Le Shell :

Le shell est l'interface de commandes (instructions simples) qui permet une communication Homme/machine sans utiliser beaucoup de ressources de l'ordinateur.

Les commandes se composent de cette manière :

[commande] [-paramètres] [argument1] [argument2]

Ouvrir Terminal de commande : Ctrl + Alt + T

<u>Commandes shell principales</u>

- **pwd :** Renvoie le chemin de la localisation actuelle.
- **ls :** Affiche la localisation actuelle et son contenu (fichiers et sous-dossiers)
- **ll** : Vaut un ls -al, afficher la localisation actuelle et son contenu en incluant les fichiers cachées et les informations (permissions, date de modification,..;)
- ../ pour le dossier d'origine et ./ pour le dossier actuel (attention installé de base que sur certaines versions de linux)
- **cd :** Permet de se déplacer, à la racine sans argument et dans un endroit précis avec.
- **cd ..** : Permet de revenir un cran en arrière
- **locate :** Localise aussi les fichiers passés en arguments. Besoin de préciser à l'origine de la recherche.
- **find :** Localise tous les fichiers passés en arguments.
- **mkdir :** Créer un dossier. Prend un argument nom. (-p pour créer un dossier et son sous-dossier)
- **touch :** créer un fichier. Prend un argument nom avec ou sans extension.
- **cat :** Affiche le contenu d'un fichier directement dans la console
- **less :** Entre dans un fichier et affiche dans le terminal son contenu. Plus pratique pour des longs fichiers. q pour quitter
- **nano :** Édite un fichier, si le fichier n'existe pas, le créé.
- **which :** Localise une commande.
- **mv :** déplacer ou renommer un fichier ou un dossier, renommer sur Linux est juste déplacer quelque chose vers une autre destination. Prend deux arguments, la source puis la destination.
- **cp :** Copier/coller un fichier ou un dossier. Fonctionne comme **mv** avec deux arguments.
- **chmod :** Modifie les droits d'accès d'un fichier ou dossier. **chmod 755** est cool
- **wget :** Installe une application directement sur internet, nécessite une URL en argument.
- **apt-get ou apt :** installer une application dans bin/bash grace a la bibliotheque du noyaux linux.
- **rm :** Supprimer un dossier

- **rm -rf :** Supprime un dossier et tout son contenu. Attention: Force et peut supprimer le contenu total d'un ordinateur.
- **-i :** Demande une confirmation de suppression. Répondre y ou n, yes or no.
- *exemple : rm -rfi mondossier me demande confirmation puis supprime mondossier et tout son contenu*
- **rmdir :** Supprimer un dossier si et seulement si il est vide
- **ssh :** se connecte à un serveur distant. Attend un argument username@adresse_serveur
- **scp :** Envoie un fichier à un serveur distant. Nécessite deux arguments : un pour la source du fichier, un pour la cible sous forme username@adresse_serveur:dossier
- **exit :** quitte le serveur
- **ping :** afficher le temps de latence vers une IP
- **passwd :** changer son mot de passe. Pas d'argument nécessaire.
- **hostname - I :** Trouver son IP

- Résumé facile des commandes essentielles Shell : http://cheatsheetworld.com/programming/unix-linux-cheat-sheet/

Éditeurs de texte:

- **vim** : plus complet
- **nano** : plus simple d'utilisation

<u>À noter :</u>

- **man :** donne des indications sur l'utilisation d'une fonction. Son but, ses paramètres et ses arguments.
- Auto-complétion en utilisant la touche *tab*
- Utiliser flèche haut et flèche bas pour naviguer dans l'historique des commandes entrées
- Le Shell est sensible à la casse, une majuscule et une minuscule sont différentes
- **sudo :** Placé avant une commande, permet de l'exécuter en tant que super utilisateur. (Attention).
- **clear :** Nettoie l'historique de la console.
- **ctrl + shift + t** = ouvre un nouvel onglet terminal.
- installer **wget** = sudo apt-get install wget.
- **sudo apt update :** Affiche les mises à jour possibles pour la "bibliothèque" apt-get .
- **sudo apt upgrade :** mets à jour la bibliothèque du noyaux linux aux versions que le apt update à trouvées (root = admin)
- **Sortir de quelque chose : Ctrl + C**

