OS : operating system

Gestionnaires de bureau (gestion du mode graphique de Linux) : Unity, Gnome, KDE, XFCE

Penser à défragmenter mon disque dur

GNU GRUB : GRand Unified Bootloader

ls : lister les fichiers et dossiers du répertoire actuel

a == **all** : afficher tous les fichiers et dossiers, même cachés

Les raccourcis

Ctrl + L == clair

Ctrl + D == exit : envoie message EOF (si le terminal est vide, cela fermera la console)

Ctrl + A == revenir au début de la ligne de commande

Ctrl +E == ramener le curseur à la fin de la ligne de commande

Ctrl + U == supprime tout ce qui se trouve à gauche du curseur (curseur exclu)

Ctrl + K == supprime tout ce qui se trouve à droite du curseur (curseur inclus)

Ctrl + W == supprime le 1er mot à gauche du curseur (ie séparé par des espaces du reste)

Ctrl + Y == colle ce qu'on vient de supprimer avec Ctrl + U, K,W

Séparateur de dossier : \(backslach) sous windows et

/(slach) sous linux

(majuscule # minuscule)

Les paramètres de la commande ls :

ls -a : lister tout (même les fichiers cachés)

ls -F : donner aussi le type du document (via indicateurs)

ls -l : liste détaillée (avec les droits, la taille en octets)

ls -h : affiche la taille en Ko, Mo, Go

ls -t : trie par date de dernière modification

ls -r : renverse l'ordre d'affichage des fichiers

ls -larth : lister tout, dans l'ordre des fichiers les plus anciennement modifiés avec les droits et les tailles en Ko,Mo ou Go

Commande **du** (disk usage) : donne la taille de l'espace occupée dans le disque dur par chaque élément du repertoire

Les paramètres de la commande du :

Du -h : afficher la taille en Ko,Mo,Go

Du -a : afficher aussi la taille de l'espace occupé des fichiers (contenus dans les sous-dossiers)

Du -s : juste la taille totale

cat : afficher tout le contenu d'un fichier

cat -n : pour afficher le numéro de la ligne

less : affiche le contenu d'un fichier page par page

head : afficher les 1ères lignes d'un fichier

head -n nbreDeLignes : permet d'afficher les nbreDeLignes premières lignes

tail : afficher les dernières lignes d'un fichier

tail -f : pour suivre l'évolution du ficher (toutes les secondes par défaut, mais le paramètre [-s nbreSec] pour changer la frequence) ex : tail -f -s 3 => mise à jour toute les 3s

touch : créer un fichier

mkdir : créer un dossier

mkdir -p animaux/vertebres/chat : crée tous les dossiers intermédiaires à la fois

cp : copier les fichiers

cp -R : copier des dossiers

mv : déplacer un fichier, un dossier / renommer un fichier

rm : supprimer un fichier

rm -i : demande de confirmation

rm -f : forcer la suppression

rm -v : dire ce qu'elle fait (en occurrence « supprimer le fichier »)

rm -r ; supprimer un dossier (**rmdir** ne peut supprimer un dossier que s'il est vide)

ln : créer un lien physique entre les fichiers (même contenu) == allias

ln -s : créer un lien symbolique entre fichier (raccourci)

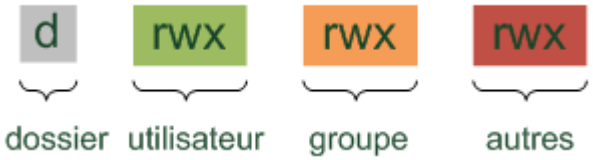
chown : changer le propriétaire d'un fichier (se mettre en mode superUser : sudo su)

chgrp : changer le groupe propriétaire d'un fichier

chown user**:**group : changer à la fois le propriétaire et le groupe propriétaire d'un fichier

chown -R : pour changer la propriété sur un dossier

chmod : modifier les droits d'accès d'un fichier



chmod -R : modifier les droits d'accès d'un dossier (tous les fichiers qu'il contient avec)

