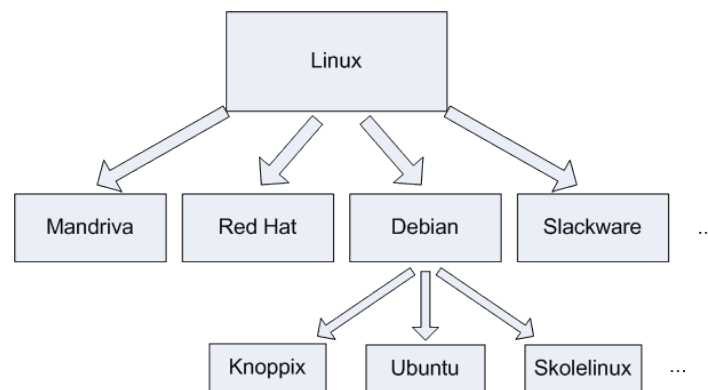
- Slackware : une des plus anciennes distributions de Linux. Elle existe encore aujourd'hui ;
- Mandriva : éditée par une entreprise française, elle se veut simple d'utilisation ;
- Red Hat : éditée par une entreprise américaine, cette distribution est célèbre et très répandue, notamment sur les serveurs ;
- SuSE : éditée par l'entreprise Novell ;
- Debian : la seule distribution qui soit gérée par des développeurs indépendants plutôt que par une entreprise. C'est une des distributions les plus populaires.

Debian est donc la seule distribution éditée par des particuliers bénévoles à travers le monde. Un autre gros avantage de **Debian** est le gestionnaire de paquets **apt-get**. C'est un programme qui gère tous les logiciels installés et qui permet de les désinstaller très facilement. D'autre part, tous les logiciels sont centralisés en un même endroit, ce qui fait que l'on n'a pas à parcourir tout le Web pour retrouver un programme. En fait, on a juste à indiquer le nom du logiciel que l'on désire : Debian ira le télécharger et l'installer tout seul.

Debian a tellement de succès que de nombreuses distributions se sont basées dessus :

- Knoppix ;
- Skolelinux ;
- Ubuntu ;
- ...

Ce sont donc des distributions de distributions.



Source Image : NSI, Numérique & Sciences Informatique, Mickaël Barraud, pdf

Sources Texte et Image : Reprenez le contrôle à l'aide de Linux!, OpenClassRooms

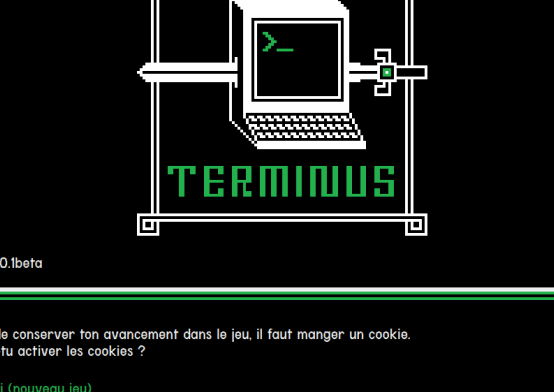
<https://openclassrooms.com/fr/courses/43538-reprenez-le-contrôle-a-laide-de-linux>

3. Théoriquement, on doit donc parler de GNU/Linux mais par abus de langage, on dit juste souvent Linux.

Source Image : Wikipédia



On peut y jouer en ligne à l'adresse suivante : <http://ujuridec.free.fr/Terminus/>.



version : 0.1beta

Afin de conserver ton avancement dans le jeu, il faut manger un cookie.
Veux-tu activer les cookies ?

- ▶ oui (nouveau jeu)
- ▶ non (pas de sauvegarde)

VII. A la découverte de notre système d'exploitation

Après vous être logger en mode élève, répondez aux questions suivantes :

- Quelle est la distribution employée ?
- Quelle heure est-il ?
- Où se trouve le navigateur Mozilla Firefox ?
- Où se trouve le logiciel Mu ?
- Où se trouve le terminal ou shell ou console ?

Vous voyez, nous ne sommes pas sur un Windows mais on s'y retrouve malgré tout (à part peut-être la dernière question).

VIII. L'invite de commandes

Ouvrez le terminal soit comme fait précédemment, soit avec les touches **Ctrl + Shift + T**.

```
1 .....@.....:~$
```

Ce que vous voyez là est ce qu'on appelle l'invite de commande. C'est un message qui vous invite à rentrer une commande en vous donnant par la même occasion une foule d'informations. Cette invite s'affiche avant chaque commande que vous tapez. Décortiquons :

