

Administration de systèmes Linux

2017-04-11

Nicolas Michel

Table of Contents

1. Comment est né UNIX ?	1
1.1. Les premiers pas	1
2. Les distributions Linux	2
3. Le shell	5
3.1. Qu'est-ce que le shell ?	5
3.2. Quel shell utilisons-nous ?	5
3.3. Comment accéder au shell ?	6
3.4. Qu'est ce que l'invite de commande ?	7
3.5. La complétion automatique	8
3.6. Linux est sensible à la casse !	9
3.7. Le copier-coller	9
4. Les commandes de base	9
4.1. Structure d'une commande	9
4.2. cd – Change Directory	10
4.3. ls – List	10
4.4. pwd – Print Working Directory	13
4.5. Changer d'utilisateur	13
4.6. Arrêter le système	15

1. Comment est né UNIX ?

1.1. Les premiers pas

Pour comprendre comment et pourquoi UNIX est né, il faut d'abord essayer de se remettre dans le contexte et comprendre où en était l'informatique dans les années 50 et 60. Il s'agissait d'énormes machines prenant la place d'une très grande pièce et pesant plusieurs tonnes. Bien entendu leur taille diminuait d'années en années, mais cela restait des machines très coûteuses, uniquement accessibles aux états, aux universités, aux plus grandes entreprises, et qui demandait des réparations et un entretien quotidien. Ces machines-là ressemblaient fort aux machines industrielles: constituées d'un grand nombre de parties mécaniques. Elles étaient lentes par rapport à aujourd'hui. Pour économiser le temps machine, au lieu d'encoder directement les programmes informatiques sur un clavier relié à l'ordinateur, les informaticiens encodaient les programmes sur des cartes perforées de 80 colonnes, 8 bits par colonne.

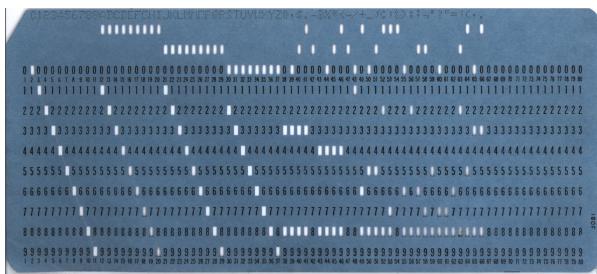


Figure 1. Une carte perforée 80 colonnes d'IBM d'un des types les plus utilisés au xx^e siècle.

Une carte ou un ensemble de cartes perforées, selon la complexité, représentaient un programme, qui était chargé en mémoire, exécuté, pour finalement fournir le résultat via une imprimante, un écran ou une autre carte perforée pour une utilisation future. La chose importante à retenir est que ces ordinateurs ne pouvaient exécuter qu'un seul programme à la fois. Pour optimiser le rendement, les ingénieurs préparaient des batchs [1: batch qui signifie en anglais: lot, paquet.] , c'est à dire une compilation d'un ensemble de carte perforées (plusieurs programmes) sur une bande de plus grande longueur. De cette manière, l'ordinateur pouvait charger et exécuter programme après programme rapidement, sans intervention humaine pour insérer chaque nouvelle carte perforée. Il faut également savoir que s'il y avait la moindre erreur dans un programme, son exécution était stoppée. Il fallait donc corriger, puis réintégrer la nouvelle version dans le prochain batch. En 1957, un ingénieur d'IBM, Robert William Bemer [2: Robert William Bemer a été un ingénieur de son temps fort actif: outre le concept du time-sharing, il a créé le code ASCII, il a participé à l'élaboration du langage COBOL, il a été le premier en 1971 à anticiper le bug de l'an 2000.] , publie pour la première fois le concept du temps-partagé: plutôt que d'exécuter un seul programme à la fois, il propose qu'un grand nombre de terminaux pourraient être connectés au même moment au serveur central (au main frame), et que chaque utilisateur puisse avoir la possibilité d'interagir directement et d'entrer son code au clavier. Il a pris pour hypothèse que le grand nombre d'utilisateurs simultané aurait pour effet de diminuer les périodes idle (au repos) du processeur central, tout en procurant aux utilisateurs plus de flexibilité car ils ne doivent pas attendre que leur programme soit intégré dans un batch. Il peuvent l'écrire et l'exécuter en temps réel. D'un point de vue technique, la difficulté réside dans la création d'un système d'exploitation capable de gérer l'exécution d'un grand nombre de programmes simultanément au moyen d'un seul processeur. Il faut pouvoir interrompre et reprendre l'exécution pendant une courte durée des

différents processus afin de les traiter en alternance et de donner l'impression aux utilisateurs que leur programme est exécuté en temps réel. Cela implique notamment de sauvegarder l'état de chaque programme en mémoire, composant qui était très coûteux à l'époque.

