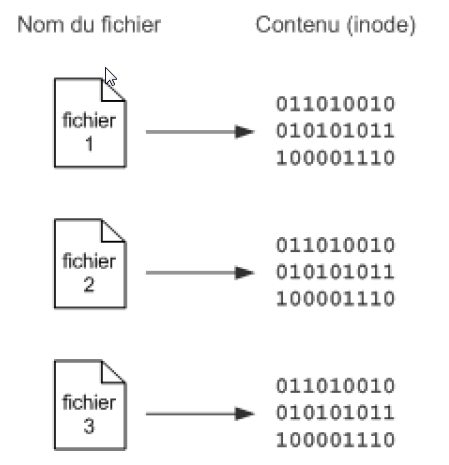
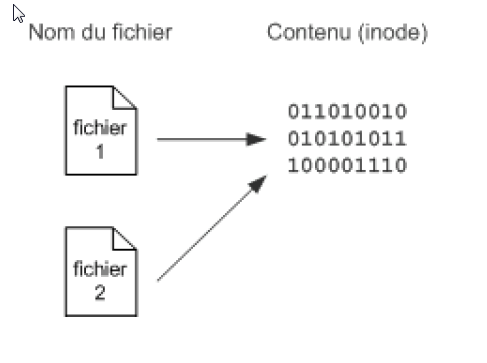
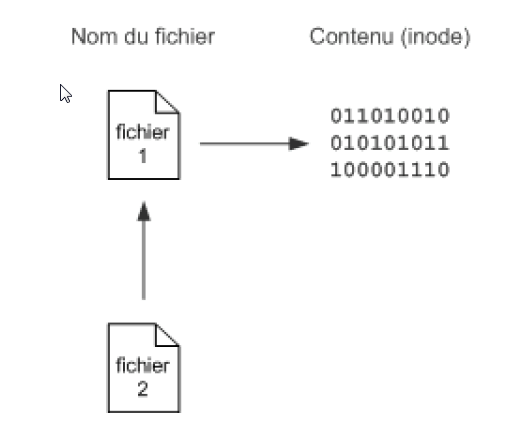
1. pour se connecter en **super-User (root)**, il faut taper : sudo –s( ou sudo su -, ou su), et après taper le mot de passe de ton user.
2. Se connecter à un utilisateur : su -l utilisateur
3. Le proxy se trouve dans le fichier /etc/apt/ apt.conf.d/proxy
4. Version linux :
   1. uname –a
   2. cat /etc/issue
5. **Commande**
   1. pwd : afficher le dossier actuel
   2. which : connaître l'emplacement d'une commande. Exemple : which pwd
   3. ls : lister les fichiers et dossiers
   4. ls –a : : afficher tous les fichiers et dossiers cachés
   5. ls -F : indique le type d'élément. Ajoute un symbole à l’élément pour qu’on puisse faire une distinction entre dossier, fichier et raccourci.
   6. Ls –l : afficher les détails.
   7. ls –lh : affiche la taille de chaque fichier et du total.
   8. ls –lt : trier par la date de dernière modification.
   9. du : taille occupée par les dossiers (et pas les fichiers, du -a : afficher la taille des dossiers ET des fichiers).
   10. du -h : la taille pour les humains.
   11. du –s : avoir juste la taille totale.
6. **Manipuler les fichiers**
   1. cat, less, more : deux commandes basiques permettant d’afficher le contenu d’un fichier.
   2. cat : affiche le fichier entier. Intéressantes pour afficher les fichiers de petites tailles. Le paramètre –n permet d’afficher le numéro de ligne (cat –n)
   3. less : affiche progressivement le contenu du fichier :
      1. espace : défile d’un écran vers le bas pour afficher la suite du fichier,
      2. entrée : affiche la ligne suivante,
      3. d : affiche les onze lignes suivantes (moitié d’un écran),
      4. b : retourne en arrière d’un écran (on peut utiliser aussi Up)
      5. y : retourne une ligne vers l’arrière
      6. u : retourne en arrière d’une moitié d’écran
      7. q : arrête la lecture du fichier
      8. = : indique où vous en êtes dans le fichier (numéro des lignes affichées et pourcentage).
