REPPRENEZ LE CONTROLE A L'AIDE DE LINUX

CHAPITRE 1 Differentes commandes

commande -d -a -U -h <=> **commande -daUh**
commande -p 14 <=> **commande –parametre=14**

Une unique Tabulation termine l'écriture d'une commande
Double Tabulation affiche liste des commandes débutant par ce qu'on écrit

history historique des dernières commandes
history >> texte AJOUTE dernières commandes dans text (créé si inexistant)
history > text.ok RAJOUTE dernières commandes dans text.ok (écrase si existant sinon créé)
export HISTSIZE=1000 modifie nombre de commandes enregistrées à 1000

Ctrl+Alt+T ouverture nouvelle fenêtre de Terminal
Ctrl+L <=> **clear**
Ctrl+D <=> **exit**

Sous Linux tout est un fichier; 2 types de fichiers sur Linux: - fichiers normaux (txt,avi)
- fichiers spéciaux (lecteur CD)
Ctrl (+Fn) +Alt+F1 pour accéder aux différentes consoles (de F1 à F6)
Ctrl (+Fn) +Alt+F7 pour revenir sur le mode graphique
Terminal = Console | tty1 = Terminal 1 | seveur X = l'interface graphique

Ctrl + Alt + Backspace redémarre le serveur X utile si plantage (rare)
Remplacé sur Ubuntu par **Alt + ImprEcran + K (+Fn)**

Telnet : protocole de communicat° non sécurisé (tt le monde peut ecouter)
SSH : protocole le + utilisé (peut crypter les données donc la co au serveur)

PuTTy : programme qui permet de se connecter en SSH (logiciel sur Windows)
Stocker l'empreinte permettra de vérifier si c'est le bon serveur et pas un spoof

@ signifie «chez» | **:** est un séparateur
~ indique le dossier dans lequel on est (synonyme de dossier personnel)
$ signifie que l'on est sur un compte «normal»
# signifie que l'on est en mode super-utilisateur

pwd : affiche le dossier actuel
which : connaître l'emplacement d'une commande
which pwd par exemple donne /bin/pwd où pwd n'est pas un dossier mais le prog lui-même

ls : liste les fichiers et dossiers (dossiers en bleu)
Les fichiers cachés commencent toujours pas un point (i.e. «.gaim»)
s'il n'y a pas de couleur => **ls --color=auto** (normalement on a cet alias sur ls)
ls -a affiche tous les fichiers et meme ceux cachés
ls -A affiche tous les fichiers et meme ceux cachés et naffiche pas . et ..
ls -F indique le type d'élément dans le dossier (@Examples est un raccourci)
ls -l (L min) indique le détail de chaque élément du dossier
ls -lh affiche taille des fichiers et dossiers en Ko Mo Go (**-h** = human readable)
ls -lt avec t trie par date de derniere modification
ls -ltr avec r qui inverse la liste (donc on aura le dernier fichier modifié en tête)
ls -l Example.txt permet d'obtenir uniquement les infos sur Example.txt
ls -l *.jpg affichera uniquement les infos des fichiers ayant pour extensions .jpg
ls -i afficher les inodes
ls -lArth (**ls -Arthi** i pour afficher les inodes)

cd : change working directory
cd / amène à la racine (**/** est la racine du disque dur (appelé C:\ sur Windows))
cd .. permet de revenir au dossier precedent
cd ~ ou **cd** permet de retourner dans le dossier courrant

du : taille occupée par les fichiers (Disk Usage)
du –h affiche la taille de chaque dossier + facilement lisible
du -a affiche la taille des dossiers ET des fichiers
du -s obtenir le grand total du dossier

cat et less : afficher le contenu des fichiers
/var/log dossier contenant != fichiers log qui garde 1 trace de l'activité du pc
Sous Linux il est courant d'avoir des fichiers sans extension

cat : permet d'afficher TOUT le fichier en 1 cp (dc + adaptée pr les ptits fichiers)
cat -n permet d'afficher le numéro des lignes du contenu

less : permet d'afficher le contenu du fichier page par page
less NomDuFichier affiche le fichier NomDuFichier

Raccourcis claviers dans less pour naviguer dans le fichier :

Espace	affiche une nouvelle page entière
Haut et **Bas**	naviguer
q	permet de quitter le fichiers
=	indique la ligne d'où l'on est dans le fichier
h	permet d'afficher les commandes possibles dans less
/ puis le texte	pour rechercher ce que l'on veut dans le fichier
n	permet de passer au suivant dans la recherche de texte
N	permet de rechercher le précédent

head and tail permet d'afficher le début et la fin du fichier
head -n 3 example affiche les 3 premières lignes du fichier example
tail -n 5 affiche les 5 dernières lignes du fichier example
tail -f (f=follow) permet de suivre en direct les changements
tail pratique for log'files qui add continuellement des lignes en fin du fichier
Ctrl+C pour quitter le tail et le head (équivalent de Alt+F4 sur Windows)
Par défaut tail met a jour les changements toutes les secondes
tail -f -s 3 syslog màj de l'affichage du fichier syslog all 3 sec

touch et mkdir créent des fichiers et des dossiers
touch : créé les fichiers
touch est à la base faite pour modifier la dernière date d'un fichier
si le fichier n'existe pas il sera créé
touch Fichierbidon créé le fichier FichierBidon dans le working directory
touch FichierBidon.txt on peut choisir lextens° qu on veut ou ne rien mettre
touch text1 text2.txt on peut créer plusieurs fichiers en meme temps
pour choisir un nom avec un espace il faut **touch «nom du fichier»**

mkdir : créé des dossiers
mkdir NouveauDossier NouveauDossier2 «Nouveau dossier 3»
mkdir -p Secret/porn/love/Bigtits creates les folders intermediaires

cp et mv copier et deplacer un fichier

cp : copier
cp fichierbidon fichiercopie copie de fichierbidon appelée fichiercopie
cp fichierbidon folder copie de fichierbidon dans foolder sous le mm nom
cp file folder/truc copie de file dans dossier folder sous le nom truc
cp fichierbidon /var/log
cp -R dossier1 dossier2 copie dossier1 sous le nom dossier2 (on peut aussi utiliser **-r**)
cp *.jpg mondossier/ copie tous les .jpg dans mondossier/
cp so* tree2/ copie tous les fichiers commencant par so dans tree2
cp -i fichierbidon fichcopie avertira si fichier fichcopie déjà existant

mv : move
mv permet de déplacer fichier et dossier mais aussi de les renommer
mv animaux.txt folder/ pdeplace annimaux.txt folder/
mv ficjier folder/truc deplace fichier dans folder et le renomme en truc
mv fichier ../truc2 deplace fichier dans dowwier precedent sous le nom truc2
mv fixhier ./truc deplace fichier ds dossier actuel ss nom truc (on renomme)

. signifie dossier actuel
.. signifie dossier précédent (cd ../../truc recule de 2 dossiers et va ds truc)

ln : créer des racourcis (= liens sous Linux)
Il y a les **liens physiques** et les **liens symboliques**. Sur le HDD chaque fichier est **séparé** par son nom, ses droits d'accès et son contenu aidant ainsi Linux à s'organiser. Chaque fichier est identifié par un **inode** (numéro d'identification) qui permet de pointer vers le contenu

* **le lien physique** => 2 noms de fichier != & 1 ! inode (i
on peut donc accéder au contenu de != façons sous des noms et emplacements !=
i.e. : **mkdir test** => **cd test** => **touch fichier1** =>**ln fichier1 fichier2**
impossible de créer des liens physiques pour des dossiers
ls -i permet d'afficher l'inode des fichiers
l'inode sera supprimé si tous les fichiers qui pointent dessus sont supprimés

* **le lien symbolique** => raccourcis comme sur Windows
ln -s file Tc pour créer un lien symbolique (fichier qui pointe un autre fichier)
Sur la première colonne de Tc on voit lrwxrwxrwx, le l signifie link et que Tc pointe vers file

rm : remove

rm Tuc1 Tuc2	supprime les deux fichiers Tuc1 et Tuc2
rm -i fichierbidon	supprime le fichier avec une vérification (utile pour script bash)
rm -f fichier	force la suppression du fichier
rm -v fichier	(-v = --verbose) demande a la machine les operations qu'il fait
rm -r animaux/	supprime le dossier animaux et son contenu
rm - rf *	permet de supprimer par force all files et dossiers du working diretory

NE JAMAIS FAIRE CETTE COMMANDE : rm - rf /*

CHAPITRE 2 Les utilisateurs et les droits

sur Ubuntu pas de session super-utlisateur
sudo permet d'exécuter une commande en **root** / **sudo su** pour rester en mode super-user
exit pour quitter le mode super-user

adduser / deluser / passwd / addgroup / delgroup (à effectuer en root)

adduser patrick	ajouter utilisateur patrick
passwd patrick	modifier le mot de passe de patrick
passwd	modifier le mot de passe du root
deluser patrick	supprimera le compte de patrick mais pas son home
deluser --remove-home patrick	supprime le home de patrick (**à effectuer en plus**)

NE JAMAIS SUPPRIMER TOUTES LES SESSIONS CAR PAS DE SESSION ROOT

* **adduser** et **deluser** uniquement sur Debian et descendants de Ubuntu, sinon ce sera **useradd** et **userdel** qui sont les commandes UNIX traditionnelles qui fonctionnent partout, elles sont plus basiques et il faudra ajouter **passwd** pour que la session soit activée
* root fera parti du groupe root
addgroup amis créé le groupe amis
delgroup amis supprime le groupe amis

usermod : modifier un utilisatueur

usermod -l truc patrick	renomme l'utilisateur patrick en truc
usermod -g amis patrick	mets l'utilisateur patrick dans le groupe amis
usermod -G amis,truc,patate Pat	mets l'utilisateur Pat dans les trois groupes **Pat quittera les autres anciens groupes**
usermod -aG truc Pat	add Pat au groupe truc en + de ses anciens groupes

*meme remarque que pour adduser et deluser , on aura groupadd et groupdel (- d'options)

chown : gestion du proprietaire d'un fichier (à effectuer en root)
chown Pat texte.txt fera de Pat le proprio de texte.txt
chown Pat:amis texte.txt associe le fichier texte.txt au proprio Pat et au groupe amis
chown -R nicolas:nicolas /home/patrick pour faire de nicolas le proprio du dossier patrick

chgrp : change le groupe proprio du fichier1
chgrp amis texte.txt

chmod : modifie les droits d'accès (ls -l pour voir les droits)
d (Directory) indique si l'élément est un dossier
l (Link) indique si l'élément est un lien (raccourci)
r (Read) indique si on peut le lire
w (Write) indique si on peut le modifier
x (eXecute) indique si on peut exécuter
* si lettre présente => droit ok | si tiret a la place de la lettre => pas de droit
* si un dossier est x cela indique qu l'on peut le traverser (que l'on peut voir les sous-dossiers)
* autre = autres comptes
* avec un droit 000 SEUL le root peut faire ce qu'il veut sur le fichier

d	rwx	rwx	rwx
dossier	utilisateur	groupe	autre

droit	r	w	x
chiffre	4	2	1

---	r--	-w-	--x	rw-	-wx	r-x	rwx
0	4	2	1	6	3	5	7

chmod 666 rapport.txt
chmod g+w rapport.txt - enleve + ajoute un droit (u=user,g=groupe,o=others)
chmod u+rw rapport.txt
chmod g+w,o-w rapport
chmod +x text donne le droit d'exécution a tout le monde
chmod u=rwx,g=r,o=- rapport
chmod -R 700 /home/nicolas pour donner les droits sur le dossier nicolas

Vim / Nano / Emacs : Editeurs de texte (**nano** le plus simple à utiliser)
nano pour le démarrer
nano text pour ouvrir texte sur nano sil nexiste pas il sera créé
nano -m autorise lutilisation de la souris
nano -i respectera l'indentation (utile pour scripts)
nano -A active la retour intelligent au début de la ligne à l'indentation s'il y en a une

Raccourcis claviers dans nano pour naviguer dans le fichier :
^ signifie Ctrl
Ctrl+O pour enregistrer
Ctrl +C pour annuler la commande quitter
Ctrl +G pour afficher l'aide

* appelé **nano** car son script est minuscule
* **nano** est un éditeur de texte mais (pas un traitement de texte)
* éditeur de texte = permet de modifier le contenu brut d'un fichier (comme bloc notes)
* traitement de texte = permet de faire des mises en forme (comme Word => italique gras,etc)

.nanorc (fichier de **configuration des nano**)
* pour les configurations personnelles modifier .nanorc dans **home/nicolas/**)
* pour les configurations globales pour toutes les sessions modifier **/etc/nanorc**
* chaque commande dans un nano débute par **set** (activer) ou **unset** (désactiver)
set mouse permet d'activer la souris (comme **nano -m**)
set autoindent permet d'activer l'indentation (comme **nano -i**)
set smarthome active le retour intelligent en début de ligne (comme **nano -A**)

.bashrc (fichier de **configuration des bash**)
.bashrc permet de **personnaliser l'invite de commande**

les alias : commandes programmables
par exemple **ls** a pour alias **ls --color==auto** dans le .bashrc
Taper alias dans le Terminal affichera la liste des alias actifs
alias nom='commande' pour configurer l'alias directement en commande
alias rm='rm -- preserve - root' to protect root si ordre deffacer root
* le bash commun pour toutes les sessions est dans **/etc/bash.bashrc**
* le bash personnel est dans home/nicolas/
* SI User logged => bash commun + perso actifs / SI User NON Logué => ! bash commun actif