2. Les distributions Linux

Dans le langage courant, on parle de "Linux" pour désigner le système d'exploitation complet. Techniquelement, Linux est uniquement le noyau (ou kernel en anglais) développé par le finlandais Linus Torvalds en 1991.



Figure 2. Linus Torvalds, créateur du noyau Linux

Le kernel est la pièce fondamentale de tout système d'exploitation : Windows, Mac OS, Linux, *BSD... Il gère la communication entre les logiciels et le matériel, la planification des processus (les programmes qui sont lancés), et bien entendu, il initialise et pilote le matériel. Le cœur du kernel Linux sur les distributions courantes (sans les modules) fait environ 4 Mo. Un kernel seul, bien qu'il soit la pièce maîtresse, ne fait pas l'entièreté d'un système d'exploitation. Il y a beaucoup d'autres logiciels nécessaires. Par exemple, toutes les commandes de base (cd, ls, cat et autres) sont chacune des mini-logiciels. Toutes ces commandes de bases font parties d'un autre projet open-source, appelé GNU [3: <http://fr.wikipedia.org/wiki/GNU>] (GNU is Not Unix). Puis il y a encore l'environnement de bureau (l'interface graphique). Il y en a plusieurs disponibles pour Linux.

Table 1. Interfaces graphiques les plus connues

Nom	Description
Unity	l'interface d'Ubuntu, basé sur Gnome
Gnome	reprend quelques idées de Mac OS, c'est à dire une interface simple et épurée
KDE	reprend quelques idées de Windows, notamment le menu démarrer, la liste des programmes. Son but n'est pas d'être simple ou épuré, mais de proposer le plus de fonctionnalités possibles
Cinnamon	un dérivé de Gnome créé par la distribution Linux Mint
XFCE	un bureau minimaliste ressemblant à Gnome, mais moins gourmand en CPU et mémoire, permettant ainsi de le faire tourner sur des machines plus modestes
LXDE	encore plus léger que xfce, l'interface semble aujourd'hui un peu obsolète par rapport à ce qu'on connaît, mais il tourne sur de très vieilles configurations

Il y a encore beaucoup d'autres logiciels nécessaires comme la gestion du démarrage des services, le login screen , l'assistant d'installation, et. Une distribution est donc un package constitué de ces nombreux logiciels qui, ensemble, forment un système d'exploitation complet utilisable par vous et moi. Chaque distribution fait des choix dans les programmes employés pour remplir les divers besoins. Ces choix sont basés sur des objectifs et une philosophie différent. Les distributions peuvent également être regroupées en familles car certaines distributions sont dérivées d'autre.

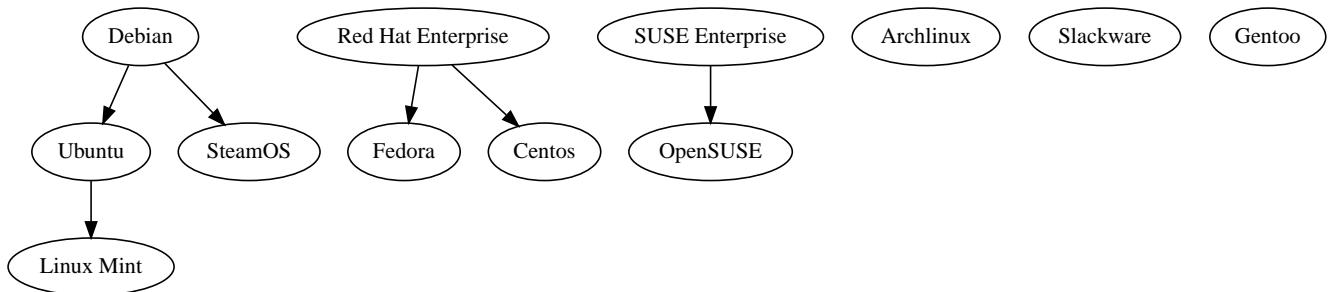


Figure 3. Liste des distributions Linux les plus courantes.