      9. h : affiche l'aide (toutes les commandes que je vous apprends ici, je les tire de là).
      10. Tapez q pour sortir de l'aide.
      11. / : tapez / suivi du texte que vous recherchez pour lancer le mode recherche.
      12. n : après avoir fait une recherche avec /, la touche n vous permet d'aller à la prochaine occurrence de votre recherche. C'est un peu comme si vous cliquiez sur le bouton Résultat suivant.
      13. N : pareil que n, mais pour revenir en arrière.
   4. more : ressemble à less, elle est plus vieille.
   5. head : afficher le début d’un fichier. head –n nom\_fichier : permet d’afficher n premières lignes.
   6. tail : afficher la fin d’un fichier. tail –f nom\_fichier : permet d’afficher la fin du fichier en temps réel.
   7. touch : modifier la date de modification d’un fichier. si on tape touch nom\_fichier et le fichier n’existe pas, il sera crée.
   8. mkdir : créer un dossier. –p sert à créer tout les dossiers intermédiaires. Exemple : mkdir -p animaux/vertebres/chat.
   9. cp : copier un fichier. cp cheminSource cheminDestination
   10. mv : déplacer un fichier.
   11. rm : supprimer un fichier. –i : demander confirmation. –f : forcer la suppression. –r : détruit le dossier et son contenu
   12. ln : créer des liens entre fichiers. Sous linux, il existe deux types de liens : lien physique et lien symbolique. Pour comprendre la différence entre ces deux, il faudra d’abord comprendre comment linux stocke ces fichiers.
       1. Sur le disque dur, chaque chier est grosso-modo séparé en deux parties : son nom ; son contenu.
       2. Chaque contenu de fichier se voit attribuer un numéro d'identification appelé inode. Chaque nom de fichier est donc associé à un inode (son contenu).



* + 1. Créer des liens physiques : Ce type de lien est plus rarement utilisé que le lien symbolique, mais il faut tout de même le connaître car il peut se révéler pratique. Ainsi, que vous passiez par fichier1 ou par fichier2, vous modifiez exactement le même contenu. En quelque sorte, le fichier est le même. On peut juste y accéder via deux noms de fichiers différents. La création se fait par : ln fichier1 fichier2. ls –i : permet d’afficher les numéro d’inodes et savoir si des fichiers sont associés au même inode. l'autre fichier reste en place et le contenu sera toujours présent sur le disque. L'inode est supprimé uniquement quand plus aucun nom de fichier ne pointe dessus.



* + 1. Créer des liens symboliques : Les liens symboliques ressemblent plus aux raccourcis dont vous avez peut-être l'habitude sous Windows. Le principe du lien symbolique est que l'on crée un lien vers un autre nom de fichier. Cette fois, on pointe vers le nom de fichier et non vers l'inode directement. ln -s fichier1 fichier2 (s comme symbolique).

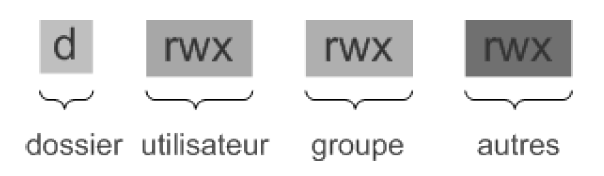


1. **Les utilisateurs et les droits**
   1. adduser : ajouter un utilisateur.
   2. passwd : changer le mot de passe.