Editeur de texte Nano

Les paramètres de la commande **nano** :

-m : pour avoir la souris

-i : pour respecter l'aliéa

-A : revient au tout début du texte en respectant l'alinéa

Configurations : **set**, **unset**

Télécharger des logiciels

Dépôt Ubuntu en France : `fr.archive.ubuntu.com`

Dépôt orange : `ftp.oleane.net`

Dépôt free : `ftp.free.net`

Programmes consoles de gestion de paquets : apt-get, aptitude…

Les commandes apt-get/cache(en mode root)

apt-get update : mettre à jour le cache des paquets

apt-cache search monpaquet : rechercher un paquet

apt-cache show monpaquet : ample description d' un paquet

apt-get install monpaquet : télécharger et installer un paquet

apt-get autoremove monpaquet : supprimer un paquet

apt-get upgrade : mettre à jour tous les paquets

RTFM

man / -h (--help) : afficher le manuel d'une commande

apropos : retrouver une commande à partir d'un mot clé

whatis : donne l'entête du man (définition de la commande)

Rechercher un fichier ou un dossier

locate : trouver un fichier ou un dossier sur la base de données (mais pas les fichiers récemment crées : défaut)

slocate : comme locate mais cache les fichiers dont on n'a pas le droit de lecture

sudo **updatedb** : pour forcer la mise à jour de la data base et réparer le défaut de locate

find [où] quoi [que faire avec]

où ? : la commande find recherche par défaut dans le répertoire actuel.

quoi ? : Avec find, on peut faire une recherche par le nom (-name), la taille (-size), la date de dernière modification (-atime), les fichiers uniquement (-type f), les dossiers uniquement (-type d)

Que faire avec ? : -print , -printf, -delete

-exec : Appeler une commande : (exemple1 : find -name '' *.cpp '' -exec chmod 600 {} \;)

- La commande se termine toujours avec un \
- Les **{}** seront remplacés par le(s) nom(s) du(des) fichier(s)

(exemple2 : find -name ''*.jpg'' -exec mv {} images \;

Contrôler les processus et les flux de données

grep : **filtrer** des données [ie en récupérer une partie] (sensible à la casse)

grep -i : ne pas tenir compte de la casse

grep -n : connaître le numéro des lignes

grep -v : donner toutes les lignes où l'expression tapée n'existe pas

grep -r (rgrep): rechercher dans un dossier entier (liste des fichiers dans lequel le mot recherché apparaît)

grep -E (egrep) : pour utiliser les expressions régulières

sort : **trier** [ie ranger dans un ordre précis] (alphabétiquement) des lignes (pas sensible à la casse)

sort -o : enregistrer le tri dans un fichier

sort -r : trier en ordre inverse

sort -R : trier aléatoirement

sort -n : trier des nombres

wc : compter dans l'ordre le nombre de lignes – mots - octets

wc -l : compter le nombre de lignes uniquement

wc -w : compter le nombre de mots uniquement

wc -c : compter le nombre d'octets

wc -m : compter le nombre de caractères

uniq : **affiche** supprimer les doublons successifs (faut donc trier avant) et enregistrer dans un autre fichier (optionel)

uniq -c : compter le nombre d'occurrences

uniq -d: afficher uniquement les lignes en doublons

cut : couper et afficher une partie du fichier

cut -c : délimiter la coupure sur une ligne (en octets)

CSV : *Comma Separated Values*

-d : indique quel est le délimiteur dans le fichier

-f : indique le numéro du ou des champs à couper.

Exemple : cut -d , -f 1,2 notes.csv

Ecrire le résultat d'une commande dans : la console, un fichier ou en paramètre d'une autre commande

> : écrire le résultat dans un autre fichier (écrasé s'il existe déjà et crée dans le répertoire actuel [par défaut] sinon)

>> : pour rédiger à la fin d'un fichier

NB : corbeille de linux : **/dev/null**

2> : rediriger les erreurs dans un fichier à part (autre que la sortie standard)

Exemple : cut -d , -f 1 fichier_inexistant.txt > eleves.txt 2> erreurs.log

2>> : pour écrire à la fin du fichier d'erreurs.

