LISTE DES COMMANDES LINUX

TD1 :

-passwd : commande pour changer de mot de passe
-ls : commande pour voir le contenu d'un dossier
-cd : commande pour changer de dossier courant
-pwd : commande pour voir le chemin du dossier courant
-exit : commande pour se déconnecter de linux1
-nano : commande pour éditer le contenu d'un fichier
Commandes supplémentaires :

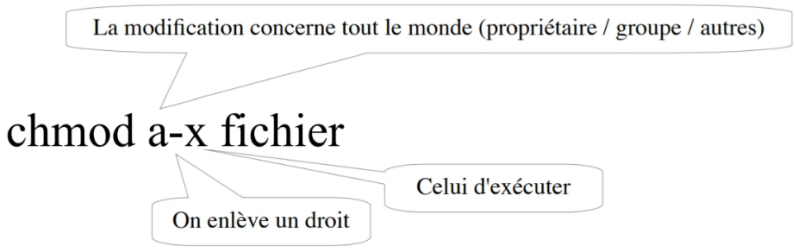
-**cat** " *nomDuFichier* "affiche à l'écran le contenu du fichier dont le nom est donné (ce n'est pas un éditeur, on voit le contenu et c'est tout) ;
(-**more** : commande qui affiche le contenu d'un fichier, idéal s'il est long) ;
-**mkdir** "*nomDuDossier* " crée un dossier (vide) nommé "*nomDuDossier*" ;
-**mv** " *nomDuFichier nouveauNomDeFichie*r " renomme le fichier donné "*nomDuFichier*" sous le nom "nouveauNomDeFichier" ;
-**mv** "*nomDuFichier nomDuDossier*" déplace le fichier donné dans le dossier indiqué ;
-**cp** " *nomDuFichier nouveauNomDeFichier* " crée une copie du fichier sous le nom "*nouveauNomDeFichier*" ;
-**cp** " *nomDuFichier nomDuDossier* " copie le fichier donné dans le dossier indiqué ;
-**rm** "*nomDuFichier*" détruit le fichier dont on donne le nom ;
-**rmdir** "*nomDuDossier*" détruit le dossier dont on donne le nom (Attention, le dossier doit être vide !)
-**javac** : commande pour compiler un fichier java
-**java** : commande pour exécuter un fichier java (toujours javac avant java !)

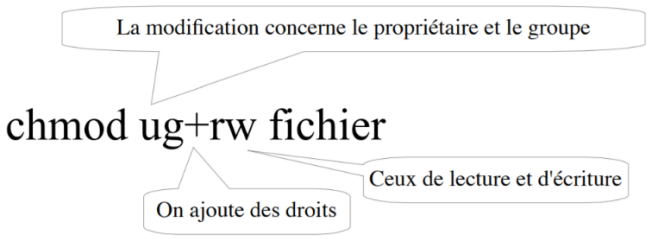
TD2 :

-ls – l : affiche tous les fichiers/ dossiers, ainsi que leur propriétaire, les permissions et le groupe dans lesquels ils se trouvent
-**commande /~** : raccourci pour n'importe quel commande (au lieu d'écrire un chemin)
-groups : commande qui affiche les groupes auxquels l'utilisateur appartient
-**groups** ''*utilisateur*'' : affichera les groupes d'un utilisateur en particulier
-**chown** ''*NomUtilisateur monFichier*'' : commande qui change le propriétaire du fichier monFichier qui devient NomUtilisateur
-**chgrp** ''*etudiants1 monFichier*'' : commande qui place le fichier monFichier dans le groupe etudiants1
-**chmod** (4,2,1) ''*NomDuFichier/NomDuDossier*'' : commande qui change les permissions d'un dossier/fichier
Exemple :
-*chmod 640 monFichier.java* donne au propriétaire le droit en lecture/écriture, aux utilisateurs dans le même groupe que le fichier le droit en lecture uniquement et aucun droit aux autres.
-*chmod 777 monFichier.java* donne à tout le monde tous les droits

Lettres	Nombre	Signification
r- -	4 (4+0+0)	Droit en lecture uniquement
rw-	6 (4+2+0)	Droit en lecture et écriture
--x	1 (0+0+1)	Droit d'exécution uniquement

Exemples :





-**umask** : commande qui change les permissions par défaut d'un fichier
-**ls - -help** : commande qui affiche toutes les commandes possibles avec ls
-**man** : commande qui affiche toutes les commandes
-**man ls** : commande qui affiche toutes les commandes de ls (q pour quitter)
-**clear** : commande qui permet de nettoyer l'écran

TD3 :