- : le premier élément est votre pseudonyme. C'est le pseudo sous lequel vous vous êtes loggés. On peut créer plusieurs comptes utilisateurs sous Linux. Il est en général conseillé d'en générer un par personne susceptible d'utiliser l'ordinateur (un pour chaque membre de la famille, par exemple).
- @ : ce symbole n'indique rien de particulier. C'est le symbole « at » qui signifie « chez ». Si on lit l'invite de gauche à droite, on doit donc comprendre « chez ».
- : ça, c'est le nom de l'ordinateur sur lequel vous êtes en train de travailler. Dans mon cas il s'appelle, mais j'aurais pu lui attribuer n'importe quel nom lors de l'installation.
Si vous suivez toujours, la ligne d'invite de commandes se lit donc « chez ». En d'autres termes, je suis identifié en tant que sur la machine
- : ce symbole ne veut rien dire de spécial, c'est un séparateur.
- ~ : c'est le dossier dans lequel vous vous trouvez actuellement. Vous pouvez naviguer de dossier en dossier dans la console et il est très utile qu'on vous rappelle systématiquement où vous vous trouvez avant chaque commande.
Pour information, le symbole ~ signifie que vous êtes dans votre dossier personnel, ce qu'on appelle le « home » sous Linux ; c'est l'équivalent du dossier « Mes documents » de Windows. Nous étudierons plus en détail le fonctionnement des dossiers sous Linux.
- \$: ce dernier symbole est très important ; il indique votre niveau d'autorisation sur la machine. Il peut prendre deux formes différentes :
\$: signifie que vous êtes en train d'utiliser un compte utilisateur « normal », avec des droits limités (il ne peut pas modifier les fichiers système les plus importants). Votre compte élève est donc un compte normal avec des droits limités ;
: signifie que vous êtes en mode superutilisateur, c'est-à-dire que vous êtes connectés sous le pseudonyme « root ». Le root est l'utilisateur maître qui a le droit de tout faire sur sa machine (même de la détruire!).

Comme vous le voyez, une fois qu'on parle la même langue que l'invite de commandes, on comprend ce qu'elle veut dire !
« Bonjour et bienvenue, vous êtes sur la machine Vous vous trouvez actuellement dans votre dossier home et possédez des droits utilisateur limités. »

Comme un peu tout sous Linux, l'invite de commandes est totalement paramétrable. Vous pouvez la raccourcir si vous trouvez qu'elle est trop longue, ou la rallonger si vous trouvez qu'elle ne donne pas assez d'informations. Vous pouvez en théorie mettre vraiment tout ce que vous voulez dans l'invite, comme par exemple l'heure actuelle .

On travaille dans la console en tapant ce qu'on appelle des commandes. Ces dernières étant nombreuses, vous ne pourrez jamais toutes les connaître... et ce n'est pas le but : le but, c'est que vous sachiez vous servir par cœur de la plupart des commandes courantes et, pour les moins courantes, que vous soyez capables d'apprendre à vous en servir en lisant leur manuel d'utilisation.

Le manuel d'utilisation est la véritable bible de tous les linuxiens. Vous verrez rapidement qu'ils ne jurent que par ça. Pourquoi ? Parce que c'est tout simplement un outil de référence, là où l'on peut trouver la réponse à TOUTES ses questions pour peu qu'on sache lire le manuel et qu'on prenne la peine de le faire.

Source Texte : Reprenez le contrôle à l'aide de Linux!, OpenClassRooms

<https://openclassrooms.com/fr/courses/43538-reprenez-le-controle-a-laide-de-linux>

IX. Quelques commandes pour commencer

Tapez **date** puis appuyez sur la touche Entrée du clavier. Le résultat devrait ressembler à cela :

```
1 nsi-16@NSI-P07:~$ date
2 lundi 16 septembre 2019, 09:39:51 (UTC+0200)
```

La première ligne contient l'invite de commandes suivie de la commande que j'ai tapée. La seconde ligne est la réponse de l'ordinateur à cette commande.

Essayons une toute autre commande : tapez **ls**. C'est l'abréviation de « list », qui signifie « lister les fichiers et dossiers du répertoire actuel ».

```
1 nsi-16@NSI-P07:~$ ls
2 anaconda3      Bureau          Modèles         pt              StartPrep
3 animaux        Documents       mu_code         Public          Téléchargements
4 Arduino        Images          Musique          sketchbook      Vidéos
```

Cela signifie que le répertoire actuel est constitué de 12 dossiers dont **Bureau**, **Documents** et **Images**. En général, le système colore les éléments pour que l'on puisse distinguer facilement les dossiers des fichiers. Si vous n'avez aucune réponse, c'est que vous êtes dans un dossier qui ne contient ni fichier ni dossier.

Voilà, c'est aussi simple que cela. Une commande est constituée d'un mot et ne contient aucun espace. Dans des cas très simples comme ceux que l'on vient de voir, il suffit juste de taper la commande pour avoir une réponse ; mais dans la quasi-totalité des cas on peut (et parfois on DOIT) rentrer des options, qu'on appelle paramètres.