2>&1 : rediriger le résultat et/ou les erreurs dans le même fichier (et de la même façon)

- Exemple : cut -d , -f 1 fichier_inexistant.txt >> eleves.txt 2>&1

(ie tout envoyer à la fin du fichier eleves.txt, même les erreurs s'il y en a)

Le Shell : le programme qui gère la console

< : lire depuis un fichier

<< unMot : lire la saisie du clavier. **unMot** pour indiquer la fin de la saisie (il faut retaper le même mot lorsqu'on a terminé la saisie.

Exemple :
sort -n << **FIN** > nombres_tries.txt 2>&1
> 18
> 27
> 1
> **FIN**

| (le païpe): enchaîner les commandes (la sortie de la commande précédente est l'entrée de la suivante)

Exemple : cut -d , -f 1 notes.csv | sort

Surveiller l'activité du système

w : pour savoir qui fait quoi (état du système)

ps : donne une liste statique des processus qui tournent sur la machine avec leur PID (affiche uniquement les processus lancés par le même utilisateur sur la même console)

ps -ef : affiche exactement tous les processus lancés sur la machine

top : affiche une liste dynamique des processus lancés sur la machine

ctrl + C : arrêter un processus lancé en console

kill (+numeroPID) : tuer un processus

kill -9 (+numeroPID) : tuer un processus sans lui donner le temps de se terminer proprement

killall (+nomDuProcessus) : tuer tous les processus portant ce nom

(en mode root)

shutdown : arrêter, redémarrer ou mettre hors tension la machine

halt : arrêter l'ordinateur

reboot : redémarrer l'ordinateur

Exécuter des programmes en arrière-plan

& : à la fin de la commande pour l'exécuter en tache de fond (ie qu'on peut faire autre chose sur la console le temps). **Problème** : le processus en arrière-plan reste « attaché » à la console. Si on ferme la console sur laquelle on est, le processus sera tué et ne s'exécutera donc pas jusqu'au bout.

nohup : au début de la commande pour détacher le processus (qu'on veut mettre en arrière-plan) de la console (ie immuniser le processus à la fermeture de la console)

&

⇔

Si on a oublié dès le départ de mettre le « & » à la fin de la commande, on fait :

{**Ctrl + Z** : pour mettre en pause le programme et récupérer l'invite de commandes

bg : pour que le processus continue à tourner mais en arrière-plan}

fg %(num_en_arrière_plan) : remet un processus en premier plan

jobs : connaître les processus qui tournent en arrière-plan

NB : **Screen** est une application qui permet d'émuler plusieurs consoles dans une seule.

- **screen** : créer une nouvelle session
- **screen -S** nom: créer et nommer une nouvelle session
- **ctrl+D** ou **exit** ou **ctrl+a** puis \: tuer la session screen
- **screen -r** num_du_pid : recupérer une session là où on l'a laissé (sans l'avoir tué)
- **screen -ls** pour afficher la liste des sessions actives.

Les principales commandes de Screen (il faut relâcher le clavier après Ctrl+a)

Ctrl + a puis **?** : pour afficher toutes les commandes.

Ctrl + a puis **c** : créer une nouvelle « fenêtre ».

Ctrl + a puis **w** : afficher la liste des « fenêtres » actuellement ouvertes.

Ctrl + a puis **A** : renommer la fenêtre actuelle.

Ctrl + a puis **n** : passer à la fenêtre suivante (next). [une étoile apparaît sur la fenêtre pointée]

Ctrl + a puis **p** : passer à la fenêtre précédente (previous).

Ctrl + a puis **Ctrl + a** : revenir à la dernière fenêtre utilisée.

