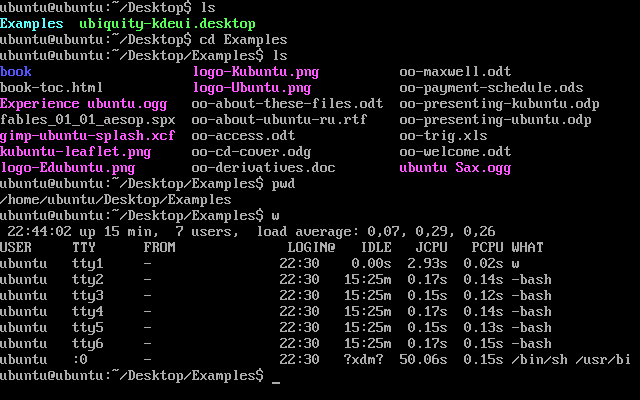
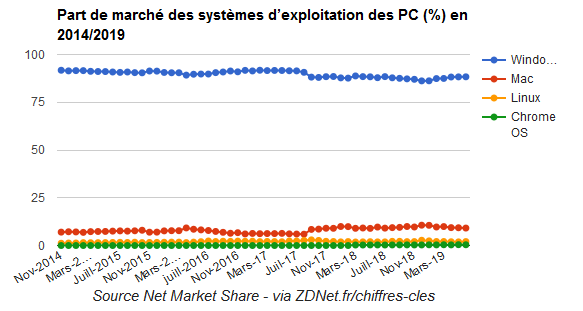
**SYSTEMES D’EXPLOITATION**

Omniprésents dans notre vie actuelle, les systèmes d’exploitation sont indispensables au fonctionnement de tous nos appareils numériques. Sans eux, on ne pourrait effectuer la moindre tâche, que ce soit sur notre ordinateur ou notre téléphone ; retirer de l’argent à un distributeur serait impossible, tout comme utiliser le GPS de nos voitures, voire de regarder un bon film sur l’écran de l’avion pendant un Paris/New-York. Vous l’aurez compris, même si on ne les voit pas forcément dans nos activités quotidiennes, ce sont des programmes incontournables. Plus que des programmes, on leur donne également le doux nom de « super-logiciel » : en effet, c’est l’outil qui va permettre de faire la liaison entre l’humain et la machine ; gérer les ressources de l’ordinateur, permettre de lancer tel ou tel programme en passant par une interface plus ou moins agréable (aussi appelée UI pour « User Interface » - ou Interface Utilisateur). Cette interface a d’ailleurs beaucoup évolué depuis ses débuts : elle est devenue une interface graphique après être restée un simple affichage console aux débuts de l’informatique. Cela a permis d’offrir plus d’ergonomie et d’intuitivité à l’utilisateur (il faut dire que le texte blanc sur fond noir était assez rudimentaire…)