CHAPITRE 3 Installer les programmes avec apt-get

* pas de programme d'installation sous Linux, on appelle ça des **paquets**
* un **paquet** est un sorte de .zip cest **.deb** comme DEBian (prog prêt à l'empoi)
* les **dépendances** sont les autres prog nécessaires pour faire fonctionner le prog
* lors de l'installation le programme montrera la liste des dépendances
* l'endroit (serveur) où tous les paquets se trouvent est appelé **dépôt** (**repository**)
* lors de nouvelles **màj** le **serveur par défaut** est **surchargé** => vaut mieux en sélectionner un autre par défaut (fichier contenant la liste des dépôts **/etc/apt/sources.list**)
dans ce fichier les commandes commencent soit par **deb** soit par **deb-src**
Par exemple **deb http://fr.archive.ubuntu.com/ubuntu/ hardy universe**
* on utilise la version de distribution hardy
* universe correspond a la section du dépôt
* **pour changer les dépôts par défaut il suffit de changer l' adresse http**

deb sert à **télécharger la version compilée binaire** (**version prête a l'emploi**)
deb-src permet de **récupérer le code source du programme** (inutile si qu'installer)

changer les dépôts graphiquement
Système=>logiciels&màj=>dl depuis=>autre=> select best serveur
possible d'avoir besoin d'une màj pour compatibilité avec les paquets

apt-get
apt-get update pour màj de notre cache
apt-cache search monpaquet search le paquet voulu à dl si on connaît pas le nom exact
* apt-cache search recherche les paquets dans le cache (pas besoin d'internet)
apt-get install monpaquet pour dl et installer le paquet (avec le nom exact!)
apt-get update dl le nvo cache (liste) des paquets et leur vers° proposé sur le dépôt
* màj le cache si nouveau dépôt par défaut
apt-get upgrade màj des paquets installés à partir d'une comparaison du cache
* donc **apt-get updat**e puis **apt-get upgrade**
apt-get autoremove supprimer les paquets inutiles
apt-get remove lepaquet désinstalle lepaquet mais pas ses dependances
apt-get autoremove lepaquet desinstalle lepaquet et ses dépendances devenues inutiles

man commande affiche le manuel de la commande (i.e. **man mkdir**)
lience GPL permet d'avoir le **droit de lire la source** et de **le redistribuer librement**
commande -h ou **commande --help**
apt-get install manpages -fr avoir les manuels en francais (deconseillé svt pas maj)
apt-get autoremove manpages -fr uninstal manuels traduits en FR

Dans le manuel :
- SYNOPSIS (partie la + importante) montre les != manières de use la cmde
- les … après DIRECTORY =l'on peut en mettre +sieurs (i.e. +sieurs directories)
mots en crochets <=> facultatif
mots en gras <=> mot à taper tel quel
mots soulignés <=> mots à modifier par leur but
barres vertiales | <=> OU

apropos : trouver une commande (i.e. **apropos copy** donne liste des cmds en lien ac copy

whatis commande : savoir ce qu'est la commande recherchée (i.e. whatis mkdir)

locate : localiser les fichiers (i.e. locate truc localisera ttes files ayant truc ds leur noms**)**
défaut de locate : recherche done on a **DataBase of our files** (liste des files & de leur position) : la màj s'effectue tous les jours => pour forcer la màj **sudo updatedb**
* **slocate est nouveau** (=> apt-get install slocate) il n'affichera pas les fichiers dont on n'a pas les droits de lecture alors qu'avec **locate** SI

find effectue des recherches approfondies (fichiers et dossiers actuellement present)
find ne va pas lire dans une DB, elle va **parcourir tout le disque dur** (peu être long)
find «ou» «quoi» «que faire»

ou ? pour déterminer le dossier ds lequel on va chercher (si rien de noté elle se fera dans WD)
quoi ? Type de recherche : par nom, date de création, taille etc.
que faire ? Post traitement (i.e. afficher la liste des recherches [par défaut si rien de noté])

Recherche par Nom
find -name 'fichier.txt' cherche le nom exacte dans ~ (**-name** = nom exact)
find /var/log/ -name 'photo.png' (photo2.png ne sera pas affiché car ! nom exact)
find /var/log/ -name 'photo*' trouve tout fichier commençant par photo
find /var/log/ -name '*photo*' recherche les fichiers contenant photo dans leur nom
find / -name 'photo' recherche partout sur le disque dur
Recherche par Taille (**k,M** et **G** pour **Ko Mo** et **Go**)
find ~ -size +10M recherche les fichiers dans ~ **ayant + de 10Mo**
find ~ -size -10M **find ~ -size 10M**
Rechercher par date de dernier accès
find -name «*.odt» -atime -7 recherche fichier ayant pour ext .odt datant De - de 7 jours
find -name «*.odt» -atime +7 **find -name «*.odt» -atime +7**
Recherche ! répertoires ou fichiers
find -type d recherche uniquement dossiers
find -type f recherche uniquement fichiers

Utilisation avancée avec des resultats
find -name '*.jpg' <=> **find -name '*.jpg' -print** liste ! of files founded (automatique)
Afficher les fichiers de facon formatée
find -name «*.jpg» -printf « %p -%u \n» liste results en affichant nom of file (**%p**) et proprio (**%u**)
man find pour plus de détail sur -printf
Supprimer le fichiers trouvés
find -name «*jpg» -delete supprimera tout .jpg se trouvant dans ~
Appeler une commande
find ~ -name '*.jpg' -exec chmod 600 {} \; pour chmod 600 sur les fichiers trouvés dans ~
* **les accolades remplaceront le nom de chaque fichier lors de l'exécut° de la cmde)**
* **la commande doit finir par un \; obligatoirement**
* **pour demander confirmation on utilise -ok a la place de -exec**

grep : filtrer les données (recherche 1 mot ds 1 fichier & to show lines of mots trouvés)
grep texte nomdufichier (i.e. **grep alias .bash.rc)**
grep «truc de la vie qui tue» fichier guillemets pour rechercher la combinaison de mots
grep -i alias .bashrc permet de ne pas tenir compte de la casse (Alias,ALIAS,alias,etc.)
grep -n alias .bashrc permet d'afficher les numéros de ligne
grep -v alias .bashrc inverser la recherche affiche ttes les lignes qui na pas le mot
grep -r alias .bashrc rechercher dans tous les dossiers et sous dossiers (**-r** = recursive)
grep -r 'Site Du Zéro' code/ cherche la chaîne ds ts files du dossier code y & ds ses ss dossiers

Utiliser grep avec des **expressions regulieres**

.	Caractère quelconque
^	Début de ligne
$	Fin de ligne
[]	Un des caractères entre les crochets
?	L'élément précédent est optionnel (peut être présent 0 ou 1 fois)
*	L'élément précédent peut être présent 0, 1 ou plusieurs fois
+	L'élément précédent doit être présent 1 ou plusieurs fois
|	Ou
()	Groupement d'expressions

grep -E avertie q'on utilise des **expressions regulieres**
grep -E ^Alias .bashrc recherche Alias en début de chaque ligne
grep -E [Aa]lias .bashrc renvoie toutes les lignes contenant Alias ou alias
grep -E alias$ .bashrc recherche le mot en chaque fin de ligne
grep -E [0 -4] .bashrc revoie toutes les lignes contenant 0 1 2 3 ou 4
grep -I (i majuscule) permet d'omettre les fichiers binaires
-E conservé pr raisons de compatibilité avec les autres distrib Unix (pas besoin de l'add nomralmt)

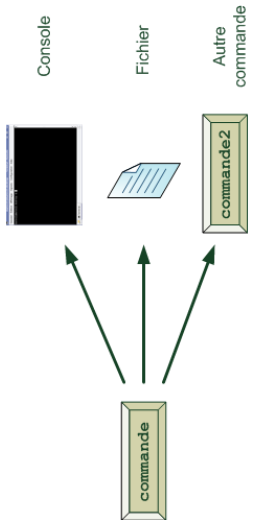
sort : trier des lignes (ordre alphabétique , ordre numérique, ordre aléatoire)
sort noms.txt trie alphabétiquement les noms (automatique si aucun param addded)
sort -o noms_tries.txt noms.txt permet décrire le résultat dans noms-tries.txt
(pour l'ex créer un fichier noms.txt contenant les noms sur chaque ligne
François Marcel Albert Jean Stéphane Patric Vincent jonathan)
sort -r noms.txt permet de trier en ordre inverse
sort -R noms.txt trie aléatoirement (utile ds certains cas)
sort -n noms.txt permet de trier des nombres

wc : compter le nombre de lignes
wc noms.txt affichera **le nombre de lignes, de mots et d'octets**
wc -l noms.txt compte uniquement le nombre de lignes
wc -w noms.txt le nombre de mots
wc -c noms.txt le nombre d octets
wc -m noms.txt le nombre de caractères
* le nbr de caractères et d'octets sont != car caractères spéciaux sur plusieurs octets (svt 2o)

uniq : supprimer les doublons dans un fichier
uniq ne fonctionnera que si la liste est déjà trié dans l'ordre alphabétique
uniq doublons.txt (valable aussi pour les triplets , etc.)
uniq doublons.txt sans_doublons.txt créé un fichier sans doublons
uniq -c permet de compter le nombre d'occurrences
uniq -d permet d'afficher uniquement les lignes présents en doubles

cut : couper une partie du fichier
soucis avec les caractères spéciaux comme è qui prennent plus d'un octet
cut -c 2-5 noms.txt conserve uniquement du caractère 2 au 5 (2et 5 inc**lus)**
cut -c 2- noms.txt conserve les caractères de chaque ligne à partir du 2ieme
Couper selon un délimiteur
* **les fichiers CSV (comma separated values) st des valeurs séparées par des virgules**
* **Excel use ; comme séparateur pour faciliter l'échange et le traitement des données**
Travaillons sur un fichier notes.csv contenant sur chaque ligne:

Fabrice ,18 / 20 , Excellent travail | Sophie ,14 / 20 , En nette progression
Mathieu ,3 / 20 , Nul comme d ' hab | Mélanie ,9 / 20 , Allez presque la moyenne !
Albert ,20 / 20 , Toujours parfait | Corentin ,11 / 20 , Pas mal mais peut mieux faire
Benoît ,5 / 20 , En grave chute



cut -d , -f 1 notes.csv extirpe les noms de la liste on a
où| **-d ,** indique que la virgule est le séparateur
| **-f 1** indique les numéros du ou des champs a séparer (ici colonne 1)
cut -d , -f 1 notes.csv extirpe la première colonne du fichier
cut -d , -f 1,3 notes.csv extirpe les noms et les appréciations de chaque ligne
cut -d , -f 2-4 notes.csv extirpe les colonnes 2 3 et 4 (inclus)
cut -d , -f 3- notes.csv extirpe les colonnes a partir du 3 inclus

CHAPITRE 4 Les Flux De Redirection

> et >> : rediriger le résultats dans un fichier (cf doc 1 a droite)

> : rediriger dans un nouveau fichier
si le fichier existait déjà il sera écrasé
cut -d , -f 1 notes . csv > eleves.txt inscris le résultat dans eleves.txt
commande_bavarde > /dev/null **astuce pour ni afficher ds la cmd ni de le record**
/dev/null est le «**trou noir**» de Linux tt ce qui va dedans y disparaît

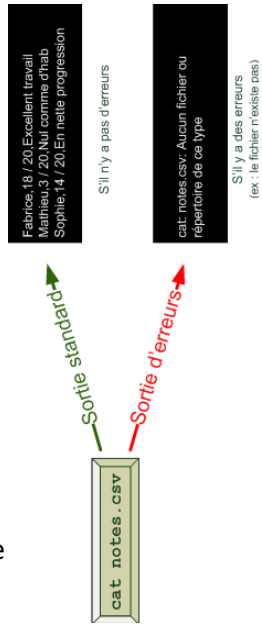
>> : rediriger à la fin d'un fichier
macommande >> resultats.log astuce pour vérifier les cmdes tapées

2>, 2» et 2>&1 : rediriger les erreurs (cf doc 2 a droite)
il existe deux type de flux de données :
– la **sortie standard** : pour tous les messages (sauf les erreurs)
– la **sortie d'erreurs** : pour toutes les erreurs

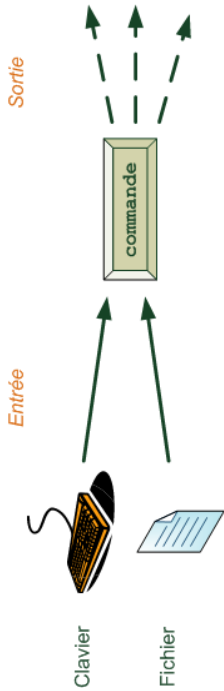
Par exemple :
cut -d , -f 1 fichier_inexistant.csv > eleves.txt
cut : fichier_inexistant.csv : Aucun fichier ou répertoire de ce type

cut -d , -f 1 fichier_inexistant.csv > eleves.txt 2> error redirige erreur ds error
On peut aussi utiliser **2>>** celui ci inscrira l'info en fin de fichier s'il existe déjà

2>&1 : Fusionner les sorties
cut -d , -f 1 fichier_inexistant.csv > eleves.txt 2>&1 tout ira aussi ds eleves.txt
cut -d , -f 1 fichier_inexistant.csv >> eleves.txt 2>&1 pr ajouter si file existante



< et << : lire depuis un fichier ou d'un clavier (cf doc 1 a droite)
décider d'où vient l'entrée dune commande (clavier ou a partir d'un fichier)



< : lire depuis un fichier (**chevron ouvrant indique d'où vient la cmde**)
cat < notes.csv ou cat notes.csv idem graph ms ce qui se passe behind is!=

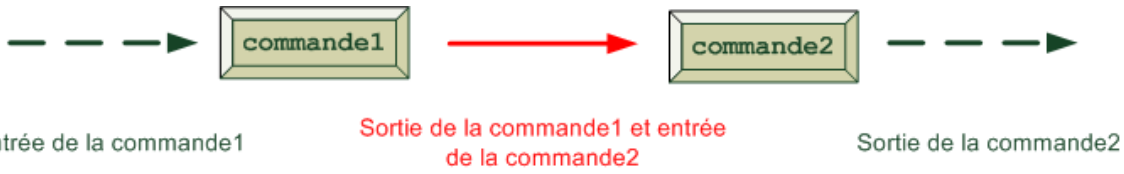
cat notes.csv cmde cat reçoit en entry le nom du fichier notes.csv qui open pr show it
cat < notes.csv cmde cat reçoit le contenu de notes.csv qu'elle affiche ds la console.
C'est le shell qui se charge d'envoyer le contenu de notes.csv à la commande cat.
2 façons de faire la mm chose mais de manières != ce sera utile plus loin