Source Texte : Reprenez le contrôle à l'aide de Linux!, OpenClassRooms

<https://openclassrooms.com/fr/courses/43538-reprenez-le-controle-a-laide-de-linux>

X. Les paramètres

Les paramètres sont des options que l'on écrit à la suite de la commande. La commande et les paramètres sont séparés par un espace, comme ceci :

```
1 nsi-16@NSI-P07:~$ commande parametres
```

Les paramètres peuvent eux-mêmes contenir des espaces, des lettres, des chiffres... un peu de tout, en fait. Il n'y a pas de règle véritable sur la forme des paramètres, mais heureusement les programmeurs ont adopté une sorte de convention pour que l'on puisse reconnaître les différents types de paramètres.

a) Les paramètres courts (une lettre)

Les paramètres les plus courants sont constitués d'une seule lettre précédée d'un tiret. Par exemple :

```
1 nsi-16@NSI-P07:~$ commande -d
```

Si on doit donner plusieurs paramètres, on peut faire comme ceci :

```
1 nsi-16@NSI-P07:~$ commande -d -a -U -h
```

Ou, plus court :

```
1 nsi-16@NSI-P07:~$ commande -daUh
```

Attention à la casse des paramètres (majuscules / minuscules) ! Si vous écrivez **-u**, cela n'a en général pas du tout le même sens que **-U** !

Faisons un essai avec la commande `ls` et rajoutons-lui le paramètre **-a** (en minuscule) :

```
1 nsi-16@NSI-P07:~$ ls -a
2 .                .dbus             .ipython          pt
3 ..               .dmrc             .java             Public
4 anaconda3        Documents         .jupyter          .pyzo
5 animaux          .filius           .linuxmint        sketchbook
6 Arduino           .fontconfig       .local            StartPrep
7 .arduino15        .gconf            Modèles           .sudo_as_admin_successful
8 .bash_history     .gnome            .mozilla          Téléchargements
9 .bash_logout     .gnupg            mu_code           .themes
10 .bashrc          .gtkrc-2.0        Musique           Vidéos
11 Bureau           .gtkrc-xcfe       .oracle_jre_usage .Xauthority
12 .cache           .gvfs             .packettracer     .xsession-errors
13 .cinnamon        .ICEauthority     .pki              .xsession-errors.old
14 .conda           .icons            .processing
15 .config          Images            .profile
```

Cela affiche tout le contenu du dossier, même les fichiers cachés. Un **fichier caché** sous Linux est un fichier qui commence par un point. Normalement, si vous vous trouvez dans votre répertoire **home**, vous devriez avoir une bonne flopée de fichiers cachés. Ce sont en général des fichiers de configuration de programmes.

b) Les paramètres longs (plusieurs lettres)

Les paramètres constitués de plusieurs lettres sont précédés de deux tirets, comme ceci :

```
1 nsi-16@NSI-P07:~$ commande --parametre
```

Cette fois, pas le choix : si vous voulez mettre plusieurs paramètres longs, il faudra ajouter un espace entre chacun d'eux :

```
1 nsi-16@NSI-P07:~$ commande --parametre1 --parametre2
```

On peut aussi combiner les paramètres longs et les paramètres courts dans une commande :

```
1 nsi-16@NSI-P07:~$ commande -daUh --autreparametre
```

Il y a parfois deux écritures possibles pour un paramètre de commande : une version courte et une version longue. Cela permet de vous laisser le choix selon que vous préférez l'une ou l'autre.

Testons cela sur la commande `ls` avec le paramètre `-all`, qui signifie « tout » en anglais :

```
1 nsi-16@NSI-P07:~$ ls --all
2 .                .dbus             .ipython          pt
3 ..               .dmrc             .java             Public
4 anaconda3        Documents         .jupyter          .pyzo
5 animaux          .filius           .linuxmint        sketchbook
6 Arduino           .fontconfig       .local            StartPrep
7 .arduino15       .gconf            Modèles           .sudo_as_admin_successful
8 .bash_history    .gnome            .mozilla          Téléchargements
9 .bash_logout     .gnupg            mu_code           .themes
10 .bashrc          .gtkrc-2.0        Musique           Vidéos
11 Bureau          .gtkrc-xcfe       .oracle_jre_usage .Xauthority
12 .cache           .gvfs             .packettracer     .xsession-errors
13 .cinnamon        .ICEauthority     .pki              .xsession-errors.old
14 .conda           .icons            .processing
15 .config          Images            .profile
```

Comme vous le voyez, `-all` est un synonyme de `-a`.

c) Les valeurs des paramètres

Certains paramètres nécessitent que vous les complétiez avec une valeur. Cela fonctionne différemment selon que vous travaillez avec un paramètre long ou avec un paramètre court.

- Avec un paramètre court :

```
1 nsi-16@NSI-P07:~$ commande -p 14
```

cela indique que l'on associe la valeur 14 au paramètre `p`. Avec ce genre de technique, on peut par exemple faire comprendre à l'ordinateur : « Je veux voir la liste de tous les fichiers de plus de 14 Mo ».