**I. Les différents OS**

*a) L’histoire de MS-DOS / Windows*

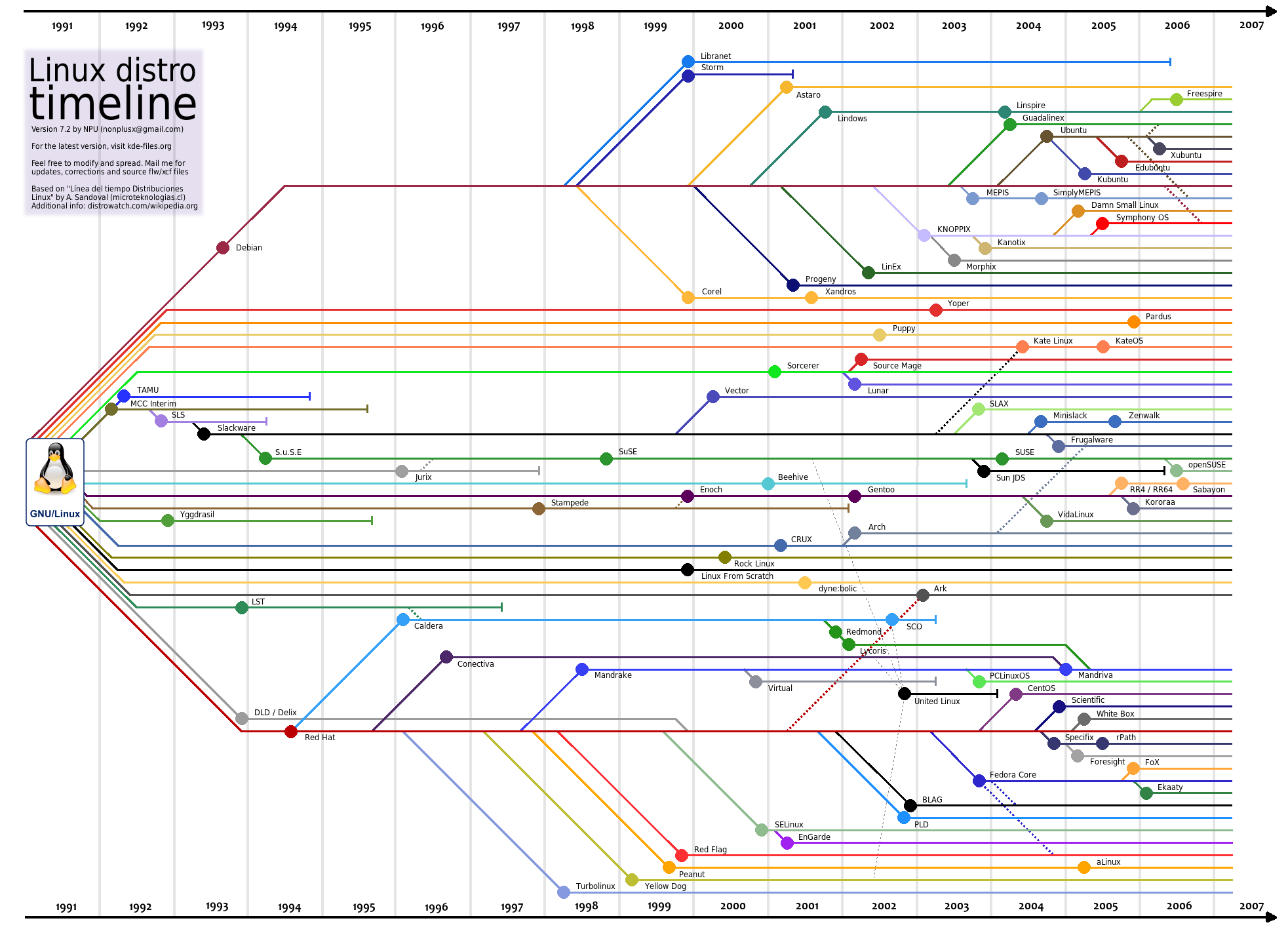
En 1981, Microsoft sort son tout premier OS : MS-DOS. Il s’agit en fait de l’ancêtre de Windows, très populaire aujourd’hui et utilisé par environ 1.5 milliards de personnes dans le monde.



En 1985, Microsoft, qui développait en parallèle un autre système d’exploitation (du petit nom de Windows) en collaboration avec IBM, en sort la première version : Windows 1.0. A la sortie de MS-DOS 7 en 1995, de nombreuses versions de Windows étaient déjà sorties : ainsi, cette nouvelle version permettait de lancer Windows 95 directement depuis MS-DOS pour la première fois. Mais revenons un peu en arrière, lors de la première version de MS-DOS (en 1981, vous vous souvenez ?). En fait, MS-DOS n’était pas le seul OS existant : il y avait également Unix, présent depuis 1969, bien moins connu du grand public. En revanche, celui-ci était bien plus puissant et complet que son rival ; ainsi, seuls les informaticiens professionnels l’utilisaient.