<< : lire depuis le clavier progressivement

permet d'entrer directemt le contenu sur la cmde jusqu'à 1 certaine indication
sort -n << FIN trie une fois que FIN est entré
permet d'éviter de créer un ficher si l'on en a pas besoin
wc -m << FIN calcule le nombre de caractères tapés
* on peut remplacer FIN par un tout autre mot

< : envoie le contenu d'un fichier à une commande ;
<< : passe la console en mode saisie au clavier, ligne par ligne.
Toutes ces lignes sont envoyées à la cmde lorsque le mot-clé de fin aura été écrit.
* possible de combiner les symboles i.e. **sort -n << FIN > nombres_tries.txt 2 >&1**

| : chaîner les commandes
| (= pipe) permet de **Chaîner des commandes**
Chaîner des commandes = connecter la sortie d'une cmd à l'entrée d'une autre cmd



Tout ce qui sort de la commande1 est immédiatement envoyé à la commande2


```
Fabrice,18 / 20,Excellent travail        Fabrice          Albert
Mathieu,3 / 20,Nul comme d'hab           Mathieu          Benoît
Sophie,14 / 20,En nette progression      Sophie           Corentin
Mélanie,9 / 20,Allez presque...          Mélanie          Fabrice
Corentin,11 / 20,Pas mal mais...         Corentin         Mathieu
Albert,20 / 20,Toujours parfait          Albert           Mélanie
Benoît,5 / 20,En grave chute             Benoît           Sophie
```

Exemple : Trier les élèves par nom **cut -d , -f 1 notes . csv | sort > noms_tries.txt**

Trier les répertoires par taille :
du | sort – nr
du | sort - nr | head permet d'afficher les fichiers les plus lourds si bcp de répertoires
du | sort - nr | less pour naviguer a travers la liste

Exercice : peut-être avez-vous toujours trop de répertoires sous les yeux et que vous vous intéressez seulement à certains d'entre eux. Pourquoi ne pas filtrer les résultats avec grep, pour afficher uniquement la taille des répertoires liés à. . . Firefox par exemple ? ⇒ **du -ah | grep firefox | sort | cut -d . -f 1 | tail -n 15**

Lister les fichiers contenant un mot

⇒ **sudo grep log -Ir /var/log | cut -d : -f 1 | sort | uniq**

CHAPITRE 5 Surveiller l'activité du système

w : qui fait quoi ?
Heure: (= **date**)
Uptime : (=**uptime**) durée depuis démarrage
Charge : (**uptime** & **tload**) indice d'activité du processeur 1)1min 2)5min 3)15min
Liste des connectés : (**who**)
TTY : nom de la console ou est le user (tty1 à 6=console et tty7=graphi, pts= console graphi)
FROM : addresse IP depuis on se co (0.0 c'est qu'on est physiquement sur la machine)
Login@ : heure à laquelle on s'est connecté
Idle: depuis combien de tps l'user est inactif (qu'il na pas fait de commande)
What : La commande qu'il est en train d'utiliser en ce moment (bash => invite de commande ouverte et aucune commande particulière n'est exécutée

* Charge = nbr moyen de processus en train de tourner réclamant l'utilisation du processeur
* Charge maximal en 'pourcentage' : 1 pour 1 coeur , 2 pour dualcore , 4 pour quadcore
* Linux nécessite rarement de reboot (parfait pour serveur Unix) et que quand maj noyau NUIX
* Si la charge est très élevée pendant un certain temps alors il y a un problème
* tload => graphique de la charge selon le temps
* Idle = inactif , inoccupé

ps et top : Lister les processus (équivalents de Ctrl+Alt+Suppr. sur Windows)
* un processus est un programme qui tourne en mémoire
* la plupart des prog ne font tourner qu'un ps en mémoire ms il est possible qu'il y en ai +sieur

ps : liste les des ps qui tournent au moment où l'on effectue la commande
PID : numéro d'identification du processus
TTy : console depuis laquelle à été lancé le ps
TIME : durée d'exécution du ps depuis son lancement
CMD : commande qui a généré le ps
* Bash correspond au Terminal
ps affiche seulement les processus lancés par le même utilisateur
* beaucoup sont lancés par root
ps -ef liste tous les processus lancés par tous les utilisateurs
colonne **UID** (=UserID) indique le nom d'utilisateur qui a lancé le ps
ps -ejH affiche les ps en arbre (+ieurs ps st des «enfants» d'autres ps
ps -u UTILISATEUR liste les ps lancés par les utilisateurs (filtre par user)
* défaut de ps : la liste est statique

top : liste dynamique des ps (liste interactive et régulièrement màj)
* **top** ne peut pas tout afficher : il affiche les **plus gourmands par defaut (%CPU)**
h affichera l'aide de top
B met en gras certains éléments.
f ajoute ou supprime des colonnes dans la liste.
F change la colonne selon laquelle les processus sont triés. En général, laisser le tri par défaut en fonction de %CPU est suffisant.
u filtre en fonction de l'utilisateur que vous voulez.
k tue un processus, c'est-à-dire arrête ce processus.
s change le temps de màj de la liste

Ctrl+C et kill : arrêter un ps

Ctrl +C : demande «gentillement» au prog de s'arreter

kill : tuer un ps
kill PID où le PID est a trouver dans ps
ps -u mateo21 | grep firefox pour trouver par exemple firefox
kill 34051 1235 4651 par exemple pour tuer plusieurs ps
SI LE PROGRAMME PLANTE COMPLTEMENT kill -9 PID TUERA LE PS SANS POLITESSSE

killall : tuer plusieurs ps
killall NomDuPs tuera les process

halt & reboot : arrêter et redémarrer l'ordinateur (a effectuer en root)
halt : arrêter l'ordinateur
reboot : redémarrer l'ordinateur

*les commandes halt et reboot st en faite des cmds a paramètres spécifiques de **shutdown**

CHAPITRE 6 Executer un prog en arrière-plan

"&" & nohup : lancer un processus en arrière-plan
& : lancer un processus en arrière-plan (& = et «commercial»)
cp video.avi copie_video.avi & pour faire une copie en arrière-plan ce qui donnera **[1] 16504**
où **[1] 16504** est le **PID du ps cp video.avi copie_video.avi**
le ps sera en fond de tâche et on pourra continuer a taper des commandes
find / - name "* log " > sortiefind & donnera **[1] 18191 et** les results seront dans sortiefind
find / - name "* log " > sortiefind 2 >&1 & pour y rediriger aussi les errreurs
Soucis avec cette cmde ! Si on ferme le terminal la tache s'arrêtera !

nohup : détacher le processus de la console
nohup commande pour détacher la tache du Terminal
nohup cp video.avi copie_video.avi
nohup : ajout à la sortie de 'nohup.out '
la sortie de la commande est redirigée vers le fichier nohup.out
si l'on ferme la fenêtre ca continuera , il faudra utiliser kill pour l'arrêter
commande utile pour les serveurs empêchant de rester connecté pr que ça ne s'arrête pas
voir ce qui est inscrit dans le fichier nohup.out en faisant un test

Ctrl+Z, jobs, bg & fg : passer un processus en arrière-plan
si oublie de passer le ps en a-p et que le ps mets plus de temps que prévu on a +sieurs solut°

Ctrl+Z : mettre en pause l'exécution du programme
top => **Ctrl+Z** => **[1]+ Stopped** **top** => mets le ps en pause et reste en mémoire
bg : passer le processus en arrière-plan (background)
maintenant que le ps est en pause on tape **bg** => **[1]+ top &** => reprise du ps en a-p
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.
Tester pour voir si cest dependant du temrinal
jobs : connaître les processus qui tournent en arrière-plan
fg : reprendre un processus au premier plan (foreground)
fg %2 pour choisir le ps [2]

screen : plusieurs consoles en une (en root) **multiplicateur de Terminal**
sudo apt-get install screen **exit** ou **Ctrl+D** pour en sortir

Racourcis clavier pour screen :
Ctrl+A puis **?** : pour afficher laide
Ctrl+A puis **C** : créer une nouvelle « fenêtre ».
Ctrl+A puis **A** : renomme la fenêtre actuelle
Ctrl+A puis **W** : la liste des fenêtres
Ctrl+A puis **N** : passer à la fenêtre suivante (next).
Ctrl+A puis **P** : passer à la fenêtre précédente (previous).
Ctrl+A puis **Ctrl+A** : revenir à la dernière fenêtre utilisée.
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 deux parties puis 3 puis 4 , etc.
Ctrl+A puis **Tab** : pour passer dans une fenetre plus bas
Ctrl+A puis **X** : pour fermer la fenetre utilisée
Ctrl+A puis **D** : détacher screen (revenir a l'invite de commande normale)
* On sera dans la fenetre contenant l'etoile * a coté du nom de la fenetre
Les screens continueront à tourner en fond de tâche même si l'on ferme le terminal
screen -r pour retrouver les screens detached
screen -r 20930 pour reouvrir la session screen souhaité
screen –ls affiche la liste des sreens actuellement ouverts
Verifier la commande Ctrl+A puis X avec un ecran splitté pour savoir son utilité

.screenrc /home/nicolas fichier de configuration des screens

date : régler l'heure
date «+%Hh:%Mm%Ss» affichera 12h:15m:16s
date «+%H» affichera 12
date «+Bienvenue %Y» affichera Bienvenue 2016
sudo date MMDDhhmmYYYY pour changer le mois jours heure min année

at : exécuter une commande plus tard (fonctionnera en fond de tâche)
at 14:17
warning : commands will be executed using /bin/sh
at > touch fichier.txt
at > <EOT >
job 5 at Mon Nov 10 14:17:00 2010

Ctrl+D pour terminer la programmation des commandes a effectuer
at 14:17 tomorrow pour effectuer la commande le lendemain
at 14:17 11/15/10 15 novembre 2010
at now +5 minutes pour executer une cmde après un certain délai

atq et atrm : lister et supprimer les jobs en attente
atq permet d'obtenir la liste des jobs en attente
atrm 13 pour supprimer la tache planifiée 13

sleep : faire une pause
touch fichier.txt ; rm fichier.txt pour effectuer plusieurs cmd avec le ;
touch fichier.txt ; sleep 10 ; rm fichier.txt pour temporiser 10 sec entre les cmd
sleep 2m / **sleep 3d** / **sleep 4** / **sleep 3h** 2min/3days/4sec/3heures

touch fichier.txt && sleep 10 && rm fichier.txt où && séparateur de cmde
si erreur en touch fichier.txt , ca sarrete et sleep et rm ne seront pas exec

crontab : exécuter une commande régulièrement
* **vi** est l'éditeur par défaut (plus compliqué que **nano**)
pour mettre nano en éditeur par défaut il faut écrire
export EDITOR=nano ou **echo "export EDITOR=nano" >> ~/.bashrc**
crontab est une cmd qui permet de lire et modifier un fichier appelé «crontab»
ce fichier contient la liste des prog qu'on souhaite exec régulièrement et a quelle hour les exec

crontab et cron : modifier la liste des programmes à exec, exec les programmes
les parametres de crontab :
crontab –e modifier la crontab
crontab –l afficher la crontab actuelle ;
crontab –r supprimer votre crontab (la suppression est immédiate et sans confirmation !)
modifier la crontab
après **crontab -e** on peut voir **m h dom mon dow command**
[min(0-59) hour(0-23) dayofmonth(1-31) month(1 -12) dayofweek(0-6) command]
* 0 = dimanche
exemple **47 15 * * * touch/home/mateo21/fichier.txt** (tous les jours a 15h47)
* toujours préférable décrire le chemin entier (on ne sait pas ou il lancera la manip)
* préférable d'utiliser **contrab -e** plutôt qu'utiliser le **nano** pke il y aura verificat° de syntaxe
(une fois la modification terminée on peut voir que contrab installe le nouveau fichier modifié)

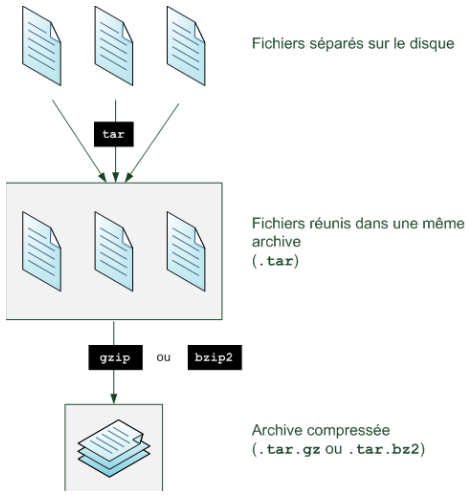
Crontab	Signification
47 * * * * commande	Toutes les heures à 47 minutes
0 0 * * 1 commande	Tous les lundis a minuit
0 4 1 * * commande	Tous les 1er du mois à 4h du matin
0 4 * 12 * commande	Tous les jours de decembre à 4h du mat
0 * 4 12 * commande	Toutes les heures le 4 decembre
* * * * * commande	Toutes les minutes

Les différentes notations possibles remplaçant l'étoile :
un nombre \ * \ 3,5,7 \ 3-7 \ */3 tous les multiples de 3

Crontab	Signification
30 5 1-15 * * commande	A 5h30 du 1au15 chaque mois
0 0 * * 1,3,4 commande	A minuit tous les lundis,mercredis,jeudis
0 */2 * * * commande	Toutes les 2 heures
*/10 * * * 1-5 commande	Toutes les 10 minutes du lundi au vendredi