- Si c'est un paramètre long, on fait en général comme ceci :

```
1 nsi-16@NSI-P07:~$ commande --parametre=14
```

Le résultat sera le même, il est juste plus lisible mais aussi plus long à écrire.

d) Les autres paramètres

Il n'y a pas de règle absolue au niveau des paramètres et vous en rencontrerez sûrement qui fonctionnent différemment. Heureusement, les conventions que je viens de vous donner sont valables dans la grande majorité des cas, ce qui devrait vous permettre de vous repérer.

Certains paramètres sont donc un peu différents et dépendent vraiment des commandes. Par exemple avec `ls`, si on ajoute le nom d'un dossier (ou sous-dossier), cela affichera le contenu de ce dossier au lieu du contenu du dossier courant :

```
1 nsi-16@NSI-P07:~$ ls Bureau
2 Photo  Programmation  Réseau-Internet  Son-vidéo
```

e) Retrouver une commande

Linux propose tellement de commandes différentes qu'il est facile de s'y perdre et d'en oublier une, mais ce n'est heureusement pas un drame. En effet, Linux vous propose toute une série de façons de retrouver une commande que vous avez oubliée.

- Autocomplétion de commande

Le premier truc à connaître, c'est l'autocomplétion de commande. Prenons la commande **date** par exemple : vous êtes un peu tête en l'air et vous ne savez plus comment elle s'écrit. Par contre, vous êtes sûrs des premières lettres de la commande.

Tapez juste « da » dans la console, puis tapez deux fois sur la touche **Tabulation** située à gauche de votre clavier. Le résultat sera le suivant :

```
1 nsi-16@NSI-P07:~$ da
2 dash          dask-remote      dask-ssh         dask-worker
3 dask-mpi       dask-scheduler  dask-submit     date
4 nsi-16@NSI-P07:~$ da
```

En tapant deux fois sur **Tabulation**, vous avez demandé à l'ordinateur la liste des commandes qui commencent par « da ». On vous a répondu **dash** et **date**. Il y a donc 2 commandes qui commencent par « da », et vous venez de retrouver celle que vous cherchiez, c'est-à-dire **date**.

Bien sympathique, l'ordinateur a réécrit l'invite de commandes en dessous ainsi que le début de la commande que vous aviez tapée. Vous n'avez plus qu'à compléter avec les lettres « te » qui manquent et à taper **Entrée**, et ce sera bon.

- L'autocomplétion

Plus sympa encore, s'il n'y a qu'un seul résultat correspondant à votre recherche, l'ordinateur complètera avec les lettres qui manquent et vous n'aurez plus qu'à taper sur **Entrée**. Par exemple, il n'y a qu'une commande qui commence par « dat ». Tapez donc « dat » dans la console, puis appuyez une seule fois sur **Tabulation**. La commande se complète comme par magie.

- Trop de commaaaaandes !

Parfois, il y a trop de commandes correspondant à votre recherche. Ne rentrez que le **l** faites deux fois **Tabulation**⁴. Ainsi on demande de faire la liste de toutes les commandes disponibles sur votre ordinateur commençant par un « l ».

```
1 nsi-16@NSI-P07:~$ l
2 Display all 2173 possibilities? (y or n)
```

Il y a 160 commandes disponibles sur l'ordinateur. Pensez aussi que plus on installera de programmes, plus on aura de commandes utilisables, n'espérez donc pas toutes les connaître, de nouveaux programmes sortent tous les jours.

A cette question, vous pouvez répondre « y » (yes) et la liste s'affichera page par page. Quelques raccourcis à connaître quand une liste s'affiche page par page :

tapez **Espace** pour passer à la page suivante ;

tapez **Entrée** pour aller à la ligne suivante ;

tapez **q** pour arrêter la liste.

Si vous répondez « n » (no), il ne se passera rien.

- L'historique des commandes

On a très souvent besoin de retrouver une commande que l'on a tapée il y a cinq minutes (ou même cinq secondes). Parfois c'est parce qu'on a oublié la commande, mais c'est souvent aussi parce qu'on a vraiment la flemme de réécrire nous-mêmes la commande en entier.

Ce raccourci vaut de l'or : appuyez sur la flèche directionnelle **Haut** ; vous verrez apparaître la dernière commande que vous avez tapée.

Si vous appuyez de nouveau sur la flèche directionnelle **Haut**, vous verrez l'avant-dernière commande, puis l'avant-avant-dernière, etc...

Si vous appuyez sur la flèche directionnelle **Bas**, vous reviendrez aux commandes les plus récentes.

4. Sous certaines distributions, cette manipulation ne fonctionne pas. Il faut au moins entrer un caractère pour avoir une réponse.