Table 2. Description des principales distributions

Nom	Parenté	Bureau par défaut	Description
Debian	-	Gnome	Distribution entièrement gérée par une communauté de bénévoles qui a vu le jour en 1996. Elle a créé le système de paquet apt-get et beaucoup de distributions actuelles sont basées dessus.
Ubuntu	Debian	Unity	Probablement la plus connue des distributions. Créée par Marc Shuttleworth, un sud-africain, cette distribution a pour but d'être la plus facile d'accès et de masquer au maximum la complexité technique. Elle est basée sur Debian.
Linux Mint	Ubuntu	Cinnamon	Basé sur Ubuntu, Linux Mint est une distribution récente qui pense pouvoir faire mieux qu'Ubuntu. Ils ont donc fait d'autres choix. Ils ont notamment ré-écrit l'environnement de bureau pour donner naissance à Cinnamon.
SteamOS	Debian	Gnome	Basé sur Debian, cette distribution a été créée en 2013 par Valve Software dans le but de créer une console de jeux basée sur Steam.
Red Hat Enterprise Linux	Fedora	Gnome	Red Hat est la société américaine bien connue pour sa distribution Linux orientée entreprise. Le contrat oblige l'utilisateur à payer la licence dès qu'une Red Hat est installé, et bénéficie ainsi du support. On ne peut pas installer une Red Hat sans payer. Cette distribution vise la stabilité avant tout, afin de faire tourner des serveurs. Elle n'a donc pas toujours toutes les dernières nouveautés.
Centos	Red Hat Enterprise Linux	Gnome	Centos est un projet maintenu par des bénévoles, qui reprennent le code source de Red Hat, et le re-package. Une Centos est donc un Red Hat « re-brandé », et entièrement gratuit.

Nom	Parenté	Bureau par défaut	Description
Fedora	Red Hat Enterprise Linux	Gnome	Fedora est la distribution « laboratoire » de Red Hat. Elle est mise à jour tous les 6 mois avec les dernières versions de tous les logiciels. Les meilleurs logiciels dans leur meilleure version sont retenus pour la création des Red Hat Linux.
Archlinux	-	-	C'est une distribution qui suit le principe KISS [4: Keep It Simple, Stupid : http://fr.wikipedia.org/wiki/Principe_KISS], et est à conseiller pour toute personne curieuse qui veut en connaître plus sur le fonctionnement interne de Linux. En effet, après l'installation, on se retrouve avec un linux en command-line minimal. Tout le reste doit être installé et configuré à la main. La documentation sur leur site web est exemplaire.
Slackware	-	KDE	Slackware fait figure de dinosaure car elle a été la première distribution réalisée à grande échelle, en 1992 (le kernel Linux date de 1991). Elle est toujours maintenue par une seule personne : Patrick Volkerding. C'est une distribution extrêmement dépouillée et simple.
OpenSUSE	-	Gnome	Distribution gérée par une communauté ainsi que des employés de la société allemande SUSE. Elle vise la facilité d'administration via des outils comme yast, un menu global permettant de configurer l'entièreté du système. Elle est donc plus accessible pour des gens qui seraient moins experts en Linux. Le danger étant bien entendu de ne pas comprendre ce qui se passe réellement derrière, et donc une difficulté de troubleshooting.
Suse Enterprise	OpenSUSE	Gnome	Version entreprise et payante d'OpenSUSE.
Gentoo	-	KDE	Le concept de Gentoo est que toute application est compilée avant installation alors que toutes autres distributions téléchargent une version pré-compilée. Selon la puissance de votre machine et la grosseur du logiciel, la compilation peut prendre plus ou moins de temps. L'idée c'est qu'en compilant sur votre machine, on pourra exploiter toutes les instructions spécifiques du CPU alors que lorsque c'est pré-compilé, on doit utiliser un set d'instructions standard afin que cela fonctionne sur toutes les machines. C'est donc un léger gain de performance. Mais cela n'a vraiment de sens que pour des supercalculateurs, ou au contraire, des vieilles machines pour lesquels tout gain de performance est intéressant.

Vous trouverez une liste de toutes les distributions connues sur le site distrowatch [5: <http://distrowatch.com/>].