Ctrl + a puis un chiffre de **0 à 9** : passer à la fenêtre n° X. Ctrl + a puis " : choisir la fenêtre dans laquelle on veut aller.

Ctrl + a puis **k** : fermer la fenêtre actuelle (kill).

Ctrl + a puis **S** : coupe l'écran en plusieurs pour afficher plusieurs consoles à la fois (split) (**Ctrl + a** puis **Tab** : pour passer dans l'autre compartiment. Une fois le curseur placé dans le compartiment, on peut soit créer une nouvelle fenêtre (**Ctrl + a** puis **c**) soit appeler une autre fenêtre que vous avez déjà ouverte (avec **Ctrl + a** puis un chiffre, par exemple)).
Ctrl + a puis **X** : fermer le split

Ctrl + a puis **d** : détacher screen de la console normale. On revient sur la console normale (par opposition à la console émulée) pendant que les programmes sur screen tournent toujours (comme nohup).

Exécuter une commande à une heure différée

at hh:mm MM/JJ/YY : exécuter des tâches (jobs) en différé (faire un **Ctrl+D** pour arrêter la liste des commandes : un <EOT> s'affiche alors).

atq : lister les tâches en attente.

atrm num_du_job : supprimer une tâche en attente.

Sleep nbre_tempsUnité : attendre avant d'exécuter la commande suivante.

crontab : gérer la liste des programmes à exécuter régulièrement.

crontab -e : modifier la crontab ;

crontab -l : afficher la crontab actuelle ;

crontab -r : supprimer votre crontab. Attention, la suppression est immédiate et sans confirmation.

Syntaxe **:** m h dom mon dow command

m : minutes (0 - 59) ;

h : heures (0 - 23) ;

dom (day of month) : jour du mois (1 - 31) ;

mon (month) : mois (1 - 12) ;

dow (day of week) : jour de la semaine (0 - 6, 0 étant le dimanche) ;

command : c'est la commande à exécuter.

- NB : l'argument * à la place d'un nombre signifie que toutes les valeurs sont valables

Transférer des données à travers un réseau

Archiver les fichiers

Tar : regrouper les fichiers dans une archive (avant de compresser)

NB : il faut d'abord rassembler tous les fichiers dans un dossier.

Tar -cvf nom_de _l_archive**.tar** nom_du_dossier_de_rassemblement**/** : créer l'archive du dossier_de_rassemblement

Tar -tf nom_de _l_archive**.tar** : afficher le contenu de l'archive sans l'extraire

Tar -rvf nom_de _l_archive**.tar** nom_du_fichier : ajouter un fichier à l'archive

Tar -xvf nom_de _l_archive**.tar** : extraire les fichiers de l'archive

Compresser les fichiers

gzip et **bzip2** : pour compresser une archive

gzip nom_de _l_archive**.tar** : compresser une archive

gunzip nom_de _l_archive_compressée**.tar.gz :** décompresser une archive

bzip2 nom_de _l_archive**.tar** : compresser une archive

bunzip2 nom_de _l_archive_compressée**.tar.bz :** décompresser une archive

NB: On peut en même temps archiver et compresser avec tar (qui appelle lui-même gzip ou bzip2).

Tar -zcvf nom_de _l_archive_à_compresser**.tar** nom_du_dossier_de_rassemblement**/** : archiver et compresser avec gzip

Tar -zxvf nom_de _l_archive_à_décompresser**.tar.gz** : extraire d'une archive avec gzip

Tar -jcvf nom_de _l_archive_à_compresser**.tar** nom_du_dossier_de_rassemblement**/** : archiver et compresser avec bzip2

Tar -jxvf nom_de _l_archive_à_décompresser**.tar.bz** : extraire d'une archive avec bzip2

NB : **unrar** et **unzip** pour décompresser resp. les .rar et les .zip

La connexion sécurisée à distance avec SSH

NB : Si le serveur SSH n'est pas sur le **port 22**, il faut indiquer le port avec l'option **-p** ou **-P** selon la commande.