Si vous voulez « remonter » très loin en arrière dans l'historique de vos commandes, pas la peine de taper cent fois sur la flèche directionnelle **Haut** comme des forcenés. Il existe la commande **history** qui vous rappelle l'historique des commandes :

```
1 152  date
2 153  ls
3 154  ls -a
4 155  ls --all
5 156  history
```

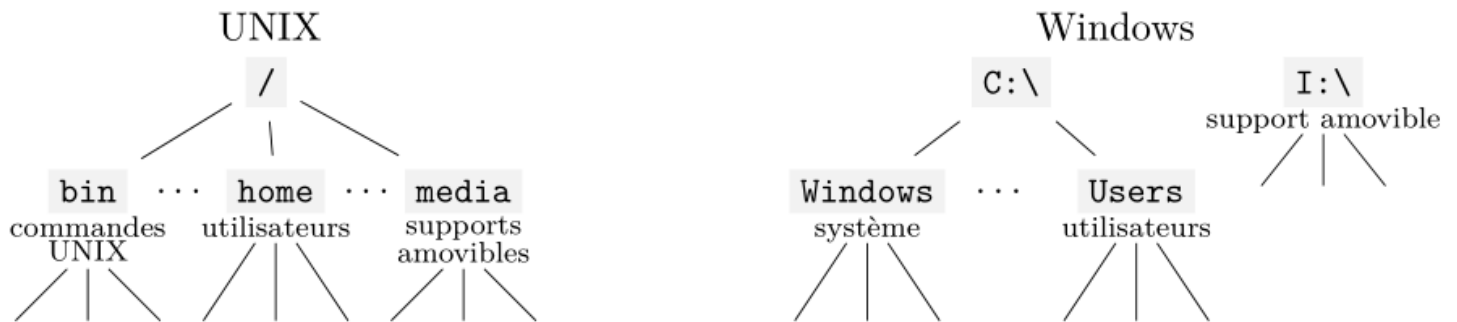
La dernière commande tapée sera toujours **history**, forcément.

Vous remarquerez que les commandes sont numérotées : ainsi, on peut savoir que **date** est la 152ème commande que on a tapée dans le terminal.

Source Texte : Reprenez le contrôle à l'aide de Linux!, OpenClassRooms

<https://openclassrooms.com/fr/courses/43538-reprenez-le-contrôle-a-laide-de-linux>

XI. La structure des dossiers et fichiers



a) 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:**, **I:**, ... 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 fait-on pour savoir si le dossier dans lequel on est appartient au premier disque dur, au second disque dur, au lecteur CD... ? C'est ce qu'on pourrait croire au premier abord, mais en fait c'est juste une autre façon de penser la chose.

b) 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**, ...), les images, les vidéos, mais aussi les programmes. Bref, 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 et même un peu déroutante, mais pas de panique, vous allez vous y faire.

c) 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, `I :` est la racine de votre clef USB (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 /.

d) 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. 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**, ...

e) 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, les dossiers sont complètement différents. Et l'on ne risque pas de trouver de dossier qui s'appelle Windows !

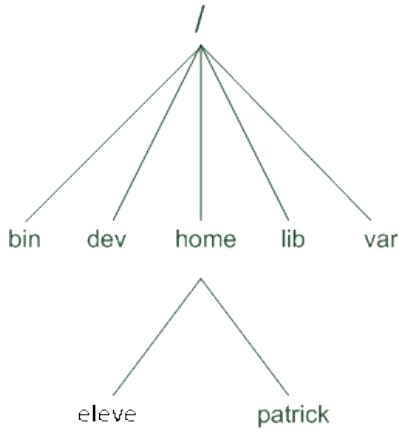
Voici la liste des dossiers les plus courants que l'on retrouve à chaque fois à la racine de Linux :

- **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. 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. 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 votre cas le dossier personnel se trouve dans `/home/eleve/`. S'il y avait un autre utilisateur (appelons-le Patrick) sur l'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, **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. Ne retenez pas tout ça. C'est juste pour vous donner une idée de ce que contiennent les dossiers à la racine de Linux.

f) Schéma résumé de l'architecture

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



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

g) pwd : afficher le dossier actuel

Le nombre de dossiers et de fichiers présents 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 !

Lorsque vous ouvrez la console pour la première fois, Linux vous place dans votre dossier personnel, votre **home**. En l'occurrence le dossier dans lequel on est placé sera **/home/nsi-16**.

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

```
1 nsi-16@NSI-P07:~$
```

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

```
1 nsi-16@NSI-P07:~$ pwd
2 /home/nsi-16
```

h) ls : lister les fichiers et dossiers

ls est une des toutes premières commandes que nous avons essayées. 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 :

```

1 nsi-16@NSI-P07:~$ ls
2 anaconda3      Bureau      Modèles      pt           StartPrep
3 animaux        Documents   mu_code       Public        Téléchargements
4 Arduino         Images      Musique       sketchbook    Vidéos

```

Ubuntu active la coloration des fichiers et dossiers par défaut, vous devriez donc voir des couleurs. 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.