3. Le shell

3.1. Qu'est-ce que le shell ?

Le shell est un programme qui prend des commandes entrées via le clavier, les donne ensuite au système d'exploitation afin de les interpréter et de les exécuter. À l'époque de la création de UNIX, c'était le moyen le plus efficace permettant d'interagir avec le système d'exploitation. Depuis, les interfaces graphiques (GUI) ont fait leur apparition, permettant des manipulations de façon plus intuitive. Le shell n'est pas un outil intuitif. Il faut connaître une série de commandes qui vont permettre de travailler avec. Cependant, une fois maîtrisé, il est généralement plus puissant que l'interface graphique [6: Bien entendu, si votre travail concerne un document typiquement visuel comme une image, une vidéo, une page web, etc, l'interface graphique sera généralement plus efficace.] car il permet de combiner différents programmes en chaîne, afin de transformer petit à petit les données de départ. On verra cela en pratique avec le pipe : |. Il permet aussi de réaliser des scripts afin d'automatiser les tâches récurrentes. Un shell, bien qu'il ressemble de prime abord à un langage de programmation, comme le C, le java, le python, le perl, le ruby ... n'en est pas un ! La différence majeure est qu'un shell se contente d'exécuter différents programmes qui ont été préalablement écrit dans un langage de programmation (souvent en C pour les commandes de base) et qui sont déjà compilés en langage machine. Il apporte en plus quelques éléments syntaxiques supplémentaires pour faciliter leur combinaison : les boucles, les conditions, les redirections ... Un langage de programmation lui, va lire le code source, l'interpréter pour finalement le compiler en langage machine pour donner UN programme. Un shell ne produit jamais de lui-même de langage machine. Il en découle que pour réaliser une tâche simple, le shell sera plus efficace et demandera moins de lignes de codes, puisqu'il combine des programmes qui s'occupent déjà, via leur programmation, de gérer par eux-mêmes une série de cas de figure. Finalement, il y a même certaines manipulations orientées systèmes qui seraient vraiment fastidieuses à réaliser dans un langage de programmation. Prenons un exemple simple : la configuration des cartes réseaux. Si on voulait le réaliser dans un langage de programmation, il faudrait que notre programme parle directement au module kernel gérant le réseau (via les bibliothèques C réseaux), ce qui demanderait beaucoup de connaissances, de temps et de lignes de code, sachant que de mauvaises instructions données au noyau pourraient faire crasher la machine. En shell, il existe plusieurs commandes que nous verrons plus tard, qui prennent en argument la configuration souhaitée et parlent au noyau afin de l'appliquer.

Exemple de la commande ip qui permet d'assigner une IP à notre carte réseau principale

```
$ ip address add 192.168.1.10/24 dev eth0
```

En une ligne, nous avons réalisé une opération qui techniquement, n'est pas si simple que cela. Les limites du shell sont atteintes dès qu'il faut manipuler des structures de données complexes. On gagnera alors à utiliser perl ou python [7: Python a la cote et est fort utilisé par les entreprises web. Savez-vous que Dropbox est entièrement écrit en python ?].

3.2. Quel shell utilisons-nous ?

Le premier shell pour UNIX a été écrit en 1971 par Kenneth Thompson. En 1977, Stephen Bourne

écrit sh pour la version 7 de UNIX. D'autres shell ont ensuite vu le jour. Csh s'inspire de la syntaxe du C, est écrit par Bill Joy [8: Bill Joy est le fondateur de Sun Microsystem (Solaris). Avant de créer sa société, il a également été l'auteur de vi, de la première pile TCP/IP de UNIX, et csh.] . En 1983, ksh ou Korn Shell, est écrit par David Korn, qui inclut certaines améliorations de csh et ajoute certaines manipulations qui étaient plutôt retrouvées dans les langages de programmation ou les utilitaires spécialisés comme awk ou sed. C'est le shell par défaut sur AIX. De nos jours, sous Linux, le shell par défaut le plus courant est bash : Bourne Again Shell, écrit par la Free Software Fondation en 1988. Il reprend beaucoup d'améliorations des précédents shell.

Une façon de connaître le shell utilisé

```
$ echo $SHELL  
/bin/bash
```

Une autre façon de connaître le shell utilisé

```
$ echo $0  
-bash
```

3.3. Comment accéder au shell ?

Votre shell (bash) doit être exécuté dans un terminal. Qu'est-ce que c'est que ça? Un terminal est simplement le logiciel qui va gérer les entrées au clavier et gérer l'affichage sur votre écran. Le shell lui, ne s'occupe que d'interpréter les commandes constituées par les lettres que le terminal lui envoi, ainsi que de dialoguer avec le kernel. Historiquement, un terminal est un ordinateur simplifié qui est juste capable de gérer l'affichage et les entrées au clavier. Il était connecté via un câble série (COM1, COM2 ...) au serveur faisant tourner UNIX.