   3. deluser : supprimer un compte. Pour supprimer aussi le home du compte patrick, faire : deluser --remove-home patrick
   4. à part ubuntu, il faut utiliser : Partout ailleurs on doit utiliser useradd et userdel
   5. pour voir les comptes existant, se placer dans /home et taper : ls –l
   6. usermod : modifier un utilisateur. Possède plusieurs paramètres, nous allons en retenir deux :
      1. –l : renomme l’utilisateur (le nom de son répertoire ne sera pas changé)
      2. –g : change de groupe. usermod -g groupe utilisateur
      3. Il est aussi possible de faire en sorte qu'un utilisateur appartienne à plusieurs groupes. Pour ce faire, utilisez le paramètre -G (majuscule). Exemple : usermod -G amis,paris,collegues patrick. Séparez les noms des groupes par une virgule, sans espace entre chaque nom de groupe.
      4. Faites très attention en utilisant usermod ! Lorsque vous avez recours à -G,

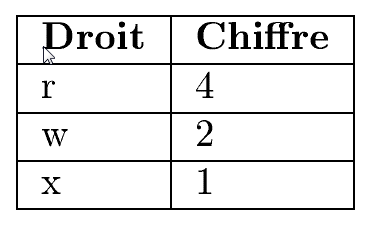
l'utilisateur change de groupe et ce peu importe les groupes auxquels il appartenait auparavant. Si vous voulez ajouter des groupes à un utilisateur (sans perdre les groupes auxquels il appartenait avant cela), utilisez -a :

usermod -aG amis patrick

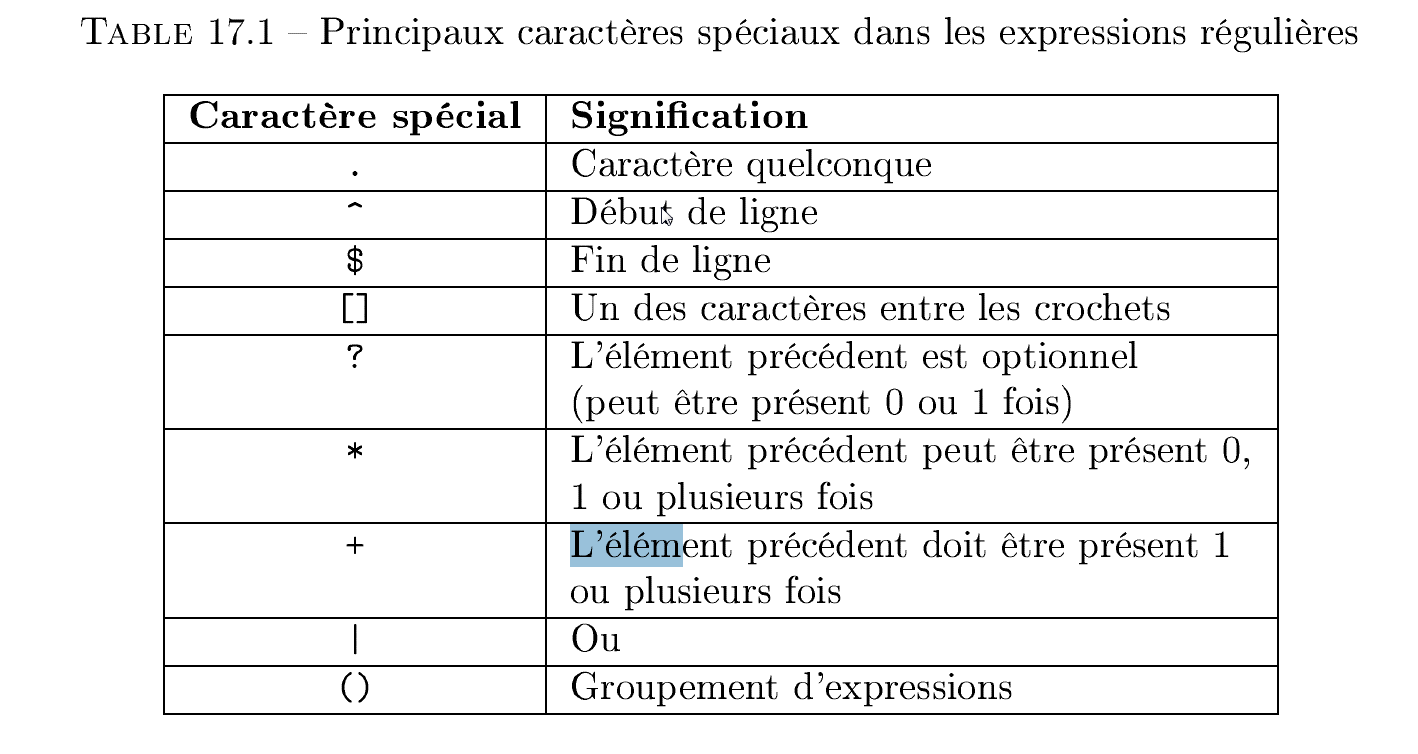
* 1. groups nom\_utilisateur : afficher les groupes auxquels l’utilisateur appartient
  2. adduser nom\_user sudo : assigner les droits administrateur à un utilisateur (le rajouter au groupe sudo).
  3. addgroup : créer un groupe
  4. delgroup : supprimer un groupe. Les commandes traditionnelles qui fonctionnent partout sont groupadd et groupdel
  5. chown : changer le propriétaire d'un fichier. La commande chown, qui doit être utilisée en tant que root, attend deux paramètres au moins :
     1. le nom du nouveau propriétaire ;
     2. le nom du fichier à modifier.
  6. chown peut aussi changer le groupe propriétaire d'un fichier ! exemple : chown patrick:amis rapport.txt
  7. -R : affecter récursivement les sous-dossiers. Exemple : chown -R mateo21:mateo21 /home/patrick/