Rediriger la sortie
47 15 * * * touch /home/nicolas/fichier.txt >> /home/mateo21/cron.log 2>&1
47 15 * * * touch /home/nicolas/fichier.txt > /dev/null 2>&1

CHAPITRE 6 Archiver Et Compresser



tar : assembler des fichiers dans une archive
gzip et **bzip2** ne peuvent **compresser qu'un seul fichier à la fois**
* sous Linux 2 étapes de compression:
1-/ réunir les files ds un seul gros fichier = **archive** (ac **tar**)
2-/ **compresser** le gros fichier obtenu (ac **gzip** ou **bzip2**)
* **zip** et **rar** permettent de compresser +sieurs fichiers à la fois
* **jpeg , png** et **gif** sont des fichiers déjà compressés

mettre les files in a same folder puis créer une archive
tar -cvf nom_archive.tar nom_dossier/ créer une archive tar
c signifie créer l'archive
v afficher le détail des opérations
f assembler l'archive en un fichier
x pour extract
t pour lister le contenu de l'archive
r pour append a file at the end of an archive
* on peut très bien ne pas créer de dossier **tar -cvf archive.tar fichier1 fichier2 fichier3** mais la coutume is de tt mettre in a folder pour que lors de l'extraction ce ne soit pas le bordel
tar -tf tutoriels.tar afficher le contenu de l'archive sans l'extraire
tar -rvf tutoriels.tar ajouter un fichier dans l'archive
tar -xvf extraire les fichiers de l'archive

gzip & bzip2: compresser une archive (**gzip** le + connu, **bzip2** compresse mieux ms + lgt)
.tar.gz si archive compressée par **gzip** ; **.tar.bz2** si archive compressée avec **bzip2**
gzip : la compression la plus courante
gzip tutoriels.tar pour compresser l'archive
gunzip tutoriels.tar.gz pour décompresser l'archive
bzip2 : la compression la plus puissante
bzip2 tutoriels.tar pour compresser l'archive
bunzip2 tutoriels.tar.bz2 pour décompresser l'archive
Archiver et compresser en même temps avec tar
tar -zcvf tutos.tar.gz tutos/ pour créer une archive tar qui sera compressée par gzip
z pour compresser avec **gzip**
tar -zxvf tutos.tar.gz pour décompresser l'archive avec **gzip**
tar -jcvf tutos.tar.bz2 tutos/ pour compresser l'archive avec **bzip2**
tar -jxvf tutos.tar.bz2 tutos/ pour decompresser l'archive avec **bzip2**
avec **tar -ztf** on peut regarder a l'intérieur de l'archive compressée en **.targz**
avec **tar -jtf** on peut regardera l'intérieur de l'archive compressée en **.tar.bzip2**

zcat, zmore & zless : afficher directement un seul fichier compressé (! les gzip)
Des fois on ne compresse qu'un seul fichier
zcat ,zmore, zless : équivalents de cat,more et less, capables de lire un fichier compressé **(gzippé)**

unzip & unrar : décompresser les .zip et .rar
on ne peut pas décompresser les .zip et les .rar avec gunzip

unzip : décompresser un .zip

sudo apt-get install unzip
unzip archive.zip
unzip -l tutoriels.zip permet de lire le contenue d'une archive zip
zip -r tutoriels.zip tutoriels/ pour compresser l'archive au format .zip

* sans le **-r** seul le dossier,vide, sera compressé

unrar : décompresser un .rar

sudo apt-get install unrar
unrar e tutoriels.rar
unrar l tutoriels.rar

on ne peut pas créer des archives compreessées en .rar car le logiciel n'est pas libre

CHAPITRE 7 La Connexion Securisée à Distance Avec SSH

* Le PC qui se connecte au serveur est appelé **client**
* Pour communiquer entre eux en réseau, deux ordinateurs doivent utiliser le même **protocole**
* Il existe +sieurs types de protocoles:**HTTP**(HyperText Transf. Protoc.), **FTP**(File Transf. Protoc) **IMAP** (Internet Message Access Protoc.),**Telnet** simple mais dangereux car **non sécurisé**

Le protocole SSH : la solution pour sécuriser les données
il existe!= méthodes de cryptage: le **cryptage symétrique** et le **cryptage asymétrique**

le cryptage symétrique (la méthode de cryptage la plus simple, assez robuste)
on utilise **une clé pour crypter le message**,**la même clé pour le décrypter**
soucis : il faut que le client et le serveur se transmettent discrètement la clé

le cryptage asymétrique
le **cryptage symétrique** utilise **une seule clé**
le **cryptage asymétrique** en a 2: une **clé publique pr crypter** et 1 **privée pr décrypter**
SSH combine cryptage **asymétrique** et **symétrique**
1-/ Use the cryptage asymétrique pour échanger discrètement la clé de cryptage symétrique
2-/ Use ensuite le cryptage symétrique pour communiquer
* Le cryptage asymétrique est 100 à 1000 X + lent que le symétrique c'est pourquoi on use le cryptage symétrique pour communiquer après l'échange de clé par cryptage asymétrique

les étapes de l'échange de la clef de cryptage symétrique: (procédure automatique)
1-/ le serveur créé 1 paire de clé à cryptage asymétrique, envoie la public et garde la privée
2-/ le client créé 1 clé à cryptage symétrique,la crypte ac la clé pub asym.et l'envoie au serveur
3-/ le serveur décrypte la clé symétrique ac sa clé privée : mtn ils communiquent en cypt. sym.
* C'est comme ça que **SSH** fonctionne pour **créer un canal d'échange sécurisé**

Transformer une machine en serveur (**sudo apt-get install openssh-server**)
on voit pendant l'installation:Creating SSH2 RSA key;this may take some time...#creation de clés asym.
Creating SSH2 DSA key;this may take some time...#dalgo RSA et DSA (RSA mieux)
* Restarting OpenBSD Secure Shell server sshd #prog seveur sshd lancé
sudo /etc/init.d/ssh start pour lancer le serveur (normaly serveur lancé à chaque boot)
sudo /etc/init.d/ssh stop pour l'arrêter
/etc/ssh/ssh_config fichier de config du serveur (pas besoin normaly)
si on modifie le fichier de config il faudra recharger SSH **sudo /etc/init.d/ssh reload**

Se connecter via SSH à partir d'une machine Linux
ssh login@ip pour se co (changer login par nicolas et ip par ladresse ip du pc)
ssh login@ip -p 12345 pour se co sur le **port 12345**
logout ou **Ctrl+D** pour se déconnecter du serveur
* **ifconfig** pour connaître son ip locale **; port 22 = port par défaut de connection SSH**
* **i.e. ssh mateo21@87.112.13.165 ; ssh mateo21@localhost pr se co en local**
* le **fingerprint** (empreinte) du **serveur permet d'identifier le serveur**.Si qqun essaye de se faire passer pour le serveur le **fingerprint** changera. SSH préviendra si c'est le cas
* il se peut que le fingerprint change si le serveur a été réinstallé ou possible attaque de man-in-middle
* important de se souvenir de how use **nohup** et **screen** pr exec des cmde en a-p
VERIFIER SUR PC MAISON POUR SE CONNECTER A DISTANCE (peut etre pb pare-feu)

Se connecter via SSH à partir d'une machine Windows
PuTTY : logiciel pour de se connecter via SHH à partir d'un Windows
* **putty.exe** pas besoin d'installat° ; **putty-0.60-installer.exe** si (conseillé pour + d'options)
* on aura aussi un message d'erreur si le fingerprint a changé ave ce logiciel

L'identification automatique par clé
2 façons de s'authentifier sur le serveur:
1-/ l'authentification par mdp
2-/ l'authentification par clé publique et privée du **client**
* il est possible d'éviter quon demande à chaque x le mdp ac 1 authentificat° spéciale par clé

L'identification automatique par clé
Avec cette nouvelle méthode **c'est le client qui va générer les clé publique et privée**

Authentification par clé depuis Linux
on effectue d'abord des opérations sur la machine du client puis envoyer les résults au serveur
1-/ Opérations sur la machine du client
ssh-keygen -t rsa (ou **dsa**) pour générer une parie de clés publique et privée sur le client
On demande un passphrase pour crypter la clé privée:
- soit on tape directement 'Entrée' et la clé ne sera pas cryptée
- soit on tape un mot de passe et la clé sera cryptée
*conseillé de ne pas changer le dossier d'enregistrement par défaut et de mettre 1 passphrase
2-/ Envoyer la clé publique au serveur
il faut envoyer au serveur la clé publique pour qu'il puisse crypter ses messages
la clé public se trouve dans **~/.ssh/id_rsa.pub** (noter que **.ssh** est un dossier caché)
la clé privée se trouve dans **~/.ssh/id_rsa** (ne jamais la communiquer à qqun!)
elle sera de plus cryptée si on a choisit de la crypter avec le passphrase
knownhosts est la liste de fingerprint que notre PC tient a jour
on doit envoyer **id_rsa.pub et l'ajouter au fichier authorized_keys** ou le serveur garde la liste des clés autorisées
ssh-copy-id -i id_rsa.pub login@ip pour envoyer une copie
ssh-copy-id -i id_rsa.pub "-p 14521 mateo21@88.92.107.7" si besoin d'autre port
la clé sera automatiquement ajouté à **~/.ssh/authorized_keys sur le serveur**
ssh login@ip pour se connecter ensuite

l'agent SSH (prog tournant en bg rentenant les private keys pdt tte la durée de la session)
ssh-add (à faire sur le PC client) pour add le passphrase qui fera tout automatiquement
l'agent ssh va cherchera la clé privé, et demandera le passphrase pour decrypter puis tt auto.
PROBLEME DE CO EN SSH SUR ORDI FIXE AVEC IP INTERNET

Authentification par clé depuis Windows (PuTTY en version installée)
CF LES DERNIERES FICHES POUR AVOIR LES INFOS DE CETTE PARTIE

CHAPITRE 8 Transférer Des Fichiers

wget : téléchargement de fichiers
wget http://cdimage.debian.org/ear.iso pr stopper a tout moment le chargement avec **Ctrl+C**
wget -c http://cdimage.debian.org/ear.iso pour reprendre un téléchargement arrêté
wget --background http://cdimage.debian.org/car.iso pr mettre dl en bg, on peut aussi utiliser **nohup**
* l'avancement du dl sera écrit in a file **wget-log**.

scp : copier des fichiers sur le réseau (Secure CoPy)
scp fichier_origine copie_destination
scp login_serveur@ip_serveur:fichier_origine login_client@ip_client:copie_destination
si l'on n'ajoute pas de login@ip la commande comprendra que c'est sur le serveur
Copier un fichier de votre ordinateur vers un autre
scp image.png mateo21@85.123.10.201:/home/mateo21/images/
Copier un fichier d'un autre ordinateur vers le vôtre
scp mateo21@85.123.10.201:image.png copie_image_sur_mo n_pc.png
scp mateo21@85.123.10.201:image.png . le point signifie copier le fichier dans le working directory
pr use un port spécific ac la cmd scp il faut mettre 1 **-P port** alors quac ssh on use un **-p port**
ftp & sftp : transférer des fichiers (FileZila prog graphique de FTP)
FTP (File Transfer Protocol) est un protocole permettant d'échanger des fichiers sur le réseau
On l'utilise généralement dans 2 cas:
- pr dl 1 file depuis 1 serveur FTP public. En general les navigateurs web le font de maniere autom et transparente qd on clique sur 1 lien de dl. La co se fait alors de manière **anonyme**
- pr trasfert files vers un serveur FTP privé.Qd on prend 1 hebergement pr 1 site, l'hébergeur ns donne en géne des acces FTP pr y déposer files. La co se fait alors en mode **authentifié**
Connexion à un serveur FTP (i.e. **ftp ftp.debian.org**)
Les serveurs FTP repondent en demandant un login et un mdp
pour les serveurs publics, le login est toojours **anonymous** et le mdp nimporte quoi
ftp> permet de taper des cmdes
Se déplacer au sein du serveur FTP (**ls** , **pwd** , **cd**)
Le transfert de fichiers(**put** send un file vers le serveur,**get** dl un file depuis serveur,**delete**)
get possible pour serveurs publiques mais pas put, chmod (évidement)
get README pr dl le fichier README /**!pwd** pr savoir le folder de reception des dl
!cd pr changer de folder de reception /**!ls** pr afficher ce quon a dans notre dossier de reception
*les commandes avec un ! s'effectuent sur notre PC et pas sur le serveur FTP

Les autres commandes (**man ftp** pr obtenir aperçu des cmdes a effectuer sur 1 serveur ftp)
sftp : un FTP sécurisé (i.e. **sftp login@ip** / **man sftp**)
le protocole FTP a pour défaut de ne pas etre sécurisé (les données ne sont pass sécuisées)
* qqun qui a acces au réseau peut intercepter le contenu des files qu'on echange ou notre mdp
* pour y remédier on a inventé sftp qui repose sur SSH pour securiser la connexion
la clé publique sera utilisée si elle est présente (Tester avec lagent ssh et en reel ac ad FTP)
les commandes sont quasi identiques a FTP (**get** , **put**, **rm** et plus mtn **delete**)
* pr se co en sftp on utilise le port 22 par defaut / ojd sftp encore assez rare et ftp souvent)

rsync : synchroniser des fichiers pour une sauvegarde
rsync conserver les files ds un sauvegarde, enregistrera les chgt et non tout tout le tps
il verifie les != entre 2 dossiers puis copie ! les chgts (qui est la signification d'incrémentiel)
sauvegarder dans un autre dossier du même ordinateur
rsync -arv Images/ Backups/ pour sauvegarder les dossier Images dans un dossier backups
-a : conserve toutes les informations sur les fichiers (droits, date de modification, etc.)
-r : sauvegarde aussi tous les sous-dossiers qui se trouvent dans le dossier à sauvegarder
-v : verbose
rsync - arv Images/ backups/ une 2nde fois après y avoir créé 1 file, copiera ! le fichier créé
Supprimer les fichiers en trop dans le repertoire de sauvegarde
rsync -arv --delete Images/ backups/ pour demander que le contenu soit strict. identique
Sauvegarder les fichiers supprimés
--backup pour ajouter un suffixe aux fichiers del ds le repertoire de sauvegarde
--backup-dir=/chemin/vers/le/repertoire pour deplacer les files del ds un autre repertoire
rsync -arv --delete --backup –backup-dir=/home/mateo21/backups_supprimes Images/ backups/
Il est conseillé d'utiliser l'option --backup-dir tout le temps au cas où et dans un autre dossier que le dossier de sauvegarde pour éviter les pb lors de la synchronisation
–exclude pour exclure un dossier de la sauvegarde