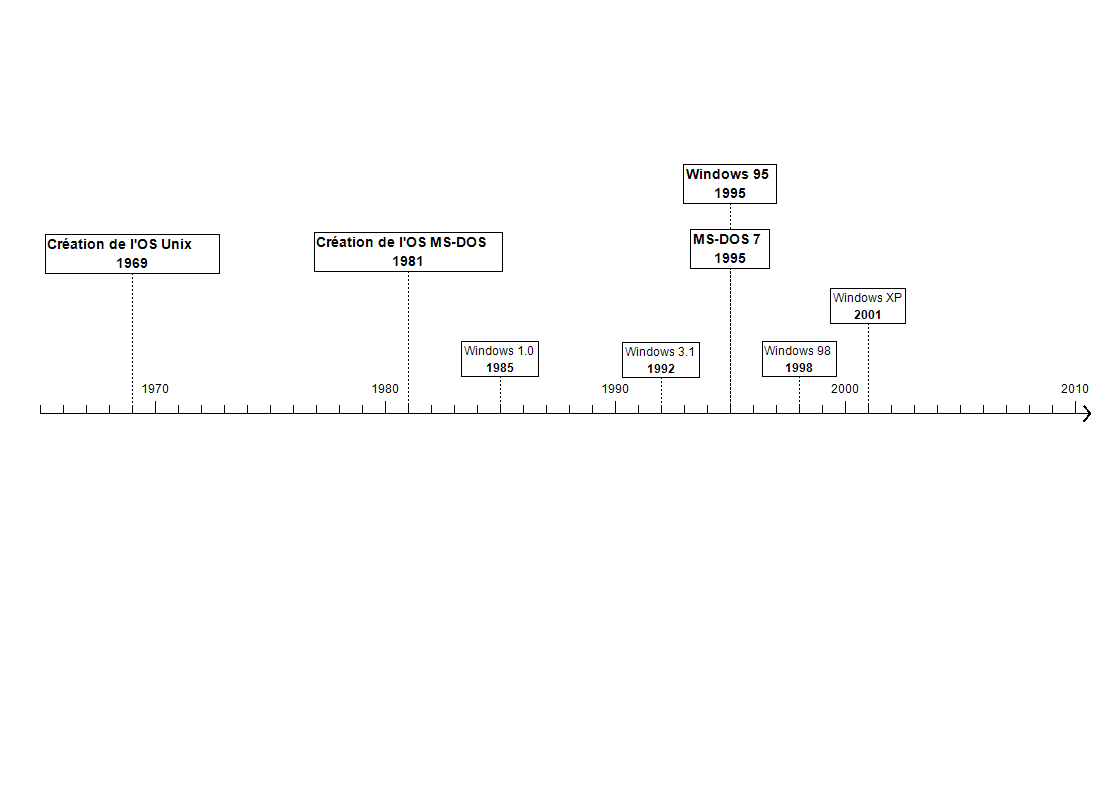
*b) Unix*

A cette époque, Unix était le seul véritable système d’exploitation adapté pour les professionnels : cependant, il était payant et son prix ne cessait d’augmenter. Ainsi, en 1984, Richard STALLMAN créa le projet GNU. Le but était de créer un OS similaire à UNIX et avec les mêmes commandes : une « copie », en quelque sorte, se présentant comme alternative gratuite. La principale différence, en dehors de la gratuité, était que GNU se voulait un logiciel libre. C’est-à-dire un programme (tout de même généralement gratuit) dont tout le monde peut avoir le code source afin de le modifier, de l’améliorer, de le redistribuer. Cette décision était fondamentale, car à l’époque, (quasiment) tous les programmes et notamment MS-DOS et UNIX était propriétaires (on ne pouvait pas avoir accès au code source, on se contentait uniquement de les utiliser). Ainsi, depuis des années, nombre de développeurs travaillent sur plusieurs « variantes » de Linux, des distributions (référez-vous à la photo ci-dessous pour vous faire une petite idée 😉).