-**TAB (touche, 2X)** : touche qui permet d'afficher la liste des commandes disponibles
- **x TAB (2X)** : touche qui affiche l'ensemble des commandes commençant par la touche x (q pour **quitter)**
- javac ''*NomFichierJava*'' : commande qui exécute un fichier écrit en java (à condition qu'il soit bien écrit, sinon il affichera les erreurs)
- **nano ~/.nanorc** puis **include "/usr/share/nano/java.nanorc"** : commande pour colorier un fichier
-**nano -c** ''*NomFichierJava*'' et une fois dans l'éditeur **CTRL -c** : commande pour indiquer le numéro de la ligne où se trouve le curseur.
- **man nanorc** : commande pour voir toutes les configurations possibles pour nano
- **nano .nanorc**. : commande pour créer un fichier caché

TD4 :

-grep *pattern ''NomduFichier''* : commande qui permet d'extraire de fichiers toutes les lignes qui contiennent un certain texte (appelé pattern).
Exemple : Pour trouver dans quel fichier vous avez utilisé une variable nommée "surface", vous pouvez écrire : *grep surface *.java*
-wc *''NomduFichier''* : commande qui indique le nombre de lignes, de mots et de caractères contenus dans les fichiers donnés.
-wc -w *''NomduFichier''* : commande qui indique juste le nombre de mots contenus dans les fichiers donnés.
-wc -c *''NomduFichier''* : commande qui indique juste le nombre de bytes contenus dans les fichiers donnés.
-wc -m*''NomduFichier''* : commande qui indique juste le nombre de caractères contenus dans les fichiers donnés.
-wc -l *''NomduFichier''* : commande qui indique juste le nombre de lignes contenus dans les fichiers donnés.
-wc -L *''NomduFichier''* : commande qui indique juste la longueur de la plus longue ligne.
-**find ~ -name** ''*NomduFichier*'': commande qui permet de rechercher dans une arborescence de fichiers ceux qui correspondent à un critère donné (taille, droits, nom, dates...).
Exemple: find ~ -name Ex.java ou find /eCours -name Color.class
→ Changer de couleur dans un fichier java :
Dans le fichier java, tapé sout(**Color.toRed**(*''Texte à mettre en couleur''*)
+**Color.toGreen***(''Deuxième texte à mettre en couleur''*)) ;
→ Le prompt Le prompt (ou invite en français) est le texte qui apparait à gauche de ce que vous tapez dans votre shell. Il est déterminé par la variable d'environnement PS1 (exemple etd@laboajava :)
Pour le changer : **echo $PS1** puis **PS1=nouvelle valeur** (exemple Bonjour)
Pour que le changement soit permanent : cd **.bashrc**. **echo $PS1** puis **PS1=nouvelle valeur**

TD5 :

Liste des commandes à connaitre absolument :

-git init : initialise un "dépôt Git''
-git status : affiche ce que Git ''comprend'' de notre répertoire ou des commandes tapées précédemment
-git add <file>... : définit les fichiers et modifications à intégrer au commit
-git commit -m « message » : crée le commit
- find . -type ''PremièreLettreFichier'' : affiche tous les dossiers commençant par la première lettre tapée
-**touch ''NomFichier'' :** crée un fichier vide
-**echo ''NomFichier'' >. gitignore :** demande au Git d'ignorer le fichier au nom correspondant
-**rm ''NomFichier''.gitignore :** nettoie les commandes précédemment tapées
-git log [--all] [--graph] [--oneline]: commande pour voir l'évolution du dépôt de son origine jusqu'à maintenant
-**gitk** : commande pour avoir une autre vue de son historique
-git show : affiche différents types d'objets

-git remote add : ajouter un nouveau dépôt Git à partir d'une URL
-git remote add [nomcourt] [url] : ajoute un nouveau dépôt distant Git comme nom court auquel il est facile de faire référence
Exemple : git remote add *''dossier''* git@gitlab.com:52318/my-first-awesome-project.git (ajoute le répertoire *''My-first-awesome-project''* disponible sur GitLab sur Git Bash et le renomme *''dossier'')*
-git remote -v : Vérifie que l'URL entrée pour le transfert de fichier est correcte
-git remote set-url *''URL''*: commande qui permet de modifier l'URL si on l'a mal entré une première fois
Exemple :
git remote set-url esi-gitlab *https://cette-fois.il/faut/pas/se/planter*
-git remote update : mettre à jour la base d'URL depuis la redirection
-git push : Commande qui permet de pousser le dépôt en amont une fois qu'il est prêt à être partagé
-git clone : Clone un dépôt dans un nouveau répertoire
Exemple : git clone git@git.esi-bru.be:nrichard/HelloWorld pour pouvoir par exemple faire une copie d'un répertoire git de GitLab à Linux
-git pull : Récupère et intègre un autre référentiel ou une branche locale/ commande pour mettre un dépôt à jour

TD6 :

-**javac -d** *chemin ''Fichier.java''* : commande qui compile le fichier donné et place le résultat dans une hiérarchie de dossiers qui correspond au package à partir du chemin donné