Sauvegarder sur un autre ordinateur
rsync peut copier les fichiers en employant plusieurs methodes !=
la plus courante est de passser pasr SSH qui permet de sécuriser tout type de transfert

rsync -arv --delete --backup --backup-dir=/home/mateo21/fichiers_supprimes/ Images/ mateo21@IP_du_serveur:mes_backups/

-e "ssh -p port" pour changer de port si le serveur en SSH est sur un autre port

rsync -arv --delete --backup –backup-dir=/home/mateo21/fichiers_supprimes/ Images/ mateo21@IP_du_serveur:mes_backups/ -e "ssh -p 12473"

Analyser le réseau et filtrer le trafic avec un pare-feu
host & whois : qui êtes-vous ?
* Les ordinateurs reliés à internet sont identifiés par une adresse IP
* aujourd'hui on utilise les adresses IP au format IPV4 mais de plus en plus au format IPV6
fe80::209:62fa:fb80:29f2 format IPV6
86.172.120.28 format IPV4
* on peut associer à chaque IP un nom d'hôte (**hostname**)
host commande qui permet d'effectuer la conversion IP <=> hostname
host siteduzero.com ou **host 92.243.25.239** pour convertir
Gérer les noms d'hôte personnalisés
les associations entre adresses IP et noms d'hôtes s'appellent les **serveurs DNS**
Chaque FAI possède des serveurs DNS fournissant la liste des équivalences IP<=>hostname
On peut néanmoins établir notre propre liste sur notre ordinateurs
sudo nano /etc/hosts on pourrait y ajouter **92.243.25.239 siteduzero.com**
* en ouvrant le navigateur et en notant siteduzero.com on irait à l'adresse IP notée
Cette technique à l'avantage de forcer l'association ms il faut la mettre a jour régulièrement !
Sur un réseau local il peut être pratique d'associer un hostname à une IP
192.168.0.5 pc-papa ecrire pc-papa reviendra à noter l'IP

whois : tout savoir sur un nom de domaine
Chaque nom de domaine doit obligatoiremt indiquer qui se trouve derrière C'EST UNE REGLE:
nom – prénom – adresse – moyen de contact
whois permet d'obtenir ces informations sur un nom de domaine

ifconfig & netstat : gérer et analyser le trafic réseau
ifconfig : liste des interfaces réseau et reglages reseaux
ifconfig interface etat etat =up|down (i.e:**ifconfig eth0 down**) pr **activer** ou non l'interface
1 PC possède souvent +sieurs **interfaces réseau** (+sieurs moyen de se connecter a internet)
* l'interface **lo** est la **boucle locale** elle correspond à la connexion à nous même

netstat : statistiques sur le réseau
netstat -i : statistiques des interfaces réseau
netstat -uta : lister toutes les connexions ouvertes
netstat -atn : afficher les numeros de ports plutot qu'une description en toute lettre
netstat -lt : liste des connexions en état d'écoute
-i : afficher les stats des interfaces réseaux
-u : afficher les connexions UDP ;
-t : afficher les connexions TCP ;
-a : afficher toutes les connexions quel que soit leur état
-l : afficher les connexions etant à l'état LISTEN
-s : afficher les statistiques du reseau résumées
-n : afficher le umero des ports plutot que leur noms

il existe != types d'états:
ESTABLISHED la connexion a été établie avec l'ordinateur distant
TIME_WAIT co wait le traitemt de ts les paquets encore sur le réseau avant debut la fermeture
CLOSE_WAIT le serveur distant a arrêté la co de lui-mm
CLOSED la connexion n'est pas utilisée
CLOSING la fermeture de la co a debuté ms ttes les données n'ont pas encore été envoyées
LISTEN à l'écoute des connexions entrantes
* **22** = SSH / **21** = FTP / **80** = WEB
* **TCP** et **UDP** sont deux protocoles différents pour envoyer des données sur le réseau
* **UDP** + used ds les jeux réseaux et pr les commu vocales / **TCP** est le protocole le + utilisé

iptables : le pare-feu de référence (en root)
iptables permet d'établir un certain nombre de **règles** sur les ports
La technique est de bloquer par défaut **tous les ports et à n'en autoriser que quelqu'uns**
il y a des portes d'entrée et des portes de sortie sur les ordinateurs
iptables -L : afficher les règles
Chain INPUT : correspond aux règles manipulant le trafic entrant
Chain FORWARD : correspond aux règles manipulant la redirection du trafic
Chain OUTPUT : correspond aux règles manipulant le trafic sortant
(policy ACCEPT) signifie que, par défaut, tt le trafic est accepté
(policy DROP) signifie que l'on ignore tous les autres paquets

Le principe des règles
l'ordre des règles est important ! (les règles sont numérotées)
iptables lit de ht en bas et la posit° des règles influe sur le résultat final
--line-numbers pour avoir les numéros des règles
target pr know ce que fait la règle.
ACCEPT signifie que cette ligne autorise un port et/ou une IP
prot indique le protocole utilisé (**tcp**, **udp**, **icmp**)
ICMP permet à votre ordinateur de répondre aux requêtes de type « **ping** »
source indique l'IP de source, **INPUT** => indique l'IP du PC distant qui se co à nous
destination indique l'IP de destination, **OUTPUT** =>indique l'IP du PC auquel on se co
la dernière colonne indique le port

Ajouter et supprimer des règles
-A chain : add une règle en fin de liste pr la chain indiquée (**INPUT** ou **OUTPUT**)
-D chain rulenum : supprime la règle n° rulenum pour la chain indiquée.
-I chain rulenum : insère une règle au milieu de la liste à la position indiquée par rulenum. Si rulenum pas indiqué, la règle sera insérée en premier (en tête)
-R chain rulenum : remplace la règle n° rulenum dans la chain indiquée.
-L : liste les règles (nous l'avons déjà vu).
-F chain : vide toutes les règles de la chain indiquée.
-P chain regle : modifie la règle par défaut pour la chain. I.e. par défaut ts les ports st fermés, sauf ceux que l'on a indiqués dans les règles.

iptables -A (chain) -p (protocole) -- dport (port) -j (d é cision)

Remplacez chain par la section qui vous intéresse (INPUT ou OUTPUT), protocole par le nom du protocole à filtrer (TCP, UDP, ICMP. . .) et enfin par la décision à prendre : ACCEPT pour accepter le paquet, REJECT pour le rejeter ou bien DROP pour l'ignorer complètement.

iptables -A INPUT -p tcp --dport ssh -j ACCEPT add à la section INPUT (donc pour le trafic entrant) une règle sur les données reçues via le protocole TCP sur le port de SSH (on peut mettre ssh ou le n° du port 22). Lorsque le PC recevra des données en TCP sur le port de SSH, elles seront acceptées ; permettra permet de se connecter a distance au PC via SSH
iptables -A INPUT -p tcp --dport www -j ACCEPT idem pour le web (80).
iptables -A INPUT -p tcp --dport imap2 -j ACCEPT idem pour les mails
Si on ne précise pas de port (en omettant dport), tous les ports seront acceptés !

Autoriser les pings

En + d'autoriser le trafic sur ces ports, il est conseillé d'aussi accepter le protocole **ICMP** (ping)
iptables -A INPUT -p icmp -j ACCEPT comme on n'a pas indiqué de section –dport, cette règle s'applique à tous les ports, mais pour les pings (icmp) uniquement !
* Votre ordinateur répondra alors aux « **pings** » pour indiquer qu'il est bien en vie.

Autoriser les connexions locales et déjà ouvertes

Pour l'instant, nos règles sont encore un peu trop restrictives et pas vraiment utilisables (vous risquez de ne plus pouvoir faire grand-chose). Je vous propose d'ajouter deux règles pour « assouplir » un peu votre pare-feu et le rendre enfin utilisable.
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m state -- state ESTABLISHED , RELATED -j ACCEPT
1-/ La 1ière règle autorise tout le trafic sur l'interface de **loopback locale** grâce à -i lo. Il n'y a pas de risque à autoriser le PC à communiquer ac lui-mm, d'autant + qu'il en a parfois besoin !
2-/ La 2ieme règle autorise ttes les co qui sont déjà à l'état **ESTABLISHED** ou **RELATED**. En clair, elle autorise toutes les connexions qui ont été demandées par votre PC : permet d'assouplir le pare-feu et de le rendre fonctionnel pour une utilisation quotidienne.

Refuser toutes les autres connexions par défaut

Il reste un point essentiel à traiter car, pour l'instant, ce filtrage ne sert à rien.
En effet, nous avons indiqué quelles données nous autorisions, mais nous n'avons pas dit que toutes les autres devaient être refusées !
Changez donc la règle par défaut pour DROP par exemple :
 # iptables -P INPUT DROP
iptables devrait maintenant indiquer que par défaut tout est refusé, sauf ce qui est indiqué par les lignes dans le tableau :
Le filtrage est radical. Nous n'avons pas autorisé beaucoup de ports et il se pourrait que vous vous rendiez compte que certaines applications n'arrivent plus à accéder à l'internet (normal, leur port doit être filtré).
À vous de savoir quels ports ces applications utilisent pour modifier les règles en conséquence. Au besoin, pensez à faire de même pour les règles de sortie (OUTPUT).

Appliquer les règles au démarrage

Si vous redémarrez votre ordinateur, les règles **iptables** auront disparu ! Le seul moyen pour qu'elles soient chargées au démarrage consiste à créer un script qui sera exécuté au démarrage (On verra ça plus loin)

CHAPITRE 9 Compiler Un Programme Depuis Les Sources

Essayez d'abord de trouver un paquet .deb

Certains programmes récents ou encore en développement ne sont pas disponibles via apt-get

trouver sur le site web du logiciel un paquetage **.deb**. C'est en quelque sorte l'équivalent du programme d'installation, mais celui-ci est spécifique à Debian et à ses distributions dérivées (dont fait partie Ubuntu). Les **.deb** ne fonctionnent pas sur les distributions utilisant d'autres outils ; **Red Hat** utilise des .**rpm** par exemple. Notez que le programme **alien** est capable de convertir un **.rpm** en **.deb** au besoin.

Une x le **.deb** dl, double-cliquez dessus. Une fenêtre apparaît pr vs proposer d'installer le logiciel

Si aucune erreur n'apparaît, vous avez de la chance, vous pouvez procéder à l'installation.
Sinon, cela signifie :
– soit que vous avez dl un **.deb** ne correspondant pas à votre machine. Vérifiez que vous n'avez pas pris une version 32bits au lieu de 64bits (ou inversement) ;
– soit qu'il vous manque des dépendances pour pouvoir installer convenablement le prog. Et là, ça peut vite devenir un casse-tête! Il faut d'abord installer le prog manquant avant d'aller + loin

Si même le paquetage **.deb** n'est pas disponible, il ne reste alors qu'une solution : récupérer le code source du programme et le compiler soi-même. On peut ainsi créer un exécutable spécialement optimisé pour sa machine. L'exécutable est l'équivalent du .exe de Windows, même s'il n'a en général pas d'extension sous Linux.

Quand il n'y a pas d'autre solution : la compilation

Si le prog que vous recherchez n'est pas dans les dépôts (apt-get) et que vous ne parvenez pas non plus à trouver de .deb prêt à l'emploi sur le web, vous allez devoir le compiler depuis ses sources. La compilation est un procédé qui permet de transformer le code source d'un programme en un exécutable que l'on peut utiliser.

Les étapes de la compilation peuvent varier d'un programme à un autre.

Compilation d'un programme pas à pas

Pour compiler des prog, vous aurez besoin avant toute chose d'installer les outils de compilation. Pour cela, rien de plus simple, il suffit d'installer le paquet **build-essential**

sudo apt-get install build–essential

je vous propose d'apprendre à compiler un petit programme assez simple: **htop** (dérivé de top)

Vous allez télécharger une archive compressée .tar.gz

tar zxvf htop-0.8.3.tar.gz

cd htop -0.8.3

Pour le moment, un seul prog nous intéresse : configure. Exécutez-le comme suit

./configure

configure est un prog qui analyse le PC et qui vérifie si ts les outils nécessaires à la compilat° du logiciel que vous souhaitez installer sont bien présents.. Un des premiers éléments qu'il va vérifier est la présence du compilateur (checking for gcc...) que vous avez normalement dû installer un peu plus tôt avec le paquet **build-essential** .Malheureusement, il arrivera fréquemment que configure affiche une erreur en raison d'un manque de dépendances. Dans notre cas, il devrait afficher une erreur comme celle-ci

```
checking for sys / time . h ... yes
checking for unistd . h ... ( cached ) yes
checking curses . h usability ... no
checking curses . h presence ... no
checking for curses . h ... no
configure : error : missing headers : curses . h
```

L'erreur (sur la dernière ligne) indique en anglais « missing headers: curses.h ».

C'est là que les choses se corsent : il faut installer l'élément manquant, en l'occurrence ces fameux headers de curses.h. Si vous n'êtes pas programmeurs, vous n'avez probablement aucune idée de ce dont il s'agit. La technique la plus efficace consiste à effectuer une recherche de la ligne d'erreur sur le web, accompagnée de préférence du mot-clé « ubuntu ». Lancez donc une recherche de « configure : error : missing headers : curses.h ubuntu ».