En 1991, Linus TORVALDS, alors étudiant à l’université d’Helsinki, entreprend la création de son propre OS sur son temps libre (qui s’appellera Linux, pour la contraction de Linus et Unix). Ainsi, alors que Richard STALLMAN s’occupait de la gestion des commandes du système, des programmes (notamment), Linus créait quant à lui le noyau du système, son « cœur ». Les deux projets ont donc fusionné pour donner naissance à GNU/Linux : ainsi, lorsque l’on parle de Linux, on parle en fait de la fusion des deux projets dont le nom complet est GNU/Linux, mais par abus de langage, seul le nom de Linux est resté.

MAC OS est d’ailleurs directement issu de Unix et est resté un programme propriétaire, dont seul Apple possède le code source.



**II. Linux aujourd’hui**

Aujourd’hui, Linux est omniprésent dans le monde numérique. Que vous sachiez vous en servir ou non, vous vous l’utilisez indirectement presque tous les jours. En effet, de nombreux serveurs tournent sous Linux pour sa fiabilité accrue : ainsi, de nombreux sites comme YouTube, Google… tournent sur des serveurs sous Linux.

Mais son utilisation, bien que majoritaire dans ce cadre, ne se résume pas qu’aux serveurs : cet OS est également utilisé dans le monde des objets connectés, et aussi par des particuliers, même si son utilisation reste très limitée par rapport à son concurrent de taille : Windows **(cf. graphique).**

Nous allons donc dans ce cours apprendre à appréhender Linux (ça ne mord pas) et l’utiliser, en apprenant quelques commandes de base mais très utiles. Nous travaillerons uniquement sur la distribution « Ubuntu », qui est une des plus populaires et la plus adaptée aux débutants. Concrètement, le fonctionnement reste relativement similaire pour les autres versions, mais selon la distribution que vous utilisez, certaines commandes peuvent différer, dans la méthode d’installation d’applications notamment.

NB : Ce chapitre suppose que vous avez déjà installé Ubuntu. Si vous avez des difficultés à le faire, n’hésitez pas à toquer chez votre meilleur ami : Google.

NB 2 : Tout ce que nous verrons dans ce chapitre concerne uniquement l’utilisation de la console (oui ce gros truc noir tout moche) : je pense que vous devriez vous en sortir touts seuls avec l’interface graphique 😊.

*a) L’arborescence sous Linux*

Sous Linux, les fichiers sont organisés différemment que sous Windows. Vous n’aurez pas de « Disque local C : » où sont stockés les programmes et le système d’exploitation, ni de « Disque local E : » lorsque vous banchez votre clé USB. En fait, le C : pourrait s’apparenter au dossier racine, celui d’où tous vos fichiers divergent : le dossier « / ». Oui, juste un slash, pour indiquer que l’on se trouve au tout début de l’arborescence.

Pour naviguer dans cette arborescence, la commande est la même que son Windows (pour ceux qui sont familiers avec le CMD) : il s’agit de **cd**. Ainsi, si je souhaite aller dans un dossier qui se trouve dans mon dossier actuel (imaginons que je sois dans le dossier *images* et que je souhaite aller dans le dossier *paysages*, je taperai simplement : **cd paysages**. C’est tout. Imaginons maintenant que me trouve dans ce dossier *paysages* et que je souhaite revenir en arrière : 2 solutions. Soit je peux taper **cd ..** afin de « reculer » d’un dossier (ou même **cd ../..** pour reculer de 2 dossiers), soit j’indique le chemin absolu du dossier dans lequel je veux aller. Qu’est-ce que le chemin absolu ? C’est simplement l’emplacement d’un dossier ou d’un fichier en fonction du dossier racine (ça fait un peu peur comme ça, mais c’est très simple vous allez voir). Prenons notre dossier *Images*et admettons que celui-ci se trouve dans le dossier *personnel* qui lui-même se trouve dans le dossier *utilisateurs* qui lui-même se trouve à la racine (ça en fait des dossiers) : alors son chemin absolu est : **/utilisateurs/personnel/images**. Si vous avez suivi jusqu’ici, vous devriez être tout à fait capables de trouver l’arborescence du dossier *Paysages* puisque celui-ci se trouve dans le dossier *images* ! Cela donnerait quelque chose comme : : **/utilisateurs/personnel/images/paysages**. Ainsi, où que je me trouve, si je souhaite aller dans ce dossier *paysages*, la commande **cd /utilisateurs/personnel/images/paysages** fonctionnera à merveille peu importe le dossier dans lequel je suis à cet instant. En revanche, la commande **cd paysages** ne fonctionnera que si je me trouve dans le dossier *images*, tout comme la commande **cd images/paysages**  ne fonctionnera que si je me trouve dans le dossier *personnel*, etc.