SSH : Secure **Sh**ell : échanges sécurisés entre client et serveur. Le serveur génère une paire de clés de cryptage asymétrique, envoie la clé publique au client en clair. Seul le serveur connait la clé privée. Ensuite, le client envoie la clé symétrique choisit et cryptée (à l'aide de la clé publique précédente), qui va servir pour tous les échanges avenirs. Ainsi, le cryptage asymétrique ne sert qu'à communiquer la clé des communications entre le client et le serveur. Les échanges suivants ne se font pas en asymétrique car ce type de cryptage nécessite trop de calculs complexes.

ssh login**@**ip : se connecter à distance (à partir d'une machine Linux) à une session d'un serveur Linux

- ip peut être l'IP internet du serveur (si on se connecte depuis un autre réseau) ; l'IP local du serveur (si on est sur le même réseau) ; ou alors 127.0.0.1 ou simplement le mot localhost (si on est sur la même session Linux).

  ifconfig : permet d'obtenir l'IP local d'une machine.

logout : ou **ctrl+D** : se déconnecter à distance

echo : permet de saisir et d'afficher sur la sortie standard

Transférer les données à travers le réseau

wget : télécharger un fichier à partir de son adresse web

wget -c : reprendre un téléchargement arrêté

wget - -background : lancer un téléchargement en tâche de fond

scp : copier un fichier d'un ordinateur à un autre et inversement (ie récupérer un fichier d'un autre ordinateur)

ftp : se connecter à un serveur FTP (File Transfert Protocol)

[le login c'est **anonymous** pour les serveurs ftp publics]

ftp -p : entrer en mode passif pour le transfert des fichiers (ça évite l'erreur 500)

put nomDuFichier : envoie des fichiers vers le serveur FTP (impossible sur les serveurs publics)

get nomDuFichier : téléchargement des fichiers depuis un serveur FTP

NB : Sur un serveur FTP, les **commandes** précédées d'un **!** permettent de s'adresser au PC client

sftp login**@**ip : un FTP sécurisé

rsync : synchroniser un backup avec le dossier source.

Analyser le réseau et filtrer le trafic avec un pare-feu

NB : port 22 = SSH , port 21 = FTP, port 80 = web

Compiler un programme à partir du code source

Etape 1 : installer **build-essential**

Etape 2 : télécharger le .tar et le décompresser, exécuter le **. /configure** , faire **make**

Etape 3 : taper la commande **sudo make install (**sudo make uninstall pour déinstaller le programme**)**

Vim : l'éditeur de texte du programmeur

vim : pour lancer vim (comme nano)

Les commandes dans l'éditeur de texte Vim

i : insérer du texte

h, j, k, l : se déplacer dans tous les sens

0 : mettre le curseur au début de la ligne

$: mettre le curseur à la fin de la ligne

w : se déplacer de mot en mot

(nombre)**x** : effacer (nombre) lettres partant de la position du curseur

(nombre)**dd** : effacer (nombre) lignes partant de la position du curseur

(nombre)**dw** : effacer (nombre) mots partant de la position du curseur

d0 : supprimer tout, du curseur jusqu'au début de la ligne

d$: supprimer tout, du curseur jusqu'à la fin de la ligne

v : sélectionner du texte [ensuite **U** et **u** permettent de mettre le texte tout en majuscule ou minuscule , **y** : copier le texte sélectionné]

yy : copier la ligne actuelle en mémoire (papier-presse)

(nombre)**p** : coller (nombre) fois

r(lettre) : remplacer la lettre sous le curseur par (lettre)

r(mot) : remplacer par (mot) partant de la lettre sous le curseur

(nombre)**u** : annuler les (nombre) dernières modifications

Ctrl + R : annuler une annulation (:p)