ssh apprenant@simplonlyon.fr
ssh : secure shell

scp -r folderToMove username@simplonlyon.fr:var/ww/etc/

Mercredi 19 Décembre 2018

La gestion des permissions

La gestion des permissions sur linux/unix permet de :

- limiter ou donner des droits aux utilisateurs sur les fichiers ou dossiers
- sur chaque fichier de gérer indépendamment utilisateur, groupe et autres (U,G & O)
- sur chaque fichier de gérer indépendamment les droits de lecture, d'édition et d'exécution (r, w & x)

Pour lire les permissions sur un **ls -la** ou **ll**, dans l'ordre :

- lettres pour permissions (modifiables avec **chmod**)
- d, tiret ou l : dossier, fichier ou lien symbolique (raccourci)
- trois paquets de lettres ou tirets (user, group, other)
- r pour read, capacité de lire le fichier
- w pour write, capacité de modifier le fichier
- x pour exécution (pour un dossier, entrer dedans)
- deux noms :
- user à qui appartient le fichier
- groupe auquel appartient le fichier. Par défaut prend le nom d'user.
-

chmod : modifie les permissions d'un fichier ou d'un dossier

- **chmod ugo** : modifier les permissions d'user (u), de group (g) et other (o) ou chaque lettre est remplacée par un chiffre de 0 à 7. read = 4, write = 2 et execute = 1
- *exemple : chmod 715 mon_fichier me donne tous les droits sur un fichier (7), uniquement les droits d'exécution au groupe et les droits de lecture et d'exécution pour les autres (5)*
- **chmod u+rwx :** Rajoute des droits spécifiques à u, g, o ou a (all) directement avec les lettres r w x (ou les enlève avec -)
- *exemple : chmod ug-rx mon_dossier enlève les droits de lecture (r) et d'exécution (x) pour l'utilisateur et le groupe*
- **chmod -R** : changer les droits pour tous les fichiers d'un dossier. (-R = recursif)

chown : Change le propriétaire d'un fichier ou d'un dossier, nécessite un argument nouvel utilisateur à qui on veut transférer les droits puis le fichier. -R comme **chmod**

- chown USER:GROUP nom_fichier pour changer en même temps l'utilisateur et le groupe propriétaires

chgrp : Change le groupe propriétaire du fichier(ex : chgrp : $USER nomdufichier)

su : Switch User, permet de changer d'utilisateur actif grâce à un argument username. Si vide, switch à root.
groups : Affiche les groupes dont fait partie l'utilisateur actif
ln -s : créé un lien symbolique (raccourci). Nécessite un argument de la cible et peut-prendre un argument nom. Si pas de nom prend le nom de la cible.

echo : écrit quelque chose dans la console

- **echo** "écriture" > fichier : écrit et remplace l'intérieur du fichier.
- **echo** "écriture" >> fichier : écrit et ne remplace pas l'intérieur du fichier.

Git :

Git est un système de gestion et de partage des fichiers, particulièrement adapté à des travaux en programmation. Git permet de :

- Simplifier le travail en équipe
- Travailler en arborescence en développant simultanément plusieurs "versions" de code
- Garder un historique des fichiers / modifications

Pour installer $sudo apt-get install git

Commandes git

git init : Initialise un dépôt
S'identifier avec git :

- git config --global user.email "Vous@exemple.com"
- git config --global user.name "Votre Nom"

git help : Affiche la liste des commandes et de leurs fonctionnalités
git status : Affiche le statut git du dossier actuel
git log : Affiche l'historique du dépôt (commits..)
git show : Suivi de l'ID du commit (voir **git log**) affiche le détail d'un commit

git add : Ajoute un ou plusieurs fichiers à suivre
git reset : Enlève un git précédemment add (ne pas oublier le nom du ou des fichiers en question)
git revert : copie le commit précédent choisit pour ne pas supprimé le commit actuel.

git commit : Envoie chacun des éléments add dans le dépôt. Important de commenter pour la traçabilité.

