- Un protocole est un ensemble de règles qui permettent aux ordinateurs de discuter entre eux (un peu comme si deux personnes devaient parler la même langue pour avoir une conversation. « openclassrooms »)
- Il existe des tonnes et des tonnes de protocoles pour communiquer par Internet, mais pour ce qui est d'accéder à la ligne de commande à distance, c'est-à-dire à la console, il y en a deux principaux.
    - **Telnet** : le protocole le plus basique, qui présente le gros défaut de ne pas crypter les données échangées entre vous et le serveur. Si un pirate « écoute » vos échanges par un moyen ou un autre, il pourrait récupérer des informations sensibles, en particulier votre mot de passe lorsque vous l'envoyez à la connexion. Ce moyen de connexion reste utilisé mais peu par rapport à SSH.
    - **SSH** : c'est de très loin le protocole le plus utilisé (et que l'on préfère) car il permet de crypter les données et de sécuriser ainsi la connexion avec le serveur.
- Par convention :
    - Les paramètres qui ne contient qu'une seule lettre commence par un seul tiret «- » peuvent être combiner (-a -b peuvent être combiner : -ab) .
    - Les paramètres qui contient plusieurs lettres commence par deux tirer «--» et ne peuvent pas être combiner.
- Ctrl + L : Effacer la console
- Ctrl + D : envoie EOF à la console, si on utilise cette raccourci sans rien taper avant, la console sera fermée

Commandes Générales LINUX :

- pwd : print working directory
- which cmd : connaitre le chemin vers le bin de cmd
- ls : lister le contenu du dossier actuel :
    - -a : afficher fichier cacher
    - -F : ajouter «/» à la fin des repertoire , @ à la fin des liens pour distinguer les répertoires , les fichiers , et les liens.
    - -l affiche plus de détails (droits, pour un fichier : nbre lien physique (nbre de fichier qui partage le même inode) pour un dossier : le nombre de fichier à l'intérieur, propriétaire, groupe, taille (en octets), date de dernière modification, nom du fichier ou dossier
    - -h (human readable) affiche la taille en o/ko/mo/Go (dépend de la taille) (doit être utilisé avec –l)
    - -t trie par date de modification
    - -r renverser l'ordre de l'affichage
    - -s afficher taille en bloc
    - -S trie par taille
- du : afficher le contenu de dossier actuel + les tailles en KO
    - -h pour une taille lisible
- stat filename : afficher le status d'un fichier (nom,size….)
    - stat -c %s filename : afficher la taille d'un fichier en O
- cat
    - cat fichier : afficher le contenu
    - cat > fichier
    - cat >> fichier

- less (et more)
  - affiche le contenu d'un fichier page par page
  - après l'ouverture du fichier : /mot_a_rechercher (occurrence suivante : n , occ précédente : N)
  - la différence entre les deux est que more est très vieux et moins évolué que less, il est donc recommandé d'utiliser less.
- tail/head : afficher la fin/le début d'un fichier ; -n permet de limiter le nombre de lignes à afficher -f (pour tail) permet d'afficher « en direct » les modifications du fichier (s'il y en a) , -s permet de definir la fréquence de la verification des changements
- sed –n kp nom_fichier pour obtenir la ligne k du fichier
- touch filename: crée un fichier s'il n'existe pas , sinon seulement sa date de modification est changé.
- mkdir : crée dossier
  - -p pour créer un dossier dans un chemin qui n'existe pas (exemple : mkdir -p dossier1/dossier2/dossier3 créera trois dossiers
- cp fichier nv_nom (copier un fichier dans la même répertoire , en lui donnant un nouveau nom)
  - -r permet de copier un dossier et son contenu (non vide)
- mv nom_fichier_ou_dossier chemin (déplacer un fichier/dossier (vide ou non vide) à un dossier)
  - mv ancien_nom nv_nom (renommer)
- rm : supprimer un fichier (ou dossier avec le parametre -r)
  - r ou R : supprimer un dossier et son contenu récursivement
  - f : force : ignorer les erreurs
  - d : supprimer un dossier vide
  - i pour demande la confirmation avant suppression de chaque fichier
  - v pour afficher les information de chaque opération
- rmdir : supprimer un dossier (doit être vide)
  - -p chemin, pour supprimer tous le chemin rmdir –p a/b/c supprimera c puis b puis a (tous ces dossiers doivent être vide)
- ln : Créer des liens
  - types des liens en linux :
    - (sans parametre) Lien physique : raccourci vers l'inode (le contenu du fichier) : la suppression d'un des fichiers n'affectera pas l'autre
    - (-s) Lien symbolique : raccourci vers le fichier , la suppresion de fichier original rendera le lien inutilisable (lien mort)
- sudo (**S**ubtitue **U**ser **D**o) : devenir root pour un instant , exécuter la commande et quitter
  - sudo su rester en mode root indéfiniment
  - « su - ». L'ajout du '-' a pour effet de rendre accessibles certains programmes destinés seulement à root. Par ailleurs, cela vous place directement dans le dossier personnel de root (/root).
  - exit pour quitter le mode su après l'exécution de su