  8. chgrp : changer le groupe propriétaire d'un fichier. chgrp s'utilise exactement de la même manière que chown à la différence près qu'il affecte cette fois le groupe propriétaire d'un fichier.
  9. chmod : modifier les droits d'accès
  10. Résumé droits : Vous voyez tous ces d, r, w et x au début ? Ce sont ce qu'on appelle les droits d'accès du fichier ou dossier. On peut voir cinq lettres différentes. Voici leur signification :
      1. d (Directory) : indique si l'élément est un dossier ;
      2. l (Link) : indique si l'élément est un lien (raccourci) ;
      3. r (Read) : indique si on peut lire l'élément ;
      4. w (Write) : indique si on peut modifier l'élément ;
      5. x (eXecute) : si c'est un fichier, indique qu'on peut l'exécuter. Ce n'est utile que pour les fichiers exécutables (programmes et scripts). Si c'est un dossier, indique qu'on peut le traverser, c'est-à-dire qu'on peut voir les sous-dossiers qu'il contient si on a le droit de lecture dessus.
      6. Si la lettre apparaît, c'est que le droit existe. S'il y a un tiret à la place, c'est qu'il n'y a aucun droit.
      7. Les droits sont découpés en fonction des utilisateurs



* + 1. Attribuer des droits avec des chiffres



1. **Installer des programmes avec apt-get**
   1. apt-get update met à jour la liste des programmes (appelés paquets) qui existent.
   2. apt-cache search permet de rechercher dans la liste des paquets.
   3. apt-get install télécharge et installe un paquet.
   4. apt-get upgrade met à jour tous les paquets installés.
   5. apt-get autoremove permet de supprimer un paquet.
2. **Commandes aide**
   1. man nom\_commande : affiche de l’aide sur la commande
   2. apropos concept : permet de retrouver les commandes qui parle de concept(concept est à remplacer par ce qu’on cherche). Exemple apropos sound permet de retrouver les commandes qui ont un lien avec le son.
   3. whatis commande : man très allégé.