L'information à chercher est le nom du paquet manquant que vous devez installer. En lisant les forums, vous devriez finir par trouver le nom du paquet que vous recherchez : libncurses5-dev. En l'occurrence, il suffit d'installer ce paquet via apt-get pour ne plus avoir l'erreur indiquée dans configure

sudo apt-get install libncurses5 –dev

Une x le paquet installé, relancez **configure** et esperer pour que l'erreur disparaisse.

Si configure n'affiche plus la même erreur, vous avez gagné (pour le moment). Il reste maintenant deux possibilités :

– soit vous avez une nouvelle erreur et vous devrez la résoudre de la même manière : en effectuant une recherche sur l'internet pour comprendre ce qui ne va pas. Le plus souvent, il suffira d'installer le paquet manquant avec apt-get ;

– soit vous n'avez pas d'erreur et configure parvient jusqu'à son terme. Victoire !

e programme est prêt à être compilé ! Rassurez-vous, le plus dur est derrière vous. :-)

Il suffit maintenant de lancer la compilation à l'aide d'une commande toute simple

make

Durant la compilation, des lignes barbares s'afficheront dans votre console. Vous ne devriez pas avoir à vous en préoccuper, tous les problèmes ayant normalement été détectés auparavant par configure

ne fois la compilation terminée, l'exécutable devrait avoir été créé. Il ne reste plus qu'à l'installer, c'est-à-dire à le copier dans le bon répertoire. Là encore, vous n'avez pas à vous poser beaucoup de questions. Exécutez la commande suivante :

sudo make install

Une fois que cela est fait, le programme est installé ! Nous pouvons à présent exécuter htop en tapant le nom de la commande **htop**

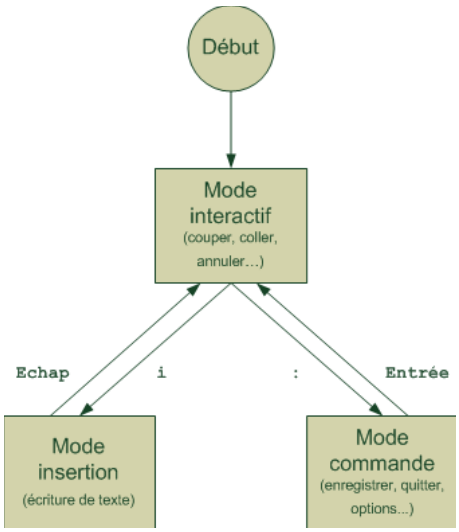
Si vous souhaitez désinstaller le programme, il suffit d'exécuter cette commande depuis le répertoire où vous l'avez compilé :

sudo make uninstall Vous pouvez sans problème supprimer le répertoire contenant les fichiers sources (celui depuis lequel vous avez compilé). Toutefois, il ne sera alors plus possible de lancer la commande de désinstallation.

CHAPITRE 9 Vim : L'éditeur De Texte Du Programmeur

Installer Vim
Sous Linux, 2 éditeurs de textes puissants : **Vim** et **Emacs**
Installer et lancer Vim
sudo apt-get install vim
Vimtutor : le programme qui vous apprend à utiliser Vim !
vimtutor pour avoir un tutoriel de vim
Les modes d'édition de Vim
vim
Vim possede 3 modes de travail != :
mode interactif: mode par defaut. En lancant vim on est eja dessus. Mode puissant ! On peut avec des racourcis faire pas mal de choses (couper coller , annuler,etc)
Chaque action peut etre declanché en appuyant sur une touche donc faire attention !
mode insertion: modee que l'on connaît. Pour en sortir **echap**
mode commande: ce mode permet de lancer des commandes telles que quitter,save,etc.
On peut y activer la coloration automatique, l'affichage du numero des lignes. On peut envoyer des commandes au shell telles que ls, locate,cp,...
pour activer ce mode on doit etre en mode interactif et appuyer sur la touche «**:**»
on validera avec «**Entrée**»

on peut installer la version graphique **gVim**



Opérations basiques (déplacement, écriture, enregistrement)
L'ouverture de Vim
vim pour lancer vim
vim nomdufichier pour ouvrir un fichier ou le créer

i : insérer du texte
appuyer sur **i** permet de passer en mode insertion pour insérer du texte
-INSERT- en bas de l'écran confirme qu'on est en mode insertion
On ecrit quelques lignes puis on appuie sur **ECHAP** pour revenir au mode interactif (le mode -INSERT- a disparu) on est en mode interactif
Le déplacement
il est possible de déplacer le curseur au sein du texte
h, j, k, l : se déplacer dans tous les sens (les flèches aussi)
0 et $: se déplacer en début et fin de ligne
:w : enregistrer le fichier
:w monfichier pour enregistrer au nom de monfichier
:q : quitter
vim interdit de quitter si on na pas enregistré
:q! pour forcer la fermeture
:wq : enregistrer puis quitter
Opérations standard (copier, coller, annuler. . .) (en mode interactif)
x : effacer des lettres (appuyer sur un chiffre puis x pour supprimer le nombre de lettre)
d : effacer des mots, des lignes. . .
dd : supprimer une ligne (chiffre dd pour supprimer le nombre de lignes)
la ligne ainsi supprimée est en fait « coupée » et placée en mémoire.
Elle peut être collée, comme on le verra plus loin, avec la touche p.
dw : supprimer un mot
si l'on met le curseur sur la première lettre d'un mot puis dw on del le mot
si on met le curseur a une certaine lettre , seul les lettres suivant du mot seront delgroup
3dw ou d3w pour supprimer 3 mots
d0 et d$: supprimer le début ou la fin de la ligne
– En tapant d0, vous supprimez du curseur jusqu'au début de la ligne.
– En tapant d$, vous supprimez du curseur jusqu'à la fin de la ligne.
yy : copier une ligne en mémoire
yw pour copier un mot
y$ pour copier du curseur à la fin de la ligne
p : coller
si vous avez copié une ligne en mémoire et que vous appuyez sur p, elle sera collée sur la ligne située après le curseur.
8p collera 8 fois la prhase en mémoire
r : remplacer une lettre
rs changera la lettre du curseur par s

R pour passer en mode remplacement
u : annuler les modifications
Ctrl + R pour annuler l'annulation
G : sauter à la ligne n° X
la numérotation des lignes commence à 1
7G pour passer a la ligne 7
gg pour revenir à la ligne 1
G pour arriver à la derniere ligne
7gg pour ariver a la ligne 7
Opérations avancées (split, fusion, recherche. . .)
/ : rechercher un mot
appuyer sur n pour passer au suivant
appyuyer sur N pour revenir au precedent
? a la place de / pour lancer une recherche depuis le debut du texte
:s : rechercher et remplacer du texte
:s/ancien/nouveau : remplace la première occurrence 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. C'est peut-être ce que vous utiliserez le plus fréquemment.
:r : fusion de fichiers
:r autrefichier inserera le fichier autrefichier a partir du curseur
Le découpage d'écran (split)
Vim permet aussi de splitter l'écran
:sp : découper l'écran horizontalement
:sp pour scinder l'écran en deux horizontalement
le fichier sera ouvert une seconde fois permettant de voir dux endroits!= du fichier1
:sp autrefichier pour ouvrir autrefichier en 2ieme ecran
:vsp : découper l'écran verticalement
Les principaux raccourcis en écran splitté
chaque morceau de l'écran est appelé **viewport**
– Ctrl + w puis Ctrl + w : navigue de viewport en viewport. Répétez l'opération plusieurs fois pour accéder au viewport désiré.
– Ctrl + w puis j : déplace le curseur pour aller au viewport juste en dessous. La même chose fonctionne avec les touches h, k et l que l'on utilise traditionnellement pour se déplacer dans Vim.
– Ctrl + w puis + : agrandit le viewport actuel.
– Ctrl + w puis - : réduit le viewport actuel.
– Ctrl + w puis = : égalise à nouveau la taille des viewports.
– Ctrl + w puis r : échange la position des viewports. Fonctionne aussi avec « R » majuscule pour échanger en sens inverse.
– Ctrl + w puis q : ferme le viewport actuel.
:! : lancer une commande externe
par exemple :!ls
Les options de Vim
Vim peut être personnalisé de deux façons différentes :
– En activant ou désactivant des options. La documentation complète des options est disponible en ligne.
– En installant des plugins. Voyez la page officielle des plugins les plus téléchargés de Vim
Le fonctionnement des options
Les options peuvent être activées après le démarrage de Vim en lançant des commandes. Cependant, ces options seront « oubliées » dès que vous quitterez le logiciel. Si vous voulez que les options soient activées à chaque démarrage de Vim, il faut créer un fichier de configuration .vimrc dans votre répertoire personnel
Activer des options en mode commande
La première méthode consiste à activer l'option en mode commande. Une fois Vim ouvert, pour activer l'option nommée « option », tapez :

:set option
Pour la désactiver, tapez :
:set nooption
Il faut donc ajouter le préfixe no devant le nom de l'option pour la désactiver.
Certaines options doivent être précisées avec une valeur, comme ceci :
:set option=valeur
Pour connaître l'état d'une option :
:set option?

Activer des options dans un fichier de configuration
C'est à mon avis la meilleure façon de procéder. Commencez par copier un fichier de configuration déjà commenté qui vous servira d'exemple : il y en a un dans /etc/vim qui s'appelle vimrc

syntax : activer la coloration syntaxique
en fonction du type de fichier ouvert, vim colorera le texte
Vim supporte un très très grand nombre de langages de programmation : C, C++, Python, Java, Ruby, Bash, Perl, etc.
syntax on pour activer l'option
Notez qu'il faut enregistrer, quitter et relancer Vim pour que le changement soit pris en compte. . . sauf bien sûr si vous activez l'option à la volée en tapant dans Vim :set syntax=ON.

background : coloration sur un fond sombre
set background = dark pour pls de lisibilité sur les colorations
Les couleurs seront largement plus adaptées.

number : afficher les numéros de ligne
set number

showcmd : afficher la commande en cours
Lorsque vous écrivez une commande comme 2dd pour supprimer deux lignes, vous écrivez à l'aveugle. Vous ne voyez pas ce que vous avez écrit.
Set showcmd

ignorecase : ignorer la casse lors de la recherche
set ignorecase

mouse : activer le support de la souris
set mouse = a
Désormais, vous pourrez cliquer avec la souris sur une lettre pour y déplacer le curseur directement. Vous pourrez également utiliser la molette de la souris pour vous déplacer dans le fichier.
Il vous sera également possible de sélectionner du texte à l'aide de la souris. Vous passerez alors en mode visuel. Dans ce mode, vous pouvez supprimer le texte sélectionné (avec x, comme d'habitude), mais aussi mettre le texte tout en majuscules (U), minuscules (u), etc.

CHAPITRE 9 : INTRODUCTION AUX SCRIPTS SHELL

sous Linux il existe deux environnements differents: l'env console / l'env graph
il existe differents env console : les shells
les principaux shells sont :
– sh : Bourne Shell. L'ancêtre de tous les shells.
– bash : Bourne Again Shell. Une amélioration du Bourne Shell, disponible par défaut sous Linux et Mac OS X.
– ksh : Korn Shell. Un shell puissant assez présent sur les Unix propriétaires, mais aussi disponible en version libre, compatible avec bash.
– csh : C Shell. Un shell utilisant une syntaxe proche du langage C.
– tcsh : Tenex C Shell. Amélioration du C Shell.
– zsh : Z Shell. Shell assez récent reprenant les meilleures idées de bash, ksh et tcsh.

Le bash (Bourne Again Shell) est le shell par défaut de la plupart des distributions Linux mais aussi celui du terminal de Mac OS X.

Le shell est le programme qui gère l'invite de commandes. C'est donc le programme qui attend que vous rentriez des commandes
C'est aussi le programme qui est capable par exemple de :
– se souvenir quelles étaient les dernières commandes tapées (vous remontez dans votre historique en appuyant sur la flèche « Haut » ou en faisant une recherche avec un Ctrl + R) ;

– autocompléter une commande ou un nom de fichier lorsque vous appuyez sur Tab (figure 28.3) ;
– gérer les processus (envoi en arrière-plan, mise en pause avec Ctrl + Z. . .) ;
– rediriger et chaîner les commandes (les fameux symboles >, <, |, etc.) ;
– définir des alias (par exemple ll signifie chez moi ls -lArth).

Installer un nouveau shell

apt-get install ksh pour l'installer
chsh pour le lancer
chsh signifie Change Shell.
On vous demandera où se trouve le programme qui gère le shell. Vous devrez indiquer /bin/ksh pour ksh, /bin/sh pour sh, /bin/bash pour bash, etc.

un script shell dépend d'un shell précis car le langage n'est pas tout à fait le même selon que vous utilisez sh, bash, ksh, etc
Il est possible d'écrire des scripts sh par exemple. Ceux-là, nous sommes sûrs qu'ils fonctionnent partout car tout le monde possède un shell sh. Il s'agit toutefois du plus vieux shell, or écrire des scripts en sh est certes possible mais n'est franchement ni facile, ni ergonomique.

Je propose d'étudier le bash dans ce cours car :
– on le trouve par défaut sous Linux et Mac OS X (cela couvre assez de monde !) ;
– il rend l'écriture de scripts plus simple que sh ;
– il est plus répandu que ksh et zsh sous Linux

Notre premier script

vim essai.sh
J'ai donné ici l'extension .sh à mon fichier. On le fait souvent par convention pour indiquer que c'est un script shell, mais sachez que ce n'est pas une obligation. Certains scripts shell n'ont d'ailleurs pas d'extension du tout. J'aurais donc pu appeler mon script essai tout court.