Exemple : *javac -d ~/classes Test.java (*pour pouvoir compiler le fichier Test dans le dossier

classe)
-**java -cp** *chemin ''NomCompletdelaClasse''* : commande pour exécuter une classe. L'option cp (une abréviation pour classpath) indique le (ou les) endroit(s) où chercher les classes.
-echo $CLASSPATH : commande pour voir le contenu de la variable CLASSPATH
-export CLASSPATH : commande pour si elle est définie pour la première fois
-**CLASSPATH=valeur** : commande pour changer son contenu
-CLASSPATH=$CLASSPATH:valeur : commande pour ajouter un dossier à son contenu
→ Si la modification doit être permanente, vous pouvez placer la commande dans le fichier **.bashrc**

TD7 :

En Java, on retrouve ces trois fichiers :
-System.in pour 0 (entrée standard) qu'on retrouve dans la déclaration
→ Scanner clavier = new Scanner (System.in); .
System.out pour 1 (sortie standard) qu'on retrouve dans l'instruction
→ System.out.println();
-System.err pour 2 (erreur standard) qu'on retrouve dans l'instruction
→ System.err.println();

- ***« commande »*** **>** ***« NomDuFichier »*** : redirige la sortie (le résultat de la commande) vers le fichier correspondant.
Exemple : ls -l > liste
Ouvre l'ensemble des fichiers du home dans le fichier « liste.
- ***« commande »*** **<** ***« NomDuFichier »*** les lectures de la commande se feront dans le fichier et pas au clavier
Exemple : java g52318.td7.Multiple4 < Multiples.com
Utilisera les nombres du fichier Multiples.com pour le fichier Mutliple4 (qui affiche les multiples de 4)
-**Ctrl-d** : commande qui est l'équivalente de la marque « fin de fichier » pour le clavier, c'est ainsi que vous terminerez l'acquisition de la série de nombres au clavier (elle peut aussi fermer linux)
-**Ctrl-c :** arrête complètement le fichier
- ***« commande »*** **|** ***« commande »*** : tube qui permet d'enchainer les commandes
Exemple: -ls /home | more
Permet d'exécuter plus rapidement : ls /home grep puis more grep
- ***« commande »*** **2>** ***« NomDuFichier »*** **:** commande qui permet une redirection de la sortie vers une sorte d'erreur
Exemple : javac MalFaite.java 2>erreur
Permet de compiler le fichier MalFaite en spécifiant qu'il y a des erreurs, il suffit de faire more erreur pour que toutes les erreurs soient affichées dans le fichier erreur nouvellement créé pour l'occasion.

Les filtres Linux :

Un filtre est une commande Linux qui acquiert des données sur l'entrée standard et les envoie vers la sortie standard après les avoir éventuellement transformées.
Tr :
-**tr** : commande permet de convertir une chaîne de caractères
-**tr** ***« argument1 » « argument2 »*** < ***NomDuFichier*** : commande qui remplace tous les éléments (argument1) d'un fichier par un d'autres (argument2)
Exemple : *tr s Z < test7*
Remplace tous les s présents dans le fichier test7 par des Z.
Ou *tr ''[a-z]'' ''[A-Z]'' < test7*
Remplace tous les charactères en minuscule d'a jusque z en majuscule (de A jusque Z).
-tr -c : Tous les caractères qui ne sont pas spécifiés dans la première chaîne sont convertis selon les caractères de la seconde.
-tr -d « caractère » < NomDuFichier : commande qui efface le caractère spécifié
Exemple : tr -d "d" < test7
Efface tous les « d » écrits dans le fichier test7
-tr -s « caractère » < NomDuFichier : commande qui efface le caractère s'il se répète plusieurs fois de suite, il est réduit à une seule unité.
Exemple : -tr -s 'd'< test7
Efface tous les « d » du fichier test7 s'il est écrit plusieurs fois
Cut :