3. **Rechercher des fichiers** 
   1. locate nom\_fichier : une recherche rapide. Le défaut de locate est qu’elle fait la recherche sur la base de données de fichiers. Si vous venez de créer un fichier (pas encore mis à jour dans la base de données), locate ne l’affichera pas. On peut forcer la mise à jour de la base de données avec updatedb
   2. find : une recherche approfondie.
      1. Les paramètres de find : find ‘où’ ‘quoi’ ‘que faire avec’
      2. où : le répertoire où on doit chercher. Si il n’est pas précisé, on cherche dans le dossier courant.
      3. Quoi : On peut rechercher des fichiers en fonction de leur nom (-name), de leur taille (-size), de leur date de dernier accès (-atime). . .
      4. Que faire avec : on peut effectuer plusieurs actions. La plus simple est d’afficher le résultat.
      5. Avec -exec, vous pouvez appeler une commande qui exécutera une action sur chacun des fichiers trouvés. Exemple : find ~ -name "\*.jpg" -exec chmod 600 {} \; La commande n'affiche rien s'il n'y a pas eu d'erreur. Pour chaque fichier .jpg trouvé, on exécute la commande qui suit -exec : cette commande ne doit PAS être entre guillemets ;les accolades {} seront remplacées par le nom du fichier ; la commande doit finir par un \; obligatoirement.
4. **Extraire, trier et ltrer des données** grep : rechercher un mot dans un fichier et d'afficher les lignes dans lesquelles ce mot a été trouvé.
   1. grep simple : grep text nomFichier. Exemple : grep SELECT Materiau\_Type.sql. affiche uniquement les lignes qui contiennent text.
   2. -i : ne pas tenir compte de la casse (majuscules / minuscules)
   3. -n : connaître les numéros des lignes
   4. -v : inverser la recherche : Si, au contraire, vous voulez connaître toutes les lignes qui ne contiennent pas un mot donné, utilisez –v.
   5. -r : rechercher dans tous les fichiers et sous-dossiers. grep –r text nomRepertoire. rgrep : est equivalent à : grep –r
   6. –E : utiliser avec les expressions régulières. egrep est équivalent à grep -Ee



* 1. sort : trier les lignes.
     1. Sort –o : écrire le résultat dans fichier. sort -o noms\_tries.txt noms.txt, écrit le résultat dans noms\_tries.txt.
     2. -r : trier en ordre inverse
     3. -R : trier aléatoirement
     4. -n : trier des nombres
  2. wc : compter le nombre de lignes. wc nom\_fichier, affiche trois numéro : nombre de lignes, nombre de mots, nombre d’octets
  3. uniq supprime les doublons d'un fichier
  4. cut : couper une partie du fichier

1. **Redirection de flux**
   1. > : rediriger dans un nouveau fichier
   2. >> : rediriger à la fin d'un fichier
   3. 2> : Rediriger les erreurs dans un fichier à part. Exemple : cut -d , -f 1 fichier\_inexistant.csv > eleves.txt 2> erreurs.log.
   4. 2>&1 : Fusionner les sorties (mettre tout dans le meme fichier, résultat de la commande et les erreurs). Exemple : cut -d, -f 1 fichier\_inexistant.csv > eleves.txt 2>&1
   5. | : chaîner les commandes. En gros, tout ce qui sort de la commande1 est immédiatement envoyé à la commande2. Et vous pouvez chaîner des commandes comme cela indéfiniment !
2. **Surveiller l'activité du système**
   1. w : qui fait quoi ?
   2. ps : liste statique des processus. ps vous permet d'obtenir la liste des processus qui tournent au moment où vous lancez la commande. Cette liste n'est pas actualisée en temps réel
      1. ps -ef : lister tous les processus
      2. ps -ejH : afficher les processus en arbre
      3. ps -u UTILISATEUR : lister les processus lancés par un utilisateur
   3. top : liste dynamique des processus.