Gestion des utilisateurs :

- adduser nom (va demander le mot de passe et quelque info)
  - --home REP pour définir REP comme répertoire utilisateur

    - –-no-create-home
    - –-group group_name
    - –-ingroup
- useradd username puis on écrit passwd username pour définir un mdp (sinon le compte reste inaccessible)
    - –create-home pour créer home directory (ou bien –m)
    - –g group
    - –G group1,group2
    - –home Repertoire_perso (-d)
- pour créer un utilisateur sans les deux commandes ci-dessus , on doit suivre les 3 étapes :
    - echo "username :x :userId :groupId :,,,:userRepertory:/bin/bash" >> /etc/passwd
    - echo "groupName :x :groupid :users " >> /etc/group
    - passwd username
- passwd nom (va demander le nouveau mot de passe , et aussi l'ancien si on est pas en root)
    - si on ajoute pas le nom , on change le mdp de l'utilisateur actuel
- deluser nom
    - --remove-home : pour supprimer le fichier par défaut de l'utilisateur aussi
    - --remove-all-files : supprimer toutes les propriétés de l'utilisateur
- deluser nom group :
    - delete user from a specific group
- Usermod : modifier un utilisateur
    - –l nouveau_nom ancien_nom renommer l'utilisateur
    - –g modifier le groupe officiel de l'utilisateur
    - –G grp,grp2,grp3… mettre un utilisateur dans plusieurs groupe
    - –aG pour affecter l'utilisateur à des nouveaux groupes (sans lui retirer des anciens)
- groups : affiche le nom de tous les groupes
- groups nom_utilisateur : affiche le nom des groupes auxquelles appartient l'utilisateur « nom_utilisateur »
- sudo chattr +i : protection contre la suppression

Gestion des propriétés :

- chown nv_proprietaire fichier : changer seulement l'utilisateur propriétaire.
    - chown nv_propr : nv_grp fichier : pour changer les deux
    - –R U_prop:grp_prop chemin_dossier : modifier le propriétaire du dossier et de tout son contenu
- chgrp nv_grp_proprietaire fichier : définir un nouveau propriétaire pour un fichier

Gestion des droits :

- chmod droits fichiers (4=r , 2=w , 1=x)
    - 777 ⇔ u=rwx,g=rwx,o=rwx
    - Pour modifier sans écraser les anciens droits :
        - Chmod u=rwx,g-r,o+r : affect tous les droits à l'utilisateur , enlève le droit de lecture au groupe et ajoute le droit de lecture à « other »
    - Chmod –R pour modifier récursivement le droit d'un dossier et son contenu
- Pour un dossier :
    - r : lister le contenu du fichier (ls)

    - w : modifier le dossier et son contenu
    - x : ouvrir le dossier
- pour un fichier :
    - r : ouvrir le fichier (read only)
    - w : le modifier
    - x : l'exécuter (seulement si c'est un exécutable comme script)

NANO :

- **-m** : autorise l'utilisation de la souris sous Nano. En console, oui, oui. Vous pouvez vous en servir pour cliquer avec votre souris sur la zone de texte où vous voulez placer votre curseur.
- **-i** : indentation automatique. L'alinéa (tabulations) de la ligne précédente sera respecté lorsque vous irez à la ligne. Très utile lorsque vous éditez un fichier de code source.
- **-A** : active le retour intelligent au début de la ligne. Normalement, lorsque vous appuyez sur la touche Origine (aussi connue sous le nom de Home) située à côté de la touche Fin, le curseur se repositionne au tout début de la ligne. Avec cette commande, il se positionnera après les alinéas. Comme -i, il s'agit d'une option utile avant tout pour les programmeurs.