- **-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 :

```

1 nsi-16@NSI-P07:~$ ls -a
2 .                .dbus           .ipython        pt
3 ..               .dmrc           .java           Public
4 anaconda3        Documents       .jupyter        .pyzo
5 animaux          .filius         .linuxmint      sketchbook
6 Arduino          .fontconfig     .local          StartPrep
7 .arduino15       .gconf          Modèles         .sudo_as_admin_successful
8 .bash_history    .gnome          .mozilla        Téléchargements
9 .bash_logout     .gnupg          mu_code         .themes
10 .bashrc          .gtkrc-2.0      Musique         Vidéos
11 Bureau          .gtkrc-xcfe     .oracle_jre_usage .Xauthority
12 .cache           .gvfs           .packettracer   .xsession-errors
13 .cinnamon        .ICEauthority   .pki            .xsession-errors.old
14 .conda           .icons          .processing
15 .config          Images          .profile

```

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/nsi-16**, « .. » représente donc le dossier **/home**.

- Le paramètre **-A** (avec une majuscule) 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.

```

1 nsi-16@NSI-P07:~$ ls -F
2 anaconda3/      Bureau/         Modèles/        pt/             StartPrep*
3 animaux/        Documents/      mu_code/        Public/         Téléchargements/
4 Arduino/        Images/         Musique/        sketchbook/     Vidéos/

```

Grâce à ça on peut voir que tous les éléments sont des dossiers, sauf **StartPrep**.

- **-l** : liste détaillée

Le paramètre **-l** est un des plus utiles. Il affiche une liste détaillant chaque élément du dossier :

```
1 nsi-16@NSI-P07:~$ ls -l
2 total 60
3 drwxr-xr-x 69 root root 4096 oct. 8 13:46 anaconda3
4 drwxr-xr-x 6 nsi-16 nsi-16 4096 nov. 13 11:23 animaux
5 drwxrwxr-x 3 nsi-16 nsi-16 4096 oct. 8 13:46 Arduino
6 drwxr-xr-x 6 nsi-16 nsi-16 4096 sept. 22 2021 Bureau
7 drwxr-xr-x 4 nsi-16 nsi-16 4096 sept. 21 2021 Documents
8 drwxr-xr-x 2 nsi-16 nsi-16 4096 oct. 8 13:46 Images
9 drwxr-xr-x 2 nsi-16 nsi-16 4096 oct. 8 13:46 Modèles
10 drwxr-xr-x 6 nsi-16 nsi-16 4096 oct. 8 13:46 mu_code
11 drwxr-xr-x 2 nsi-16 nsi-16 4096 oct. 8 13:46 Musique
12 drwxr-xr-x 6 nsi-16 nsi-16 4096 oct. 8 13:46 pt
13 drwxr-xr-x 2 nsi-16 nsi-16 4096 oct. 8 13:46 Public
14 drwxrwxr-x 7 nsi-16 nsi-16 4096 oct. 8 13:46 sketchbook
15 -rwxr-xr-x 1 nsi-16 nsi-16 1119 sept. 19 14:09 StartPrep
16 drwxr-xr-x 5 nsi-16 nsi-16 4096 oct. 8 13:46 Téléchargements
17 drwxr-xr-x 2 nsi-16 nsi-16 4096 oct. 8 13:46 Vidéos
```

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

- droits sur le fichier (on fera un chapitre entier pour expliquer comment fonctionnent les droits sous Linux) ;
- nombre de liens physiques (cela ne nous intéresse pas ici) ;
- nom de la personne propriétaire du fichier (là, c'est vous !). Si le fichier avait été créé par quelqu'un d'autre, par exemple Patrick, on aurait vu son nom à la place ;
- 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 ;
- taille du fichier, en octets ;
- date de dernière modification ;
- nom du fichier (ou dossier).

- **-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. 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) :

```
1 nsi-16@NSI-P07:~/Exemples$ ls -lh
2 total 60
3 drwxr-xr-x 69 root root 4,0K oct. 8 13:46 anaconda3
4 drwxr-xr-x 6 nsi-16 nsi-16 4,0K nov. 13 11:23 animaux
5 drwxrwxr-x 3 nsi-16 nsi-16 4,0K oct. 8 13:46 Arduino
6 drwxr-xr-x 6 nsi-16 nsi-16 4,0K sept. 22 2021 Bureau
7 drwxr-xr-x 4 nsi-16 nsi-16 4,0K sept. 21 2021 Documents
8 drwxr-xr-x 2 nsi-16 nsi-16 4,0K oct. 8 13:46 Images
9 drwxr-xr-x 2 nsi-16 nsi-16 4,0K oct. 8 13:46 Modèles
10 drwxr-xr-x 6 nsi-16 nsi-16 4,0K oct. 8 13:46 mu_code
11 drwxr-xr-x 2 nsi-16 nsi-16 4,0K oct. 8 13:46 Musique
12 drwxr-xr-x 6 nsi-16 nsi-16 4,0K oct. 8 13:46 pt
13 drwxr-xr-x 2 nsi-16 nsi-16 4,0K oct. 8 13:46 Public
14 drwxrwxr-x 7 nsi-16 nsi-16 4,0K oct. 8 13:46 sketchbook
15 -rwxr-xr-x 1 nsi-16 nsi-16 1,1K sept. 19 14:09 StartPrep
16 drwxr-xr-x 5 nsi-16 nsi-16 4,0K oct. 8 13:46 Téléchargements
17 drwxr-xr-x 2 nsi-16 nsi-16 4,0K oct. 8 13:46 Vidéos
```