      1. q : ferme top ;
      2. h : affiche l'aide, et donc la liste des touches utilisables.
   4. Ctrl + C : arrêter un processus lancé en console
   5. kill pid\_du\_process: tuer un processus.
      1. Kill -9 pid : tuer le processus sans lui laisser le temps de s’arrêter proprement
      2. Killall : arrête plusieurs processus au même temps.
   6. halt & reboot : arrêter et redémarrer l'ordinateur
3. **EXÉCUTER DES PROGRAMMES EN ARRIÈRE-PLAN**
   1. & : lancer un processus en arrière-plan. Exemple : cp video.avi copie\_video.avi &
   2. nohup : détacher le processus de la console. L'option &, bien qu'assez couramment utilisée, a ce défaut non négligeable : le processus reste attaché à la console, ce qui veut dire que si la console est fermée ou que l'utilisateur se déconnecte, le processus sera automatiquement arrêté. La sortie de la commande est par défaut redirigée vers un fichier nohup.out.
   3. Ctrl + Z : mettre en pause l'exécution du programme
   4. bg : passer le processus en arrière-plan (background)
   5. jobs : connaître les processus qui tournent en arrière-plan
   6. fg : reprendre un processus au premier plan (foreground)
4. **Exécuter un programme à une heure différée**
   1. date : régler l'heure.
      1. date "+%H:%M:%S" (Résultat : 12:36:15) : %H : heure, %M : minute, %S : seconde. À la place des deux points, on peut mettre n’importe quoi. Exemple : date "+%Hh%Mm%Ss"(Résultat : 12h41m01s)
      2. Modifier la date : il faut préciser les informations sous la forme suivante : MMDDhhmmYYYY.
         1. MM : mois ;
         2. DD : jour ;
         3. hh : heure ;
         4. mm : minutes ;
         5. YYYY : année.
         6. Notez qu'il n'est pas obligatoire de préciser l'année. On peut donc écrire : date 11101250
   2. at : exécuter une commande plus tard. Avec at, le programme ne sera exécuté qu'une seule fois. La commande s'utilise en deux temps.
      1. Vous indiquez à quel moment (quelle heure, quel jour) vous désirez que la commande soit exécutée.
      2. Vous tapez ensuite la commande que vous voulez voir exécutée à l'heure que vous venez d'indiquer.
      3. at 14:17
      4. touch fichier.txt
   3. atq et atrm : lister et supprimer les jobs en attente
   4. crontab : exécuter une commande régulièrement : Il y a trois paramètres différents à connaître, pas plus :
      1. -e : modifier la crontab ;
      2. -l : afficher la crontab actuelle ;
      3. -r : supprimer votre crontab. Attention, la suppression est immédiate et sans confirmation
      4. Pour éxecuter une commande régulièrement :
         1. Ouvrir crontab (c’est fichier qui s’ouvre) : crontab –e
         2. On trouve une ligne commentée : # m h dom mon dow command
         3. m : minutes (0 - 59) ;
         4. h : heures (0 - 23) ;
         5. dom (day of month) : jour du mois (1 - 31) ;
         6. mon (month) : mois (1 - 12) ;
         7. dow (day of week) : jour de la semaine (0 - 6, 0 étant le dimanche) ;
         8. command : c'est la commande à exécuter.
         9. Imaginons que je veuille exécuter une commande tous les jours à 15 h 47. Je vais écrire : 47 15 \* \* \* touch /home/mateo21/fichier.txt
         10. \* : exécuté tout le temps (toutes les valeurs sont bonnes) ;
         11. 3,5,10 : exécuté lorsque le champ prend la valeur 3, 5 ou 10. Ne pas mettre d'espace
         12. après la virgule ;
         13. 3-7 : exécuté pour les valeurs 3 à 7 ;
         14. \*/3 : exécuté tous les multiples de 3 (par exemple à 0 h, 3 h, 6 h, 9 h. . .).