Alias :

- alias nom_alias='commande' (sans espace)

Recherche :

- Locate fichier : chercher fichier (les fichiers récemment crées ne sont pas découverts par locate, car elle ne cherche pas dans le HDD mais dans la base de données des fichiers)
- find « où » « quoi » « que faire avec les resultats» fichier : cherche fichier (tous les fichiers sont découverts par find) :
    - find chemin_dossier_où_chercher **–name** "nom_fichier " (guillemet obligatoire si on veut utiliser le joker : *)
    - –type d ou f (fichier ou dossier)
    - –maxdepth (1 = recherche seulement dans le dossier, 2 = rechercher aussi dans les sous dossier , 3 = dans les sous sous dossier aussi.)
    - -mindepth
    - Que faire avec le resultat : -print (par défaut) , -delete pour supprimer , -exec commande `{}` \ ; pour exécuter une commande (chmod par exemple)
    - Exemple : trouver tous les fichiers qui commence par « arithmetic » et les deplacer
    - find /home/youssef/Images -name "arithmetic*" -exec mv {} /home/youssef/ \ ;
- tree repertory_name –L profondeur
- grep [paramètre] "texte" fichier
    - –i : ignorer la casse
    - –n : afficher aussi le numéro de la ligne correspondante
    - –v : cherche les lignes qui ne contiennent pas le texte rechercher
    - –r : (récursive) avec ce paramètre, on donne un dossier au lieu d'un fichier, grep va chercher dans tous les fichiers de ce dossier et de ses sous dossiers
    - –E : Pour utiliser les regexp (expressions réguliers)

        - Le parametre est optionnel, Ubuntu comprend les regex sans ce parametre , c'est pour les anciens systems d'UNIX
    - –l (un L minuscule ☺) option is used to only print filenames of matching files, and not the matching lines (this could also improve the speed, given that grep stop reading a file at first match with this option).
    - –w pour chercher un mot exact (exemple quand on cherche « az » sans utiliser ce paramètre ça va nous afficher tous les mots qui contient az , si on ajoute le paramètre –w on aura que les lignes(ou fichiers) qui contient le mot « az »
- egrep '(mot1|mot2) mot3 ' : chercher les lignes qui contient (mot1 ou mot2) suvie par mot3

Manipulation des fichiers :

- sort [parametre] fichier : afficher les lignes du fichier trier alphabétiquement :
    - –o pour mettre le resultat dans un fichier :
        - Sort –o [fichier_trié(resultat)] fichier_à_trier : si on précise pas « fichier_trié » le fichier_à_trier sera modifié
    - –r inverser le trie
    - –R trie aléatoire
    - –n comparer selon la valeur numérique de la chaîne (sans –n : ordre ASCII)
- wc (word count) :wc [parametre] fichier : compter le nombre de ligne, de mots et de caracteres dans un fichier
    - –l : compter le nombre de ligne seulement
    - –w :compter le nombre de mot seulement
    - –c : compter le nombre d'octet seulement
    - –m : compter le nombre de caractère seulement
    - affiche par défaut : nbre_ligne nbre_mot nbre_byte nom_fichier
- uniq fichier : afficher le contenu de fichier en supprimant les lignes en double (ne repère que les lignes successives)
    - uniq fichier fichier_où_écrire_le_resultat
    - –c compter le nombre d'occurrence de chaque ligne
    - –d n'afficher que les lignes qui sont en double
- cut :
    - première utilisation : couper une partie des lignes d'un fichier en indiquant les numéros des caractères à garder
        - cut –c n-m : (m>n) afficher les lignes du ficher à partir du n eme caractere jusqu'au m eme
            - cut –c n- : afficher à partir du nième caractère
        - cut –c –m afficher jusqu'au m eme caractère
    - deuxième utilisation : couper des colonnes d'un fichiers, en indiquant le délimiteur des colonnes et le nombre de colonnes à afficher :
        - cut –d delimiteur –f le(s)_colonne(s)_à_afficher :
            - cut –d delimiteur –f num_colonne
            - cut –d delimiteur –f 1,2,3..
            - cut –d delimiteur –f n-m : du n eme au n eme colonne
            - cut –d delimiteur –f n- : du n eme jusqu'au dernier colonne