- **-t** : trier par date de dernière modification

-t permet en 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 :

```
1 nsi-16@NSI-P07:~$ ls -lt
2 total 60
3 drwxr-xr-x 6 nsi-16 nsi-16 4096 sept. 22 2021 Bureau
4 drwxr-xr-x 4 nsi-16 nsi-16 4096 sept. 21 2021 Documents
5 drwxr-xr-x 6 nsi-16 nsi-16 4096 nov. 13 11:23 animaux
6 drwxr-xr-x 6 nsi-16 nsi-16 4096 oct. 8 13:46 pt
7 drwxr-xr-x 6 nsi-16 nsi-16 4096 oct. 8 13:46 mu_code
8 drwxrwxr-x 3 nsi-16 nsi-16 4096 oct. 8 13:46 Arduino
9 drwxrwxr-x 7 nsi-16 nsi-16 4096 oct. 8 13:46 sketchbook
10 drwxr-xr-x 69 root root 4096 oct. 8 13:46 anaconda3
11 drwxr-xr-x 2 nsi-16 nsi-16 4096 oct. 8 13:46 Images
12 drwxr-xr-x 2 nsi-16 nsi-16 4096 oct. 8 13:46 Modèles
13 drwxr-xr-x 2 nsi-16 nsi-16 4096 oct. 8 13:46 Musique
14 drwxr-xr-x 2 nsi-16 nsi-16 4096 oct. 8 13:46 Public
15 drwxr-xr-x 2 nsi-16 nsi-16 4096 oct. 8 13:46 Vidéos
16 drwxr-xr-x 5 nsi-16 nsi-16 4096 oct. 8 13:46 Téléchargements
17 -rwxr-xr-x 1 nsi-16 nsi-16 1119 sept. 19 14:09 StartPrep
```

De toute évidence, le dernier fichier (ici, c'est un dossier) modifié est « Bureau ».

Et si on combine un peu tous les paramètres qu'obtiendra-t-on ?

```
1 nsi-16@NSI-P07:~$ ls -larth
```

```
.....
.....
.....
.....
.....
.....
.....
.....
.....
```

i) **cd** : changement de dossier

La commande que nous allons étudier ici s'appelle **cd**, abréviation de Change Directory (changer de dossier). C'est une commande très importante. 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 /** :

```
1 nsi-16@NSI-P07:~$ cd /
2 nsi-16@NSI-P07:/$ pwd
3 /
```

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 **/**.

On peut maintenant lister les fichiers et dossiers contenus dans **/** :

```
1 nsi-16@NSI-P07:/$ ls -F
2 bin/      dev/      initrd/      lib/          mnt/      root/      sys/      var/
3 boot/     etc/      initrd.img@  lost+found/  opt/      sbin/      tmp/      vmlinuz@
4 cdrom@    home/     initrd.img.old@ media/        proc/     srv/      usr/      vmlinuz.old@
```

Vous y retrouvez un grand nombre de dossiers décrits plus tôt.

Allons dans le sous-dossier **usr** :

```
1 nsi-16@NSI-P07:/$ cd usr
```

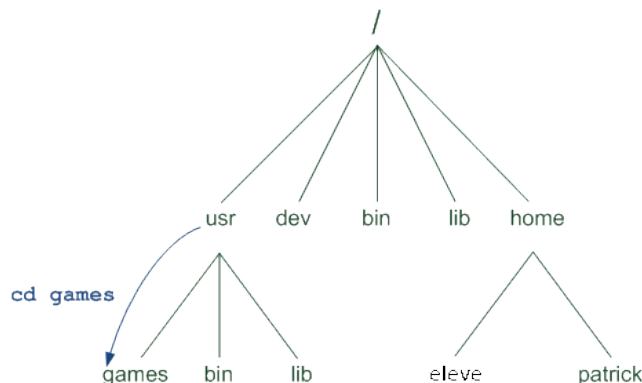
Voyons voir ce qu'il y a là-dedans :

```
1 nsi-16@NSI-P07:/usr$ ls -F
2 bin/  games/  include/  lib/  libexec/  local/  sbin/  share/  src/
```

Allons voir ce qu'il y a comme jeux :