-cut : commande qui permet de récupérer/ignorer des caractères ou des champes d'une ligne
-cut -c(sélection_colonnes) [fichiers] : commande qui permet d'afficher le caractère présent à une certaine colonne d'un fichier.
Exemple : cut -c5 test7
Affiche tous les caractères présents à la cinquième colonne de chaque ligne (s'il est par exemple simplement écrit bonjour, la commande affichera n).
-cut -d(sep) -f* [fichiers ...] : commande qui permet d'ignorer tout un champ de caractère après un séparateur (à spécifier).
Exemple : cut -d? -f2 test7
Ignore tous les mots écrits après le ? (Et le ? lui-même) à la liche f2 (même s'il le numéro n'est pas bon, ça marche)
Sort :
-**sort** : commande qui permet de trier les lignes d'un fichier
-sort -b : commande qui saute les colonnes constituées de blancs
-sort -d : commande qui permet le trie de type dictionnaire
-sort -n : commande qui permet de trier par ordre numérique
-sort -f : aucune différentiation n'est faite entre minuscules et majuscules
-sort -b : commande qui ignore les espaces placés en début de champ
-sort -r : commande qui permet le trie inverse
-**sort -M** : commande qui trie chronologiquement les mois
-sort -t : : trie suivant les champs séparés par les caractères deux points (‘’ :’’)
Uniq:
-uniq : commande qui élimine les lignes dupliquées dans un fichier trié.
-uniq -**u** : commande qui n'affiche que les lignes uniques.
-uniq -**d :** commande qui n'affiche que les lignes dupliquées.
-uniq -c : commande qui affiche également le nombre d'occurrence de chaque ligne.
→ (Autres commandes uniq disponibles, voir sur internet)
-who : commande qui affiche toutes les connexions actives.
→ Commande qui affiche le nombre d'utilisateurs qui sont connectés à linux1 pour le moment :
who | cut -f 1 -d ’ ’ | sort | uniq | wc -l

TD8:

Symétrique :

-**openssl enc -aes-256-cbc -out *NomduFichier*.ciphered -in *NomDuFichier*** : commande pour chiffrer le fichier correspondant
-**openssl enc -aes-256-cbc -in NomduFichier.ciphered -d** : commande pour déchiffrer le fichier correspondant
→ ssh-keygen génèrera directement la clé publique au format OpenSSH.
→ openssl ne génèrera pas de clé publique automatiquement., mais il est possible de le lui demander. Dans ce cas, cette clé sera au format PEM.

Asymétrique :

Pour RSA 4096 bits la commande suivante créera deux fichiers ; NomDuFichier et NomduFichier.pub
-**ssh-keygen -t *NomDuFchier* -b 4096** : commande pour générer une clé asymétrique
-**openssl *NomDuFichier* -in *NomDuFchier* -pubout -out *NomDuFichie*r.pem :** commande pour créer un fichier contenant cette même clé au format PEM
- openssl pkeyutl -encrypt -pubin -inkey <la clé publique du voisin > -out encrypted-message.bin :
Commande pour chiffrez un message avec une clé publique et partagez le message chiffré.
-sha512sum NomDuFichier : commande pour vérifier l'intégrité d'un message avec SHA-512
-ssh -v g12345@linux1 : commande qui rend le programme beaucoup plus verbeux. Il affiche dans la console toute une série de commentaires
→ Si ça ne marche pas :
-ssh -v g12345@linux1 true 2>&1 |grep "server->client"
-ssh -v g12345@linux1 true 2>&1 |grep "client->server"
-ssh g12345@linux1 : commande pour se connecter à un serveur ssh depuis Git Bash
-ssh<user> @<host> : <port> : commande pours se connecter mais avec un port
Exemple : -ssh anakin@example.org:2222
-La connexion à linux1 en tant qu'utilisateur g12345 se fera par la commande :
-ssh labo
-ssh-copy-id -i @linux1 : commande pour copier la clé de Git Bash sur linux1.
(Si ça a marché, un fichier nommé /.ssh/authorized_keys sera normalement créé).
-Code d'échappement :
-~. demande une déconnexion au serveur ;
- . ~[Ctrl-Z] passe la connexion en background ;
-Pour la ramener en avant plan, **fg**, pour voir les tâches, **jobs.** Ramener une tâche spécifique **fg %**
Par défaut, une demande de connexion ssh procure un shell. Ce comportement n'est pas obligatoire. Il est possible de ne demander l'exécution que d'une seule commande avant déconnexion :
-ssh <user> @<host> <command>

À savoir :

→ Définition : Un commit est un enregistrement de l'état de votre répertoire de travail à un moment donné.
Dans un commit, les informations suivantes sont enregistrées :
1. L'état du répertoire de travail
2. L'auteur du commit
3. Le nom du commit qui précède (appelé aussi "commit parent")
→NB : pour pouvoir bien compiler un fichier java, il faut que le nom de la classe soit celui du fichier !
-Exemple : Pour le fichier Ex1.java écrire public class Ex1 puis la suite !
→Toujours faire git add pour un fichier avant de le commiter !
-Exemple :
-git add .gitignore
-git commit -m "Ignore le répertoire 'build'"

→ A chaque modification d'un fichier NetBeans, il ne faut pas oublier de faire git add pour que le changement soit enregistré dans Git Bash
→ Pour exécuter une méthode principale, il faut taper les commandes suivantes (exemple) :
- *find /eCours/java -name Color.class* (pour trouver la classe) ce qui devrait donner : */eCours/java/esi/util/Color.class*
-puis *java -cp /eCours/java esi.util.Color*
Ou (exemple) :
-mkdir -p g12345/util
-mv Color.class g12345/util
- java -cp . g12345.util.Color