Redirection du contenu :

- rediriger le retour de la commande vers un fichier :
    - commande > fichier écraser le contenu du fichier pour mettre la commande
    - commande >> fichier ajouter le retour de la commande à la fin
- rediriger les erreurs retourner par la commande vers un fichier :
    - commande 2> fichier (log des erreurs par exemple)
    - commande 2>> fichier
- rediriger les deux type de retour vers un seul fichier
    - commande > (ou >>) fichier 2>&1

Lire les paramètres d'une commande depuis un fichier ou depuis le clavier :

- lire les paramètres de la commande depuis le fichier
    - commande `< fichier` (avec ` = ALT Gr+7)
- lire plusieurs lignes progressivement (ligne par ligne , tapez entrer pour une nouvelle ligne)
    - ~~commande << mot_pour_s'arreter~~
        - ~~mot_pour_s'arreter = le mot que vous devez taper pour terminer~~

Chainer les commandes : **cette opération n'est pas si simple comme elle peut apparaitre et ne marche pas toujours comme prévu … pour plus d'information cliquez ici**

- envoyer le retour d'une méthode à une autre : commande1 | commande2 ..
    - le retour de commande1 sera traité par commande2
    - ~~(!) si vous voulez que ça marche toujours ajouter xargs avant commande 2 :~~
        - ~~commande1 | xargs commande2~~

Surveillance du system :

- w : affiche les détails des personnes connectés au system.
- who : presque la même chose que w
- uptime : depuis quand de fonctionnement du system.
- ps : lister les procesuss actif (statique)
    - sans parametre : afficher les processus de l'utilisateur actuel
    - –f affiche plus de details (colonnes)
    - –e affiche tous les processus actifs (même ceux des autres utilisateurs…)
    - –ef pratique pour tout afficher et en details
    - -ejH : afficher les processus en arbre (processus père – processus fils)
    - –u username : aff procesuss de l'utilisateur
- top : lister les processus actif (en temps réel)
- halt : eteindre
- reboot
- kill PID
    - -9 : forcer l'arret
- killall nom_processus

// Fonction C :

lstat :

#include <unistd.h>

#include<sys/stat.h>

Déclarer une structure de type stat :

struct stat nom

lstat(chemin_fichier,&nom)

printf("la taille est %d octet",nom.st_size) ;

printf("ID propriétaire est %s",nom.st_uid) ;

printf("ID groupe propriétaire est %s" ,nom.st_gid) ;

la structure du type « struct stat » :

```
struct stat {
   dev_t     st_dev;      /* ID du périphérique contenant le fichier */
   ino_t     st_ino;      /* Numéro inœud */
   mode_t    st_mode;     /* Protection */
   nlink_t   st_nlink;    /* Nb liens matériels */
   uid_t     st_uid;      /* UID propriétaire */
   gid_t     st_gid;      /* GID propriétaire */
   dev_t     st_rdev;     /* ID périphérique (si fichier spécial) */
   off_t     st_size;     /* Taille totale en octets */
   blksize_t st_blksize;  /* Taille de bloc pour E/S */
   blkcnt_t  st_blocks;   /* Nombre de blocs alloués */
   time_t    st_atime;    /* Heure dernier accès */
   time_t    st_mtime;    /* Heure dernière modification */
   time_t    st_ctime;    /* Heure dernier changement état */
};
```

stat() récupère l'état du fichier pointé par path et remplit le tampon buf.

lstat() est identique à stat(), sauf que si path est un lien symbolique, il donne l'état du lien lui-même plutôt que celui du fichier visé.

fstat() est identique à stat(), sauf que le fichier ouvert est pointé par le descripteur fd, obtenu avec open(2).