Note : les commandes **cd /utilisateurs/personnel/images/paysages** et **cd /utilisateurs/personnel/images/paysages/** fonctionnent à l’identique. Le slash en fin de commande est inutile si on n’a plus besoin d’indiquer de dossier après.

*b) Manipuler les fichiers et les dossiers*

Bon, maintenant que vous savez naviguer sur votre navire, il est temps d’aller à l’abordage des commandes principales servant à effectuer des actions sur des fichiers et des dossiers.

Voici une petite liste de commandes de base à connaître :

* **ls**(c’est bien un L minuscule et non un i majuscule) **:** elle permet d’afficher le contenu d’un répertoire (dossier). C’est l’abréviation de « list », lister.

*Note : En fait, cette commande n’affiche pas réellement tout le contenu d’un dossier. Pour voir les fichiers cachés entre autres, il faudra passer le paramètre* ***-a*** *à la commande, ce qui donnera* ***ls -a*** .

* **less**: cette commande permet d’afficher le contenu d’un fichier (sans possibilité de modification). Son utilisation est simple : **less nomdufichier** .

Attention, vous n’avez qu’une partie du fichier qui apparaît à l’écran : en effet, elle affiche le fichier progressivement (à l’inverse de **cat**). Pour afficher la ligne suivante, appuyez sur Entrée et y pour retourner d’une ligne en arrière. Pour afficher directement une « page » d’après (c’est-à-dire le nombre de lignes qui rentrent dans l’écran de la console), faites Espace. b permet de retourner en arrière d’un écran (attention, b et pas B ! ).

* **nano :** cette commande permet de d’afficher et de modifier un fichier (et le créer s’il est inexistant). Pour l’utiliser, faites **nano nomdufichier** . Ensuite, laissez-vous guider par le programme 😉.
* **mkdir**: cette fois, on crée un dossier. Son utilisation reste basique : **mkdir nomdudossier** .
* **cp**: aucun rapport scolaire, cette commande est l’abréviation de l’anglais « copy » et permet donc… de copier. Pour copier un fichier, un **cp fichier fichiercopie** aura pour effet de copier le fichier nommé « fichier » dans votre répertoire courant (celui dans lequel vous êtes) en lui affectant un autre nom : « fichiercopie ». Si vous vous trouvez dans notre dossier *images* et que vous voulez copier une photo par exemple, dans le répertoire *paysages*, vous devrez taper : **cp photo.jpg paysages/** . Notez que le slash à la fin du dossier *paysages* est inutile, il permet d’indiquer qu’il s’agit bien d’un dossier mais peut être omis. NB : si j’avais voulu copier la photo dans le dossier *paysages* sous le nom de *photocopiee.jpg*, j’aurai dû écrire **cp photo.jpg paysages/photocopiee.jpg.** NB 2 : on évite en général les accents et les espaces sur les noms même si Linux les gère très bien (pas d’inquiétude !).
* **mv**: de l’anglais « move » ; cette fois-ci, il s’agit de déplacer un fichier ou le renommer. Pour déplacer notre image nommée *photocopiee.jpg* du dossier *paysages* vers le dossier *images* en lui donnant le nouveau nom *photodeplacee.jpg*, je vais donc, si je souhaite le faire depuis n’importe quel emplacement, saisir le chemin absolu (vous vous souvenez de ce truc?). Cela donnerait donc : **mv /utilisateurs/personnel/images/paysages/photocopiee.jpg /utilisateurs/personnel/images/photodeplacee.jpg**.Nous avons donc déplacé notre photo en la renommant, tout ça en une seule commande !
* **rm**: (« remove ») ; supprimer un ou plusieurs fichiers et/ou un ou plusieurs dossiers. Son utilisation est aussi simple que dangereuse. En effet, il n’existe pas de « corbeille » dans la console Linux : une fois cette commande exécutée, si vous supprimez par erreur quelque chose d’important, vous n’aurez que vos deux yeux pour pleurer. On l’utilise comme toutes les commandes vues jusqu’ici. Admettons que je me trouve dans le dossier *images* : exécuter ceci **rm photodeplacee.jpg** supprime instantanément sans confirmation la photo. Je peux également me servir du chemin absolu pour supprimer, pourquoi pas, la photo *photo.jpg* dans mon dossier *images*: **rm /utilisateurs/personnel/images/photo.jpg**. Pour supprimer un répertoire et tout ce qu’il contient, il faudra y ajouter un paramètre, -r : **rm -r /utilisateurs/personnel/images**. Attention, je me permets de rappeler que cette commande est très dangereuse, utilisez-la avec beaucoup de précautions, car vous pouvez même détruire votre système avec ça. Relisez-vous toujours avant d’effectuer cette commande (je parle en connaissance de cause…).