Figure 4. Port série

Aujourd'hui, s'il est toujours possible de se connecter au terminal via un port COM, et c'est bien utile dans les cas de dépannage les plus désespérés (carte graphique HS par exemple), on accède presque toujours au shell via un terminal logiciel, ce qu'on appelle un terminal virtuel. Il en existe beaucoup.

3.3.1. La console Linux

Il y a tout d'abord un terminal que vous trouverez sur toutes les distributions, en pur mode texte et fourni directement par le kernel (pas besoin d'interface graphique). Vous le trouverez en appuyant sur les touches CTRL+ALT+F1 à F7. Vous retrouverez votre interface graphique, si elle est lancée, sur l'une de ces combinaisons. Traditionnellement, la console graphique se trouve sur CTRL+ALT+F7, mais cela peut varier d'une distribution à l'autre. Sur Fedora 20 vous la trouverez sur CTRL+ALT+F1. On l'utilise en général pour le dépannage car l'absence (par défaut) de souris pour les copier-coller et sa faible mémoire d'historique rend son utilisation moins conviviale.



Pour visualiser les pages précédentes en console texte, utilisez SHIFT droit + Page UP et Page Down

3.3.2. Le terminal de l'environnement de bureau

C'est le terminal qu'on utilisera le plus souvent. Chaque environnement de bureau en propose un. Pour Gnome, c'est gnome-terminal. Pour KDE c'est konsole, etc. En général, depuis le menu applications, faites une recherche sur le mot-clé « terminal ». En cas de doute, l'icône devrait vous mettre sur la bonne voie.

3.3.3. Les terminaux graphiques tiers

Il y a toute sorte d'autres terminaux qui fonctionnent sur l'interface graphique et qui ne sont pas liés à une distribution ou un environnement de bureau. Vous avez par exemple xterm qui est l'un des plus anciens, rxvt qui a toujours ses aficionados, aterm, etc.

3.3.4. Les terminaux graphiques de type drop-down

Une série de terminaux se sont inspirés de la console du jeu vidéo Quake, qu'on pouvait faire apparaître au moyen de la touche « exposant 2 » (à gauche de la touche « chiffre 1 »). Ces terminaux sont très pratiques pour une utilisation quotidienne car en général, ils sont lancés automatiquement au démarrage de votre session graphique, et peuvent être appelés à tout moment en appuyant sur une touche définie (par exemple F12). Ils supportent en général les onglets.

Table 3. Quelques terminaux drop-down

Nom	Description
Guake	Fonctionne le mieux sur un bureau Gnome
Terra	Alternative à Guake. Également prévu pour Gnome.
Yakuake	Fonctionne le mieux sur un bureau KDE

3.4. Qu'est ce que l'invite de commande ?

Lorsque vous démarrez votre terminal, votre shell vous présente ce qu'on appelle une invite de commande (prompt en anglais). C'est à dire un petit texte qui reprend quelques informations, et un curseur clignotant vous indiquant à quel endroit va être affiché ce que vous allez taper au clavier.

L'invite de commande

```
titi@ma-tour:~$
```

Table 4. Explication des éléments de l'invite de commande

Nom	Description
titi	Indique l'utilisateur avec lequel vous êtes connecté
ma-tour	Le nom de l'ordinateur (hostname).
~	après les deux-points, le répertoire courant. Le tilde signifie la home directory [9: La home directory, ou répertoire utilisateur, est l'endroit où l'ensemble des fichiers appartenant à un utilisateur vont être stockés. C'est là qu'on va trouver les dossiers Documents, Images, Vidéos, Musiques, Téléchargements, etc. Mais aussi certains fichiers de configurations uniquement applicables à l'utilisateur.] de l'utilisateur courant (titi ici).
\$ ou #	\$ - lorsque vous êtes un utilisateur sans droit d'administration. # - lorsque vous êtes en root (super-user).



Signalons que l'arobase se dit "at" en anglais, ce qu'on peut traduire par « chez », « sur ». On pourrait donc construire une phrase avec l'ensemble de ces éléments: « je suis titi sur ma-tour dans ma home directory et je suis un utilisateur standard »

Vous devez finalement savoir que le prompt peut-être personnalisé. Vous n'aurez donc peut-être pas exactement le même, selon la distribution sur laquelle vous vous trouvez, mais généralement, cela y ressemble fort.