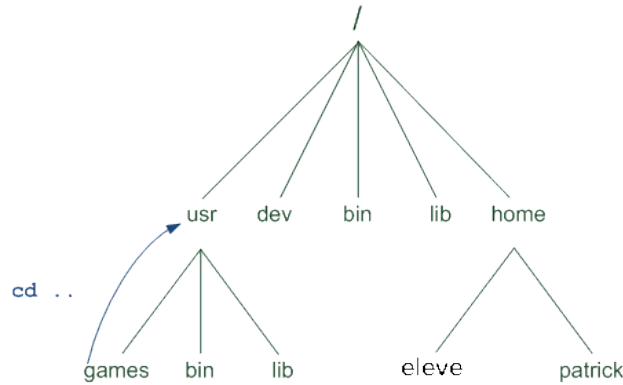
```
1 nsi-16@NSI-P07:/usr$ cd games
2 espdiff*
3 nsi-16@NSI-P07:/usr/games$
```

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



Supposons maintenant que j'aie envie de revenir au dossier précédent, aussi appelé dossier parent, c'est-à-dire **/usr**. Comment fait-on ? Il faut utiliser les deux points comme ceci :

```
1 nsi-16@NSI-P07:/usr/games$ cd ..
2 nsi-16@NSI-P07:/usr$
```



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 :

```
1 nsi-16@NSI-P07:/usr/games$ cd ../../
2 nsi-16@NSI-P07:/$
```

Il y a donc 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. En tapant :

```
1 nsi-16@NSI-P07:/usr$ cd games
```

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 :

```
1 nsi-16@NSI-P07:/ $ cd games
2 bash: cd: games: Aucun fichier ou dossier de ce type
```

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

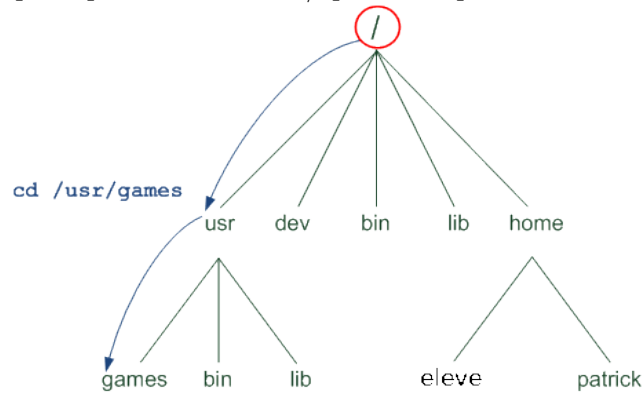
```
1 nsi-16@NSI-P07:/ $ cd usr/games
2 nsi-16@NSI-P07:/usr/games$
```

- 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 l'on soit dans **/home/eleve** et que l'on souhaite aller dans **/usr/games** avec un chemin absolu :

```
1 nsi-16@NSI-P07:~$ cd /usr/games
2 nsi-16@NSI-P07:/usr/games$
```

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 :

```
1 nsi-16@NSI-P07:~$ cd ../../usr/games/
2 nsi-16@NSI-P07:/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` ». Ici, comme c'est un chemin relatif, on part du dossier dans lequel on se trouve (ici, c'est `/home/eve`) 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 :

```
1 nsi-16@NSI-P07:/usr/games$ cd /home/nsi-16/
2 nsi-16@NSI-P07:~$
```

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

```
1 nsi-16@NSI-P07:/usr/games$ cd ~
2 nsi-16@NSI-P07:~$
```

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

```
1 nsi-16@NSI-P07:/usr/games$ cd
2 nsi-16@NSI-P07:~$
```

- Autocomplétion du chemin

L'idée est simple : taper `cd /usr/games/truchidule` est parfois un peu long de tout écrire. On va 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 précédemment : avec la touche **Tabulation** :

```
1 nsi-16@NSI-P07:~$ cd /usr
2 nsi-16@NSI-P07:/usr$
```


Tapez ensuite juste **cd ga**, puis appuyez sur **Tabulation**. Le nom du dossier a été automatiquement complété !

```
1 nsi-16@NSI-P07:/usr$ cd games/
```

Revenez maintenant dans **/usr** (en faisant **cd ..** par exemple) et essayez de taper juste **cd l**, puis faites **Tabulation**. 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 a plusieurs qui commencent par « l ». Faites à nouveau **Tabulation** :

```
1 nsi-16@NSI-P07:/usr$ cd l
2 lib/    libexec/ local/
3 nsi-16@NSI-P07:/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 **Tabulation**, et le nom sera complété !

```
1 nsi-16@NSI-P07:/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 !

Source Image : NSI, Numérique & Sciences Informatique, Mickaël Barraud, pdf

Sources Texte et Image : Reprenez le contrôle à l'aide de Linux!, OpenClassRooms

<https://openclassrooms.com/fr/courses/43538-reprenez-le-contrôle-a-laide-de-linux>

Annexe

Commande	Utilisation	Paramètres
cd
date
ls
pwd