*c) Les droits*

Sous Linux, quand vous vous connectez à votre session, vous n’avez que des droits limités. C’est-à-dire que vous ne disposez pas des droits administrateurs sur votre machine (vous avez peut-être déjà entendu ce terme sous Windows). Il s’agit en fait d’une sécurité mise en place par les développeurs : en effet, il vous sera (normalement) impossible de détruire votre système si vous n’êtes pas administrateur. De plus, si vous attrapez un virus, celui-ci ne pourra occasioner de gros dégâts sur votre machine puisque, comme vous, il n’aura pas accès aux dossiers et commandes les plus sensibles, accessibles unqiuement aux administrateurs. Ici, pour évoquer le terme d’administrateur, on parlera d’utilisateur root (racine). C’est l’utilisateur qui a la permission d’accéder et/ou de modifier les données situées à la racine de votre système (vous vous souvenez, l’emplacement « / »).

Le problème majeur de cette sécurité est que vous ne pouvez finalement effectuer aucune modification sur votre système. Il faudra donc s’attribuer les droits super-utilisateurs (root), au moins temporairement (pour l’installation d’applications entre autres).

Pour ce faire, ouvrez la console (que vous commencez à bien connaître, n’est-ce pas ?), et tapez **sudo ls**, puis rentrez votre mot de passe (ne vous inquiétez pas si rien ne s’affiche, c’est tout-à-fait normal). Si vous êtes dans votre dossier personnel, cela devrait vous renvoyer la liste des éléments présents dans celui-ci. Pas de différence avec un simple **ls**, me direz-vous. Eh bin vous avez tout à fait raison, au détail près que vous venez d’effecteur cette commande en tant que super-utilisateur. Evidemment, cette commande ne nécessite pas d’être administrateur pour pouvoir être exécutée, donc ce n’était pas très utile. L’idée était de vous familiariser avec la commande **sudo** sans risque. Néanmoins, après votre superbe **sudo ls**, vous êtes repassés en droits « limités » (les droits de votre session). Vous devrez ainsi retaper **sudo** devant chaque commande pour pouvoir repasser administrateur. Il existe cependant une petite astuce pour pouvoir rester super-utilisateur de manière permanente (jusqu’à ce que vous fermiez la console ou appuyiez sur la touche Echap) : la petite commande **sudo su** qui vous demandera une fois votre mot de passe puis vous attribuera tous les droits sans que vous ayez besoin de taper **sudo** devant chaque commande.