3.5. La complétion automatique

Dans votre shell, appuyez sur la touche tabulation pour compléter automatiquement votre commande ou votre chemin. Le shell va essayer de déterminer les différentes possibilités. S'il n'y en a qu'une, il écrit le nom de la commande en entier ou le chemin complet. S'il y a plusieurs choix, taper rapidement deux fois sur TAB pour obtenir les différentes possibilités.

"if" suivi d'un appui sur TAB - il ne se passe rien

```
$ if #suivi d'un appui sur TAB
```

"if" suivi de deux appui rapide sur TAB fait apparaître les différentes possibilités

```
$ if #suivi de deux appui sur TAB
if ifconfig ifdown ifnames ifquery ifup
```

si on ajoute un c après if

```
$ ifc #suivi d'une tabulation auto-complète la commande  
$ ifconfig
```

3.6. Linux est sensible à la casse !

Le mot casse remonte au temps de l'imprimerie mécanique et signifiait un casier en bois où l'on rangeait les caractères en plomb d'un même type [10: Casse (typographie), Wikipédia : [http://fr.wikipedia.org/wiki/Casse_\(typographie\)](http://fr.wikipedia.org/wiki/Casse_(typographie))]. En anglais on dit case sensitive (case avec un seul s). Par extension, on signifie aujourd'hui par ce terme que Linux (et les autres UNIX) font la différence entre un mot écrit en minuscule et en majuscule. Ifconfig ou IFconfig ou ifconfiG ne fonctionneront pas, car la commande est entièrement en minuscule : ifconfig. Il en va de même pour les répertoires et nom de fichiers !



Windows est insensible à la casse (ou case insensitive). Et c'est une source d'erreur fréquente lors de la découverte de Linux.

3.7. Le copier-coller

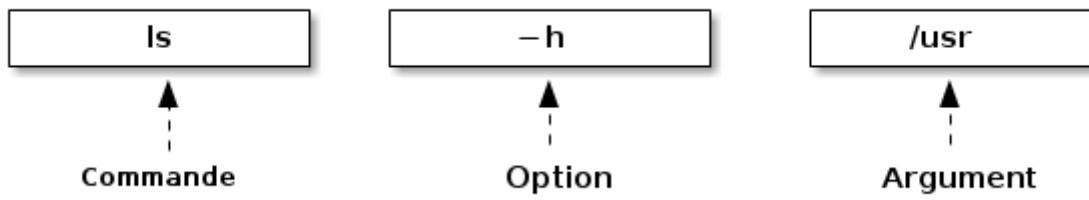
Dans un terminal graphique, lorsque vous effectuez une sélection avec la souris, il est automatiquement copié en mémoire. Vous pourrez le coller à tout moment en faisant : SHIFT+Insert. La plupart des terminaux permettent également d'utiliser le clic droit de la souris pour afficher un menu contextuel qui permet notamment le copier-coller, mais c'est plus fastidieux et donc généralement moins utilisé.

4. Les commandes de base

Une commande est une application, un exécutable, qui a été conçu pour réaliser une tâche spécifique. C'est un petit programme qui fait une chose simple. On l'utilise en tapant son nom au clavier, et on l'exécute en appuyant sur la touche "enter". Vous devrez apprendre par cœur l'ensemble des commandes de base.

4.1. Structure d'une commande

Très souvent, on donne à une commande un ou plusieurs arguments qui précisent sur quel objet la tâche doit être effectuée : un fichier, un répertoire, une ip ... Les arguments sont donnés à la suite de la commande, et s'il y en a plusieurs, ils sont séparés par un espace (s'il y a plusieurs espaces ou une tabulation, cela fonctionne aussi). On peut généralement donner des options aux commandes : -v, -m, -l ... Les options ne définissent pas l'objet sur lequel porte la tâche, mais modifient la manière dont elle va être effectuée. Le sens et l'effet d'une option est propre et spécifique à chaque commande, bien qu'on retrouve parfois certaines options avec le même sens d'une commande à l'autre.