Indiquer le nom du shell utilisé par le script

La première chose à faire dans un script shell est d'indiquer. . . quel shell est utilisé
#!/bin/bash pour dire que notre script est un shell bash
Le #! est appelé le sha-bang. /bin/bash peut être remplacé par /bin/sh si vous souhaitez coder pour sh, /bin/ksh pour ksh, etc.
En l'absence de cette ligne, c'est le shell de l'utilisateur qui sera chargé. Cela pose un problème : si votre script est écrit pour bash et que la personne qui l'exécute utilise ksh, il y a de fortes chances pour que le script ne fonctionne pas correctement !
La ligne du sha-bang permet donc de « charger » le bon shell avant l'exécution du script. À partir de maintenant, vous devrez la mettre au tout début de chacun de vos scripts.

Exécution de commandes

Après le sha-bang, nous pouvons commencer à coder.
On retrouve :
ls : pour lister les fichiers du répertoire.
cd : pour changer de répertoire.
mkdir : pour créer un répertoire.
grep : pour rechercher un mot.
sort : pour trier des mots.
etc.
par exemple :

```
#!/ bin / bash
ls
```

Les commentaires

```
#!  /bin/bash
# Affichage de la liste des fichiers
ls
```

la premiere ligne est un commentaire spéciale

Exécuter le script bash
Donner les droits d'exécution au script
avec un ls -l on voit
-rw -r - -r - - 1 mateo21 mateo21 17 2009 -03 -13 14:33 essai.sh
-rw-r–r–
chmod +x essai.sh

Exécution du script
./ essai.sh pour lancer l script
Les commandes seront exécutées une par une
Exécution de débogage
Plus tard, vous ferez probablement de gros scripts et risquerez de rencontrer des bugs.
Il faut donc dès à présent que vous sachiez comment déboguer un script.
Il faut l'exécuter comme ceci :
$ bash -x essai.sh pour lacer le mode debuggage

Le shell affiche alors le détail de l'exécution de notre script
$ bash -x essai . sh
+ pwd
/ home / mateo21 / scripts
+ ls
essai.sh

Créer sa propre commande
Actuellement, le script doit être lancé via ./essai.sh et vous devez être dans le bon répertoire. Sinon vous devez taper le chemin en entier : /home/mateo21/scripts/essai.sh.
Comment font les autres programmes pour pouvoir être exécutés depuis n'importe quel répertoire sans « ./ » devant ?
Ils sont placés dans un des répertoires du PATH. Le PATH est une variable système qui indique où sont les programmes exécutables sur votre ordinateur. Si vous tapez echo $PATH vous aurez la liste de ces répertoires « spéciaux ».
Il vous suffit donc de déplacer ou copier votre script dans un de ces répertoires, comme /bin, /usr/bin ou /usr/local/bin (ou encore un autre répertoire du PATH). Notez qu'il faut être root pour pouvoir faire cela.
Une fois que c'est fait, vous pourrez alors taper simplement essai.sh pour exécuter votre programme et ce quel que soit le répertoire dans lequel vous vous trouverez !

CHAPITRE 10 : AFFICHER ET MODIFIER DES VARIABLES

Déclarer une variable
vim variables.sh pour créer un nouveau script
#!/ bin / bash
message = ' Bonjour tout le monde '
NE PAS METTRE DESPACE AUTOUR DU SYMBOLE «=»
Pour inserer un symbole spécial tel qu une apostrophe il faudra mettre \ (caractere d'échappement)
message='Bonjour c\' est moi '
Il met en mémoire le message Bonjour tout le monde, et c'est tout! Rien ne s'affiche à l'écran !
echo : afficher une variable
echo Salut tout le monde
Chacun des mots était considéré comme un paramètre que echo a affiché.
echo «Salut tout le monde»
Salut tout le monde sera considéré comme une et une seule variables
\n pour un retour a la ligne
echo -e " Message \ nAutre ligne " pour afficher avec le symbole \n
Afficher une variable
echo $message pour afficher la variable message

#! /bin/bash
message=’Bonjour tout le monde ’
echo ’Le message est : $message’

on aura un soucis ; il s’affichera Le message est : $message

Les quotes
les quotes permettent de délimiter un parametre contenant des espaces
– les apostrophes ’ ’ (simples quotes) ;
– les guillemets « » (doubles quotes) ;
– les accents graves ‘ ‘ (back quotes), qui s’insèrent avec Alt Gr + 7 sur un clavier AZERTY français.

Les simples quotes ’ ’
message = ’ Bonjour tout le monde ’
echo ’ Le message est : $message ’
Avec de simples quotes, la variable n’est pas analysée et le $ est affiché tel quel.
Et on aura d’affiché
Le message est : $message

Les doubles quotes « »

message = ’ Bonjour tout le monde ’
echo " Le message est : $message "

on aura d’affiché
Le message est : Bonjour tout le monde

La variable sera donc affichée avec les doubles quotes
avec les doubles quotes , les variables sont ifdentifiées

Les back quotes ‘ ‘
Un peu particulières, les back quotes demandent à bash d’exécuter ce qui se trouve à l’intérieur.
Par exemple
message=‘pwd‘
echo " Vous êtes dans le dossier $message "

read : demander une saisie
Vous pouvez demander à l’utilisateur de saisir du texte avec la commande read. Ce texte sera immédiatement stocké dans une variable.
read nomvariable

par exemple
#!/ bin / bash
read nom
echo " Bonjour $nom !"

Affecter simultanément une valeur à plusieurs variables
#!/ bin / bash
read nom prenom
echo " Bonjour $nom $prenom !"

Si vous rentrez plus de mots au clavier que
vous n’avez prévu de variables pour en stocker, la dernière variable de la liste récupèrera tous les mots restants.

-p : afficher un message de prompt
#!/ bin / bash
read -p ’Entrez votre nom :’ nom
echo " Bonjour $nom !"

Notez que le message ’Entrez votre nom’ a été entouré de quotes. Si on

ne l’avait pas fait, le bash aurait considéré que chaque mot était un paramètre différent !

-n : limiter le nombre de caractères

```
#!/ bin / bash
read -p ’ Entrez votre login (5 caract è res max ) : ’ -n 5 nom
echo " Bonjour $nom !"
```

```
#!/ bin / bash
read -p ’ Entrez votre login (5 caract è res max ) : ’ -n 5 nom
echo -e "\ nBonjour $nom !"
```

pour eviter que le texte soit sur la meme ligne si on depasse le texte qui est limité a 5 caractères et qui empechera d’entrer sur entrée a la fin de la saisie

-t : limiter le temps autorisé pour saisir un message

```
#! /bin/bash
read -p ’ Entrez le code de désamorçage de la bombe vous avez 5 secondes:’ -t 5 code
echo -e "\ nBoum !"
```

-s : ne pas afficher le texte saisis :

```
#!/ bin / bash
read -p ’ Entrez votre mot de passe : ’ -s pass
echo -e "\ nMerci ! Je vais dire à tout le monde que votre mot de
passe est $pass ! :) "
```

donnera

```
Entrez votre mot de passe :
Merci ! Je vais dire à tout le monde que votre mot de passe est
supertopsecret38 !
```

Effectuer des opérations mathématiques

En bash, les variables sont toutes des chaînes de caractères
En soi, le bash n’est pas
vraiment capable de manipuler des nombres ; il n’est donc pas capable d’effectuer des opérations
la commande est let

```
let " a = 5"
let " b = 2"
let " c = a + b "
```

```
#!/ bin / bash
let " a = 5"
let " b = 2"
let " c = a + b "
echo $c
```

Les opérations utilisables sont :
l’addition : + ;
la soustraction : - ;
la multiplication : * ;
la division : / ;
la puissance : ** ;
le modulo (renvoie le reste de la division entière) : %.

quelques exemples let

```
let « a = 5 * 3 » # $a=15
let " a = a * 3"
```

Actuellement, les résultats renvoyés sont des nombres entiers et non des

nombres décimaux. Si vous voulez travailler avec des nombres décimaux, renseignez-vous sur le fonctionnement de la commande bc.

Les variables d'environnement

Actuellement, les variables que vous créez dans vos scripts bash n'existent que dans ces scripts. En clair, une variable définie dans un programme A ne sera pas utilisable dans un programme B

Les variables d'environnement sont des variables que l'on peut utiliser dans n'importe quel programme. On parle aussi parfois de variables globales

env pour obtenir les variables d'environnement actuelles

– SHELL : indique quel type de shell est en cours d'utilisation (sh, bash, ksh. . .) ;
– PATH : une liste des répertoires qui contiennent des exécutables que vous souhaitez pouvoir lancer sans indiquer leur répertoire. Nous en avons parlé un peu plus tôt. Si un programme se trouve dans un de ces dossiers, vous pourrez l'invoquer quel que soit le dossier dans lequel vous vous trouvez ;
– EDITOR : l'éditeur de texte par défaut qui s'ouvre lorsque cela est nécessaire ;
– HOME : la position de votre dossier home ;
– PWD : le dossier dans lequel vous vous trouvez ;
– OLDPWD : le dossier dans lequel vous vous trouviez auparavant.

Notez que les noms de ces variables sont, par convention, écrits en majuscules

#!/ bin / bash
echo " Votre é diteur par d é faut est $EDITOR "

pour afficher la variable EDITOR

Plus rarement, vous pourriez avoir besoin de définir votre propre variable d'environnement. Pour cela, on utilise la commande export que vous avez pu voir dans votre .bashrc.

Les variables des paramètres

./variables.sh param1 param2 param3 pour appeler notre script avec des parametres
Le problème, c'est que nous n'avons toujours pas vu comment récupérer ces paramètres dans notre script
En effet, des variables sont automatiquement créées :
$# : contient le nombre de paramètres ;
$0 : contient le nom du script exécuté (ici ./variables.sh) ;
$1 : contient le premier paramètre ;
$2 : contient le second paramètre ;
...;
$9 : contient le 9 e paramètre.

#!/ bin / bash
echo " Vous avez lanc é $0 , il y a $ # param è tres "
echo " Le param è tre 1 est $1 "

$ ./ variables . sh param1 param2 param3
Vous avez lanc é ./ variables . sh , il y a 3 param è tres
Le param è tre 1 est param1

#!/ bin / bash
echo " Le param è tre 1 est $1 "
shift

echo " Le param è tre 1 est maintenant $1 "

pour decaler le parametre 1 a 2

$ ./ variables . sh param1 param2 param3
Le param è tre 1 est param1
Le paramètre 1 est maintenant param2
Comme vous le voyez, les paramètres ont été décalés : $1 correspond après le shift au second paramètre, $2 au troisième paramètre, etc.

shift est généralement utilisé dans une boucle qui permet de traiter les paramètres un par un.

On peut mettre autant de ârametre que lon veut
./variables.sh param1 param2 param3 … param15

Les tableaux
tableau =(’ valeur0 ’ ’ valeur1 ’ ’ valeur2 ’) pour definir un tableau
${tableau [2]} pour accerder a la variable 2
Les cases sont numérotées à partir de 0 !
tableau [2]= ’ valeur2 ’ pour definir manuellement la 3 ieme variable du tableau

#!/ bin / bash
tableau =(’ valeur0 ’ ’ valeur1 ’ ’ valeur2 ’)
tableau [5]= ’ valeur5 ’
echo $ { tableau [1]}

le script affichera valeur1

Comme vous pouvez le constater, le tableau peut avoir autant de cases que vous le désirez. La numérotation n’a pas besoin d’être continue, vous pouvez sauter des cases sans aucun problème (la preuve, il n’y a pas de case n° 3 ni de case n° 4 dans mon script précédent).

${tableau[*]} pour afficher toutes les valeurs d’un tableau

#!/ bin / bash
tableau =(’ valeur0 ’ ’ valeur1 ’ ’ valeur2 ’)
tableau [5]= ’ valeur5 ’
\ surligne { echo $ { tableau [*]}}

affichera
valeur0 valeur1 valeur2 valeur5

if : la condition la plus simple
Les conditions s’ecrivent de cette facon

SI test_de_variable
ALORS
------> e f fectuer_une_action
FIN SI

avec la syntaxe

```
if [ test ]
then
        echo "C ’ est vrai "
fi
```

Vous noterez — c’est très important — qu’il y a des espaces à l’intérieur des crochets. On ne doit pas écrire [test] mais [test] !

on peut aussi ecrire

```
if [ test ]; then
        echo "C ’ est vrai "
fi
```

PAR EXEMPLE

```
#!/ bin / bash
nom =" Bruno "
if [ $nom = " Bruno " ]
then
        echo " Salut Bruno !"
fi
```

: si vous n’écrivez pas précisément « Bruno », le if ne sera pas exécuté et votre script n’affichera donc rien.

```
#!/ bin / bash
nom1 =" Bruno "
nom2 =" Marcel "
if [ $nom1 = $nom2 ]
then
        echo " Salut les jumeaux !"
fi
```

Sinon

```
SI test_de_variable
ALORS
------> ef f e c t uer_une_action
SINON
------> ef f e c t uer_une_action
FIN SI
```

```
if [ test ]
then
        echo "C ’ est vrai "
else
        echo "C ’ est faux "
fi
```

```
#!/ bin / bash
nom =" Bruno "
if [ $nom = " Bruno " ]
then
        echo " Salut Bruno !"
else
        echo "J ’ te connais pas , ouste !"
fi
```

```
#!/ bin / bash
if [ $1 = " Bruno " ]
then
        echo " Salut Bruno !"
else
        echo "J ’ te connais pas , ouste !"
fi
```

on teste le script avec un parametre

```
$ ./ conditions . sh Bruno
Salut Bruno !
```

$ ./ conditions . sh Jean
J ’ te connais pas , ouste !

Notez que le script plante si vous oubliez de l’appeler avec un paramètre.
Pour bien faire, il faudrait d’abord vérifier dans un if s’il y a au moins un paramètre. Nous apprendrons à faire cela plus loin.