Je ne vais pas lister ici toutes les possibilités offertes par cette commande, mais vous devez au moins savoir qu’elle est nécessaire à l’installation d’applications (avec la commande **sudo apt-get install nomdelapplication**). Pour plus de détails, je vous renvoie au site d’OpenClassrooms qui est très complet et offre beaucoup de précisions à ce sujet.

Une dernière chose, et pas des moindres : il va ici falloir s’occuper des permissions d’accès aux fichiers. Comme je vous l’ai dit au-dessus, l’utilisateur root a tous les droits : rien ne peut lui être empêché de faire sur la machine. Pour les simples utilisateurs, c’est un peu différent. En effet, admettons que je crée un fichier dans mon dossier *personnel* : ainsi, j’en suis le propriétaire. Je peux ainsi, si plusieurs utilisateurs sont inscrits sur la machine, dire : tel utilisateur a le droit de l’ouvrir, lui a le droit de l’ouvrir et de le modifier, lui a simplement le droit de l’exécuter, lui peut tout faire avec. *Note : le propriétaire reste toujours le même pour le fichier, sauf s’il en décide autrement grâce à la commande* ***chown*** *que nous ne verrons pas ici, mais vous pouvez toujours aller voir le cours d’OpenClassrooms pour les plus curieux).* Comme je disais, le propriétaire d’un fichier peut attribuer différents droits pour son fichier à différents utilisateurs (un utilisateur qui n’est pas propriétaire du fichier ne peut pas dire à un autre : « Tiens, toi tu auras le droit d’ouvrir et de modifier ce fichier » : cela n’a pas de sens !). Les droits sont donc gérés selon 3 lettres : r, w, x. r signifie « read », ce qui veut dire que le fichier est accessible en écriture. w signifie « write », ce qui veut dire que le fichier est accessible en écriture. Et enfin x signifie « execute », ce qui veut dire que le fichier peut être exécuté. Toujours en admettant que vous êtes propriétaire du fichier, si vous effectuez un **chmod +r fichier.txt**, cela aura pour effet d’ajouter le droit d’écriture à tous les utilisateurs de la machine. Le principe est le même pour la lecture et l’exécution. Vous pouvez voir qui a accès au fichier en faisant un **ls -l** dans le dossier contenant le fichier : vous devriez voir une série de lettres comme -rwxrw-r-- par exemple. Je ne vais pas entrer dans les détails, mais les lettres sont organisées par 3 : en partant de la fin, le bloc de 3 caractères est « r--» ce qui signifie que tout le monde a le droit de lire le fichier. Le deuxième groupe est « rw-» ce qui signifie que le groupe a le droit de lire et d’écrire dans le fichier (pour plus de précisions, je vous renvoie vers le même cours). Enfin le troisième groupe (en partant de la fin) est « rwx », ce qui signifie que le propriétaire a le droit de l’ouvrir, le modifier et l’exécuter (un peu logique, non ?). Le premier tiret signifie qu’il s’agit d’un fichier : s’il s’agissait d’un dossier, il aurait été remplacé par un « d ».

Si tout ça ne vous paraît pas très clair, j’insiste, allez faire un tour sur le chapitre « les utilisateurs et les droits » sur le cours d’OpenClassrooms afin que cela ne soit pas fouillis dans votre tête.

Un dernier conseil et je vous laisse partir : si jamais vous avez besoin d’utiliser une commande, mais que vous ne savez pas comment (quels paramètres mettre, etc.), vous pouvez taper **man nomdelacommande** pour lire la documentation de cette commande (essayez avec un **man ls** pour voir 😉). Voilà, il est temps pour moi de vous laisser faire joujou avec votre machine, mais n’oubliez pas que vous pouvez aller faire un tour sur le cours dont je n’arrête pas de vous parler si vous êtes curieux et souhaitez en savoir plus sur votre machine, car je n’ai fait qu’aborder les choses les plus simples pour cibler les plus débutants d’entre vous. GLHF !

Sources et crédits :

OPENCLASSROOMS – « Reprenez le contrôle de votre ordinateur à l’aide de Linux ! » par Mathieu Nebra

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