Règles pour écrire une commande:

- Le premier "mot" est le nom de la commande
- Tout les autres mots qui viennent à suite sont soit des options soit des arguments
- Une option commence par un ou deux tirets
- Un argument est un mot qui ne commence pas par un ou plusieurs tirets
- Tous ces éléments sont séparé par un ou plusieurs espaces

4.2. cd – Change Directory

cd est l'acronyme de Change Directory (changer de répertoire). C'est la commande qui vous permettra de naviguer dans le système de fichier et de changer votre répertoire courant. Donnez en argument l'emplacement vers lequel vous voulez vous rendre.

cd / vous amène à la racine du système de fichier

```
$ cd /
```

! Notez que contrairement à Windows, le séparateur de chemin est le slash / et non le backslash \

4.3. ls – List

ls liste les répertoires et les fichiers de votre arborescence.

4.3.1. Sans argument

Sans argument, la commande ls permet de lister le contenu du répertoire courant

```
$ cd /usr
$ ls
bin games include lib lib32 local sbin share src
```

4.3.2. Les arguments et l'affichage "long"

En donnant à ls le répertoire /usr en argument, il affiche son contenu.

Un répertoire en argument

```
$ ls /usr  
bin games include lib lib32 local sbin share src
```

On peut faire de même pour un fichier, mais sans autre option, cela aura juste pour effet d'afficher le nom du fichier, ce qui n'est pas très intéressant.

Un fichier en argument

```
$ ls /etc/fstab  
/etc/fstab
```

L'option **-l** pour « long » permet d'afficher une série d'informations sur chacun des fichiers et répertoires à lister. Exécuter un ls sur un fichier a alors plus de sens.

L'option -l affiche plus d'informations

```
$ ls -l /etc/fstab  
-rw-r--r-- 1 root root 1004 fév 1 2013 /etc/fstab
```

Table 5. Description de la sortie de la commande

Nom	Description
-rw-r--r--	Les permissions sur le fichier. Nous verrons cela en détail plus tard.
1	Indique le nombre de hard links pointant vers ce fichier.
root	L'utilisateur propriétaire du fichier.
root	Le groupe propriétaire du fichier.
1004	La taille en octets.
fév 1 2013	La date de la dernière modification du fichier.
/etc/fstab	Le nom du fichier.

Un ls -l sur un répertoire nous affiche donc le contenu avec les détails.

L'option -l avec un répertoire en argument affiche les détails sur le contenu du répertoire

```
ls -l /usr
total 188
drwxr-xr-x  2 root root 81920 avr  7 19:52 bin
drwxr-xr-x  2 root root  4096 mar 31 19:50 games
drwxr-xr-x 61 root root 20480 avr  2 19:56 include
drwxr-xr-x 215 root root 36864 avr  7 19:52 lib
drwxr-xr-x  5 root root  4096 nov 19 23:12 lib32
drwxr-xr-x 11 root root  4096 fév 15  2013 local
drwxr-xr-x  2 root root 12288 avr  2 19:56 sbin
drwxr-xr-x 409 root root 16384 avr  2 18:32 share
drwxr-xr-x 13 root root  4096 avr  2 19:56 src
```

Oui mais, comment faire si on souhaite afficher les détails du répertoire lui-même et non son contenu ? Il faut utiliser l'option -d pour directory.

L'option -l avec un répertoire en argument affiche les détails sur le contenu du répertoire

```
$ ls -ld /usr
drwxr-xr-x 11 root root 4096 fév  1  2013 /usr
```

4.3.3. Human readable please!

Avec l'option -h, Il est également possible d'afficher la taille des fichiers en utilisant des unités de mesure plus intuitives pour nous : Go, Mo ...

L'option -h nous montre la taille du noyau Linux: 5,1 Mo.

```
$ ls -lh /boot/vmlinuz-3.13.7-200.fc20.x86_64
-rw-r--r--. 1 root root 5.1M Mar 24 23:09 /boot/vmlinuz-3.13.7-200.fc20.x86_64
```

4.3.4. Les fichiers cachés

En ajoutant l'option « a » (pour all), ls liste également les fichiers cachés.

L'option -a permet d'afficher les fichiers cachés

```
$ cd /usr
$ ls -a
. .. bin games include lib lib32 local sbin share src
```

Dans notre exemple les seuls fichiers cachés sont les répertoires spéciaux dont nous avons parlés précédemment, qui représentent le répertoire courant et parent.