Ctrl + G : savoir le numéro de la ligne où se trouve le curseur

G : aller à la fin du fichier

gg : revenir à la 1ere ligne du fichier

(nombre)**G** : aller à la (nombre)ième ligne du fichier

:w nomDuFichier : enregistrer le fichier

:r nomDuFichier : coller le contenu de nomDuFichier dans le fichier courant (à partir de la position du curseur)

:q : quitter

:wq : enregistrer puis quitter Vim

:sp (autreFichier): pour découper l'écran horizontalement afin d'ouvrir plusieurs (autreFichier) [**Ctrl + w + q** pour les fermetures]

:vsp (autreFichier): pour découper l'écran verticalement afin d'ouvrir plusieurs (autreFichier)

:!(commande) : lancer une (commande) externe (comme si on était sur le shell

/(mot) : rechercher (mot) [ensuite, appuyer **n** pour naviguer dans les occurrences plus basses, ou **N** dans le sens contraire].

:s/ancien**/**nouveau : remplace la première occurrence [de ancien par nouveau] de la ligne où se trouve le curseur ;

:s/ancien**/**nouveau**/g** : remplace toutes les occurrences de la ligne où se trouve le curseur ;

:#,#s/ancien**/**nouveau**/g** : remplace toutes les occurrences dans les lignes n° # à # du fichier ;

:%s/ancien**/**nouveau**/g** : remplace toutes les occurrences dans tout le fichier.

Ecrire un script shell (fichier)

- 1ere ligne : **# !/bin/x** avec x = bash, sh, ksh, zsh [la ligne du sha-bang : elle permet d'indiquer sous quel shell (x) on souhaite exécuter le programme]
- 2è ligne… : il faut ensuite écrire les commandes à exécuter

  NB : les commentaires commencent par **#**

- On exécute le script comme n'importe quel programme en tapant :
  . /nomDuScript.sh (il faut se trouver dans le répertoire contenant le fichier nomDuScript.ch ou alors indiquer le chemin de nomDuScript)

NB : A défaut, il faut simplement placer le fichier nomDuScript.sh dans un des répertoires du PATH (listés en tapant « echo $PATH ») et ensuite taper simplement à la console le nomDuScript pour que le fichier s'exécute (où qu'on se trouve)

bash -x nomDuScript.sh : pour déboguer le script

Afficher et manipuler des variables dans un script shell

nomVariable**=**valeurVariable : déclaration d'une variable

NB : les chaînes de caractères s'écrivent dans les **'** . Et l'apostrophe s'écrit **\'**

echo :pour afficher à l'écran

echo -e : activer le retour à la ligne (il faudra ensuite utiliser le **\n**)

echo $nomVariable : afficher la variable nomVariable

read nomVariable : demander de saisir une variable et la stocker dans nomVariable

read -p 'Entrer votre nom :**' -n** 5 nomVariable : écrire un prompt pour guider l'utilisateur (la variable à entrer est limitée à 5 caractères maximum)

-t (valeur) : limiter le temps de réponse à (valeur) secondes

-s : ne pas afficher le texte saisi par l'utilisateur

let ‘‘ opération **’’** : définir une variable mathématique ou effectuer une opération mathématique

Les variables des paramètres

Si on exécute un script avec paramètres (exple ./nomDuScript.sh param1 param2 param 3)

alors, les variables qui y sont reliées sont :

: le nombre de paramètres

0 : le nomDuScript

1 : le 1er paramètre param1

2 : le 2è paramètre param2 …jusqu'à 9

NB : la commande **shift** (en plein milieu du script) permet de décaler les paramètres dans les variables vers la gauche (ie que la variable 1 contient désormais pram2 et ainsi de suite.

Les tableaux

tableau**=(**'valeur0' 'valeur1' 'valeur2'**)** : pour créer un tableau

tableau**[**2**]=**'valeur2' : définir le contenu de la **3è** case du tableau

${tableau**[**2**]}** : pour afficher le contenue de la **3è** case du tableau (***** dans les [ ] pour tout afficher)