Sinon si

```
if [ test ]
then
        echo " Le premier test a é t é v é rifi é "
elif [ autre_test ]
then
        echo " Le second test a é t é v é rifi é "
elif [ encore_autre_test ]
then
        echo " Le troisi è me test a é t é v é rifi é "
else
        echo " Aucun des tests pr é c é dents n ’ a é t é v é rifi é "
fi
```

```
#!/ bin / bash
if [ $1 = " Bruno " ]
then
        echo " Salut Bruno !"
elif [ $1 = " Michel " ]
then
        echo " Bien le bonjour Michel "
elif [ $1 = " Jean " ]
then
        echo " H é Jean , ç a va ?"
else
        echo "J ’ te connais pas , ouste !"
fi
```

Vous pouvez tester ce script ; encore une fois, n’oubliez pas d’envoyer un paramètre sinon il plantera, ce qui est normal.

Les tests

Les différents types de tests

Il est possible d’effectuer trois types de tests différents en bash :
– des tests sur des chaînes de caractères ;
– des tests sur des nombres ;
– des tests sur des fichiers.

Tests sur des chaînes de caractères

en bash toutes les variables sont considérées comme des chaînes de caractères
Il est donc très facile de tester ce que vaut une chaîne de caractères

```
#!/ bin / bash
if [ $1 != $2 ]
then
        echo " Les 2 param è tres sont diff é rents !"
else
        echo " Les 2 param è tres sont identiques !"
fi
```

Condition
$chaine1 = $chaine2
$chaine1 != $chaine2

-z $chaine verifie si la chaine est vide
-n $chaine verifie si la chaine est non vide

On peut aussi vérifier si le paramètre existe avec -z.
En effet, si une variable n'est pas
définie, elle est considérée comme vide par bash. On peut donc s'assurer que $1 existe
en faisant comme suit :

```
#!/ bin / bash
if [ -z $1 ]
then
        echo " Pas de param è tre "
else
        echo " Param è tre pr é sent "
fi
```

```
$ ./ conditions.sh
Pas de Paramtères
```

```
$ ./ conditions . sh param
Param è tre pr é sent
```

Tests sur des nombres

Bien que bash gère les variables comme des chaînes de caractères pour son fonctionnement interne, on peut faire des comparaisons de nombres si ces variables en contiennent.

Condition
$num1 -eq $num2 si les nombres sont egaux
$num1 -ne $num2 sils sont!=
$num1 -lt $num2 si num1 < num2
$num1 -le $num2 si num1 inferieur ou égale a num2
$num1 -gt $num2 si num1 > num2
$num1 -ge $num2 si num1 superieur ou egale a num2

```
#!/ bin / bash
if [ $1 - ge 20 ]
then
echo " Vous avez envoy é 20 ou plus "
else
echo " Vous avez envoy é moins de 20"
fi
```

```
$ ./ conditions . sh 23
Vous avez envoy é 20 ou plus
```

```
$ ./ conditions . sh 11
Vous avez envoy é moins de 20
```

Tests sur des fichiers

Un des avantages de bash sur d'autres langages est que l'on peut très facilement faire des tests sur des fichiers : savoir s'ils existent, si on peut écrire dedans, s'ils sont plus vieux, plus récents, etc.

Condition
-e $nomfichier   sil exixste
-d $nomfichier   Vérifie si le fichier est un répertoire. N'oubliez pas
que sous Linux, tout est considéré comme un fichier,même un répertoire !
-f $nomfichier Vérifie si le fichier est un. . . fichier. Un vrai fichier cette fois, pas un dossier
-L $nomfichier Vérifie si le fichier est un lien symbolique (raccourci).
-r $nomfichier Vérifie si le fichier est lisible (r).
-w $nomfichier Vérifie si le fichier est modifiable (w).
-x $nomfichier Vérifie si le fichier est ececutable (x).

$fichier1 -nt)Vérifie si fichier1 est plus récent que fichier2 (newer than).
$fichier2)

$fichier1 -ot) Vérifie si fichier1 est plus vieux que fichier (older than).
$fichier2)

```
#!/ bin / bash
read -p ’ Entrez un r é pertoire : ’ repertoire
if [ -d $repertoire ]
then
echo " Bien , vous avez compris ce que j ’ ai dit !"
else
echo " Vous n ’ avez rien compris ..."
fi
```

Entrez un r é pertoire : / home
Bien , vous avez compris ce que j ’ ai dit !

Entrez un r é pertoire : rienavoir . txt
Vous n ’ avez rien compris …

Notez que bash vérifie au préalable que le répertoire existe bel et bien.

Effectuer plusieurs tests à la fois

Dans un if, il est possible de faire plusieurs tests à la fois. En général, on vérifie :
– si un test est vrai ET qu’un autre test est vrai ;
– si un test est vrai OU qu’un autre test est vrai.
Les deux symboles à connaître sont :
– && : signifie « et » ;
– || : signifie « ou ».

```
#!/ bin / bash
if [ $ # - ge 1 ] && [ $1 = ’ koala ’ ]
then
echo " Bravo !"
echo " Vous connaissez le mot de passe "
else
echo " Vous n ’ avez pas le bon mot de passe "
fi
```

Le test vérifie deux choses :
– qu’il y a au moins un paramètre (« si $# est supérieur ou égal à 1 ») ;
– que le premier paramètre est bien koala (« si $1 est égal à koala »).

Si ces deux conditions sont remplies, alors le message indiquant que l’on a trouvé le bon mot de passe s’affichera.

```
$ ./ conditions . sh koala
Bravo !
Vous connaissez le mot de passe
```

Notez que les tests sont effectués l’un après l’autre et seulement s’ils sont nécessaires. Bash vérifie d’abord s’il y a au moins un paramètre. Si ce n’est pas le cas, il ne fera pas le second test puisque la condition ne sera de toute façon pas vérifiée.

Inverser un test

Il est possible d’inverser un test en utilisant la négation. En bash, celle-ci est exprimée par le point d’exclamation « ! ».

```
if [ ! -e fichier ]
```

```
then
        echo " Le fichier n ’ existe pas "
fi
```

Vous en aurez besoin, donc n’oubliez pas ce petit point d’exclamation.

case : tester plusieurs conditions à la fois

Le rôle de case est de tester la valeur d’une même variable, mais de manière plus concise et lisible.

```
#!/ bin / bash
case $1 in
        " Bruno ")
                echo " Salut Bruno !"
                        ;;
        " Michel ")
                echo " Bien le bonjour Michel "
                ;;
        " Jean ")
                echo " H é Jean , ç a va ?"
                ;;
        *)
                echo "J ’ te connais pas , ouste !"
                ;;
esac
```

Important, il ne faut pas l’oublier : le double point-virgule dit à bash d’arrêter là la lecture du case

*) signifie else

Nous pouvons aussi faire des « ou » dans un case. Dans ce cas, petit piège, il ne faut pas mettre deux || mais un seul ! Exemple :

```
#!/ bin / bash
case $1 in
        " Chien " | " Chat " | " Souris ")
                echo "C ’ est un mammif è re "
                ;;
        " Moineau " | " Pigeon ")
                echo "C ’ est un oiseau "
                ;;
        *)
                echo " Je ne sais pas ce que c ’ est "
                ;;
esac
```

while : boucler « tant que »

```
while [ test ]
do
        echo ’ Action en boucle ’
done
```

on peut aussi ecrire

```
while [ test ]; do
        echo ’ Action en boucle ’
done
```

```
#!/ bin / bash
while [ -z $reponse ] || [ $reponse != ’oui ’ ]
do
        read -p ’ Dites oui : ’ reponse
done
```

On fait deux tests.
1. Est-ce que $reponse est vide ?
2. Est-ce que $reponse est différent de oui ?

Il existe aussi le mot clé until, qui est l'exact inverse de while. Il signifie « Jusqu'à ce que ». Remplacez juste while par until dans le code précédent pour l'essayer.

for : boucler sur une liste de valeurs

Avertissement pour ceux qui ont déjà fait de la programmation : le for en bash ne se comporte pas de la même manière que le for auquel vous êtes habitués dans un autre langage, comme le C ou le PHP. Lisez donc attentivement.

Parcourir une liste de valeurs

La boucle for permet de parcourir une liste de valeurs et de boucler autant de fois qu'il y a de valeurs.

```
#!/ bin / bash
for variable in ' valeur1 ' ' valeur2 ' ' valeur3 '
do
        echo " La variable vaut $variable "
done
```

ce qui donnera La variable vaut valeur1
La variable vaut valeur2
La variable vaut valeur3

```
#!/ bin / bash
for animal in ' chien ' ' souris ' ' moineau '
do
        echo " Animal en cours d ' analyse : $animal "
done
```

la liste na pas besoin detre definie directment dans le code

```
#!/ bin / bash
liste_fichiers = ' ls '
for fichier in $liste_fichiers
do
        echo " Fichier trouvé : $fichier "
done
```

Ce script liste tous les fichiers trouvés dans le répertoire actuel :
Fichier trouv é : boucles . sh
Fichier trouv é : conditions . sh
Fichier trouv é : variables . Sh

NE PAS OUBLIER DE VOIR DANS LE CONTEUR DE CARACT7RE SI les é sont comptés comme 1 ou 2 octets

on pourrait eviter de créer une variable liste_fichiers en faisant

```
#!/ bin / bash
for fichier in 'ls '
do
echo " Fichier trouv é : $fichier "
done
```

Bien entendu, ici, on ne fait qu'afficher le nom du fichier, ce qui n'est ni très amusant ni très utile. On pourrait se servir de notre script pour renommer chacun des fichiers du répertoire actuel en leur ajoutant un suffixe -old par exemple :

```
#!/ bin / bash
for fichier in ‘ls ‘
do
        mv $fichier $fichier-old
done
```

À vous de jouer ! Essayez de créer un script multirenommage.sh, reposant sur ce principe, qui va rajouter le suffixe -old. . . uniquement aux fichiers qui correspondent au paramètre envoyé par l’utilisateur !

./ multirenommage . sh *. txt

Si aucun paramètre n’est envoyé, vous demanderez à l’utilisateur de saisir le nom des fichiers à renommer avec read.

Un for plus classique

Pour les habitués d’autres langages de programmation, le for est une boucle qui permet de faire prendre à une variable une suite de nombres.

En bash, comme on l’a vu, le for permet de parcourir une liste de valeurs. Toutefois, en trichant un peu à l’aide de la commande seq, il est possible de simuler un for classique :

```
#!/ bin / bash
for i in ‘ seq 1 10 ‘;
do
        echo $i
done
```

Explication : seq génère tous les nombres allant du premier paramètre au dernier paramètre, donc 1 2 3 4 5 6 7 8 9 10.

Si vous le voulez, vous pouvez changer le pas et avancer de deux en deux par exemple. Dans ce cas, il faut écrire seq 1 2 10 pour aller de 1 à 10 en avançant de deux en deux ; cela va donc générer les nombres 1 3 5 7 9.

DERNIER CHAPITRE **TP : générateur de galerie d’images**

Authentification par clé depuis Windows (PuTTY en version installée)

mm principe que sur Linux : créer 1 paire de clé sur le pc client et send clé publique au serveur

Générer une paire de clés (publique et privée) avec Puttygen

PuTTYgen se trouve dans le gestionnaire d’installation de PuTTY
une fois PuTTYgen lancé
la valeur par defaut de l’aglorithme de cryptage est RSA 1024bits
on peut le changer mais on peut s’en contenter
on clique sur «Generate»
on peut choisir dentrer un pasprasse ou non
ensuite on enregistre la clé dans un fichier en cliaque sur « Save public key »
on lui donne lextension .ppk donc un nom cle.ppk par exemple
on ne ferme pas encore PuTTYgen

Envoyer la clé publique au serveur

il faut envoyer la clé au serveur
il nya pas de commande sur Windows et il faut donc l’ajouter a la main danns le fichier authorized_keys
pour cela on ouvre PuTTY
on se connecte ensuite au serveur
puis se endre dans le dossier ~/.ssh
si le dossier nexiste pas il faudra le créer
on rajoute a notre clé publique a la fin du fichier authorized_keys

echo "votre_cle" >> authorized_keys il faudra copier la clé qui est dans PuTTYgen
utiliser Shift+insert plutot que Ctrl+v car Ctrl+V fera des mises en pages
et copier **echo " ssh -rsa AAAAB3NzaC1yc2E [...] AAAABJQAP ++ UWBOkLp0 = rsa –key -20081117" >> authorized_keys**

Configurer PuTTY pour qu'il se connecte avec la clé

une fois PuTTY ouvert allller dans **Window** => **Translation**
qui permet de régler les accents qui s'affichent mal dans la console mettre **UTF-8**
la plupart des serveurs encodent désormais en UTF-8
aller dans **Connection** → **SSH** → **Auth puis Browse** pour selectionner la clé privée
aller aussi dans Connection → Data pour entrer le login dans auto-login username
retourner a laccueil en allant dans session Session
entrer lip du serveur et enregistrer les paramatres
donner un nom au serveur sous saved session
faire opent entrer le passphrase

L'agent SSH Pageant

l'agent SSH de PuTTY s'appelle Pageant
conseillé de lancer l'agent SSH au démarrage de lordinateur automatiquement (4Mo de RAM)
une fois pageant lancé trouver licone en bas a droite et faire ADD KEY
on chercher la cle.ppk et ensuite le passphrase
mintenant on peut se onnecter automatiquement en cherchant daans saved sessions
conseiller de le fermer si on laisse lordinateur ouvert car sinon nimp qui peut se co

on peut automatiser le chargement de la clé privée dans licone pageant => propriétés => dans le champ cible on rajoute le parametre chemin de la clé a charger comme par exemple " C:\ProgramFiles\PuTTY\pageant.exe " c:\cle.ppk

{A..Z}

`seq 0 15`

for i in $sequence
do
done

grep -io $i < $1 | wc -l permet de compter le nombre de lettre TOTALE dans le dico
grep -i $i < $1 | wc -l ...

resultat=$(echo "scale=2; ($compte*100)/$total" | bc -l) pour float