 Les fichiers (et répertoires) cachés sous Linux, sont simplement des fichiers qui commencent par un point. C'est une convention. Bien entendu, ce n'est pas une fonctionnalité liée à la sécurité : cela n'empêche personne d'afficher et d'accéder aux fichiers cachés. Le système de permission qu'on verra plus tard est là pour ça. Il s'agit plutôt d'une fonctionnalité qui facilite l'utilisation quotidienne : au lieu d'avoir un répertoire encombré de fichiers de toutes sortes, ceux qui sont propres au système et aux diverses configurations sont cachés, et ne s'affichent pas par défaut. Il en va de même dans l'interface graphique : le navigateur de fichiers n'affiche pas, par défaut, les fichiers cachés.

4.4. **pwd – Print Working Directory**

pwd affiche votre répertoire de travail autrement appelé répertoire courant. Il n'y a jamais qu'un seul répertoire courant à un même moment. Si vous êtes perdu et que vous ne savez plus où vous êtes dans le système de fichier, utilisez **pwd** sans modération.

pwd vous indique où vous vous trouvez

```
$ pwd  
/home/titi
```

4.5. Changer d'utilisateur

4.5.1. **su – Switch User**

Switch user veut dire: changer d'utilisateur. Cette commande vous permet de vous logger avec un autre utilisateur. Vous aurez bien entendu besoin de son mot de passe. On utilise ici la commande "who" qui permet de connaître l'utilisateur courant.

L'utilisateur titi devient l'utilisateur toto.

```
$ who  
titi  
$ su - toto  
Password:  
toto@matour:~$
```



On doit utiliser le tiret comme option de "su" afin de charger l'environnement de l'utilisateur de destination, comme si on s'était connecté directement avec cet utilisateur.

Cette commande vous permet également de devenir l'utilisateur root qui est l'utilisateur administrateur du système. Pour ce faire, ne donnez pas d'argument à la commande.

Sans argument, vous vous loggez avec l'utilisateur root

```
$ who  
titi  
$ su -  
Password :  
root@matour:~#
```



Lorsque vous êtes root, vous pouvez switcher sur n'importe quel utilisateur standard sans que son mot de passe vous soit demandé!

4.5.2. sudo

Vous verrez souvent la commande « sudo » dans les tutoriels sur internet. Cette commande est différente de « su ». « su » permet de se logger avec un autre utilisateur. « sudo » permet de vous donner les droits d'administration de votre machine (équivalent aux droits root). Votre utilisateur doit être référencé dans la configuration du système pour avoir le droit d'utiliser la commande « sudo » - tous les utilisateurs n'y ont pas accès. Si c'est l'utilisateur que vous avez créé lors de l'installation de Linux, il y a de forte chances qu'il ait le droit de faire un sudo. Souvent ce droit est acquis lorsque votre utilisateur est membre d'un groupe d'administrateurs. Sur Ubuntu, c'est souvent le groupe « admin ». Sur Centos, c'est souvent le groupe « wheel ». Il y a deux façons d'utiliser « sudo ». * Lorsque vous êtes utilisateur normal, vous ajoutez « sudo » devant toute commande qui nécessite des droits « root » et vous aurez les droits uniquement pour l'exécution de cette commande. * Vous faites un « sudo -i » et vous devenez utilisateur « root » - c'est l'équivalent du « su - root »

Devenir root avec l'option -i de la commande sudo

```
$ who  
titi  
$ sudo -i  
Password :  
root@matour:~#
```

4.5.3. exit – logout

Lorsque vous utilisez **su** ou **sudo -i**, vous créez un nouveau shell. Mais cela ne remplace pas l'ancien. C'est en fait une connexion en cascade. Si je vous montre les processus (qu'on n'a pas encore vu), voilà à quoi ressemble plusieurs **su -** fait d'affilée.

```
\_ -  
  \_ sudo -i  
    \_ -bash  
      \_ su -  
        \_ -su  
          \_ su -  
            \_ -su  
              \_ su -  
                \_ -su
```

Pour sortir de cette connexion et revenir à la précédente, la première méthode est d'entrer la commande **exit**.

La commande exit

```
root@matour:~# exit  
logout  
titi@matour:~$
```

L'autre méthode est d'utiliser le raccourci clavier **ctrl-d**.

Le raccourci clavier ctrl-d

```
root@matour:~# logout  
titi@matour:~$
```

4.6. Arrêter le système

Deux commandes vous permettent d'arrêter le système. La commande **halt** et la commande **shutdown**.

Éteindre avec la commande halt

```
root@matour:~# halt -p
```

Éteindre avec la commande shutdown

```
root@matour:~# shutdown -h now
```