- *exemple : git commit -m "commentaire"*
- Penser à faire plusieurs commit par jour et assez courts pour expliquer ce que l'on a fait

git rm : Efface le fichier et son lien
git checkout : Suivi du nom du fichier, restaure la version du fichier qui est sur le dépôt

git clone : Fais une copie du dépôt actuel à l'adresse passée en argument. Possibilité de rajouter un nom au dossier qui va se créer.
git push : Dans un clone, envoie les modifications vers la branche d'origine du clone
git pull : Dans un clone, récupère les modifications de la branche d'origine vers clone
git tag: ajoute une étiquette (nom) spécifique à un commit

git stash list : Affiche la liste des stashs
git stash save "message" : Enregistre l'état du dépot à l'instant T et l'enregistre en local
git stash apply : Applique un stash dans la branche pour obtenir ce qui y était stocké

Branche master : Branche principale du dépôt (code source).
origin/master : Branche master de laquelle est issue un clone

clone sur ordinateur perso : ssh
clone sur rasberry ou simplon : https

gitlab : outil qui stocke tous les codes en commun sur un serveur, accessible depuis un navigateur

20 Décembre 2018

Branches
git fonctionne en arborescence : une branche représente une ligne de développement indépendante qui peut être fusionnée par la suite avec la Master. On utilise les branches notamment pour :

- Garder et maintenir plusieurs versions d'un travail
- Garder un master propre
- Travaille en équipe
- Faire une autre version d'un truc
- Régler un souci sans mettre en péril le code source

git branch : permet de gérer les branches (création, suppression, ...)

- **git branch nom :** créer une branche qui s'appelle nom
- **git branch -d nom :** supprime la branche qui s'appelle nom

- **git branch -a** : affiche toutes les branches en liaison avec la Master. (affiche les branches distantes)
- **git branch -f :** Force une branche à se déplacer là ou est HEAD

git checkout : changer de branche, déplace la head sur la branche passée en argument

- Suivi d'un ID de commit, déplace la HEAD sur le commit

git merge : fusionne deux branches entre elles, fusionne la branche passée en argument dans la branche où on se situe
git rebase : Rebase une branche sur la branche cible. Supprime la branche.

- **-i :** ouvre une interface graphique pour un rebase interactif

git reset : remonte head dans la liste des commits
git revert : retourne à un (ou plusieurs si passé en argument) commit en arrière
git fetch : fais un pull des dernières modifications distance sans les fusionner (permet de choisir ce que l'on veut prendre par rapport à un **git pull**)
git cherry-pick : Copie des commits par leurs ID et les ajoutes à la suite de la branche active

git fetch + git merge = git pull

Jeu pour apprendre les branches de git : https://learngitbranching.js.org

21 Décembre 2018

Remote

git remote sert à créer des dépôts distants, espaces de stockage indépendants vers lesquels on peut push pull et fetch sans qu'ils soient un espace de travail à proprement parler.
remote = dépot distant

- **git remote add :** ajoute un remote repository, nécessite un argument nom
- **git remote add origin *url-depot* :** *Lie un git initialisé sur l'ordinateur à un dépôt déjà existant*
- **git remote remove ou git remote rm:** retire un remote repository
- **git remote -v :** Liste les depository distants
- **git remote set-url** : Permet de modifier l'emplacement de stockage d'un dépôt distant, nécessite deux arguments du nom de la cible et de sa nouvelle URL

- **-v :** Indique quel est l'origin master du clone

ssh-keygen génère une clef public liée à une clef privée mathématiquement