   5. cron : Les répertoires de la commande cron sont répartis selon plusieurs planifications :
      1. /etc/cron.hourly = exécution de la tâche toutes les heures
      2. /etc/cron.daily = exécution de la tâche tous les jours
      3. /etc/cron.weekly = exécution de la tâche toutes les semaines
      4. /etc/cron.monthly = exécution de la tâche tous les mois
   6. Vous pouvez également utiliser des planifications prédéfinies pour cron :
      1. @reboot = La commande sera exécutée une fois au démarrage de la machine.
      2. @hourly = Commande exécutée toutes heures
      3. @daily = Commande exécutée tous les jours à minuit
      4. @weekly = Commande exécutée toutes les semaines
      5. @monthly = Commande exécutée tous les mois
      6. @yearly = Commande exécutée tous les ans
      7. Pour les utiliser, c’est simple : @reboot commande
5. **Archiver et compresser**
   1. tar : assembler des fichiers dans une archive
      1. tar -cvf : créer une archive tar. Exemple : tar -cvf nom\_archive.tar nom\_dossier/
         1. -c : signifie créer une archive tar ;
         2. -v : signifie afficher le détail des opérations ;
         3. -f : signifie assembler l'archive dans un fichier.
      2. tar -cvf archive.tar fichier1 fichier2 fichier3. Assembler des fichiers et non pas un dossier.
      3. tar -tf : afficher le contenu de l'archive sans l'extraire
      4. tar -rvf : ajouter un fichier
      5. tar -xvf : extraire les fichiers de l'archive
   2. gzip & bzip2 : compresser une archive
      1. gzip : c'est le plus connu et le plus utilisé ; Exemple : gzip tutoriels.tar
      2. Décompresser gzip : gunzip tutoriels.tar.gz
      3. bzip2 : il est un peu moins fréquemment utilisé. Il compresse mieux mais plus lentement que gzip. Exemple : bzip2 tutorials.tar
      4. Décompresser bunzip2 : bunzip2 tutoriels.tar.bz2
      5. .tar.gz : si l'archive a été compressée avec gzip ;
      6. .tar.bz2 : si l'archive a été compressée avec bzip2.
      7. tar -zcvf : archiver et compresser en gzip. tar -zxvf tutoriels.tar.gz
      8. -jcvf : archiver et compresser en bzip2. tar –jcvf tutoriels.tar.bz2
      9. Pour décompresser, c'est pareil, sauf que le -c est remplacé par un –x
      10. Parfois, on compresse non pas une archive tar mais directement un fichier. Exemple : gzip heritage.tuto. si nous voulions afficher le contenu de ce chier sans le décompresser auparavant. Eh bien il existe des outils qui permettent de faire cela !
          1. zcat : équivalent de cat, capable de lire un fichier compressé (gzippé).
          2. zmore : équivalent de more, capable de lire un fichier compressé (gzippé).
          3. zless : équivalent de less, capable de lire un fichier compressé (gzippé).
      11. unzip & unrar : décompresser les .zip et .rar
      12. unzip –l : voir le contenu sans l’extraire
6. **La connexion sécurisée à distance avec SSH**
   1. Transformer sa machine en serveur pour y accéder à distance : sudo apt-get install openssh-server
   2. /etc/init.d/ssh start
   3. /etc/init.d/ssh stop
7. **Transférer des fichiers**
   1. Ftp
      1. Connexion à un serveur FTP : ftp adresse\_de \_la\_machine
      2. ls : affiche le contenu du répertoire actuel ;
      3. pwd : affiche le chemin du répertoire actuel ;
      4. cd : change de répertoire.
      5. put distant-file local-file : envoie un fichier vers le serveur ;
      6. get distant-file local-file: télécharge un fichier depuis le serveur.
   2. sftp : un FTP sécurisé
   3. rsync permet de synchroniser le contenu de deux dossiers sur un même ordinateur

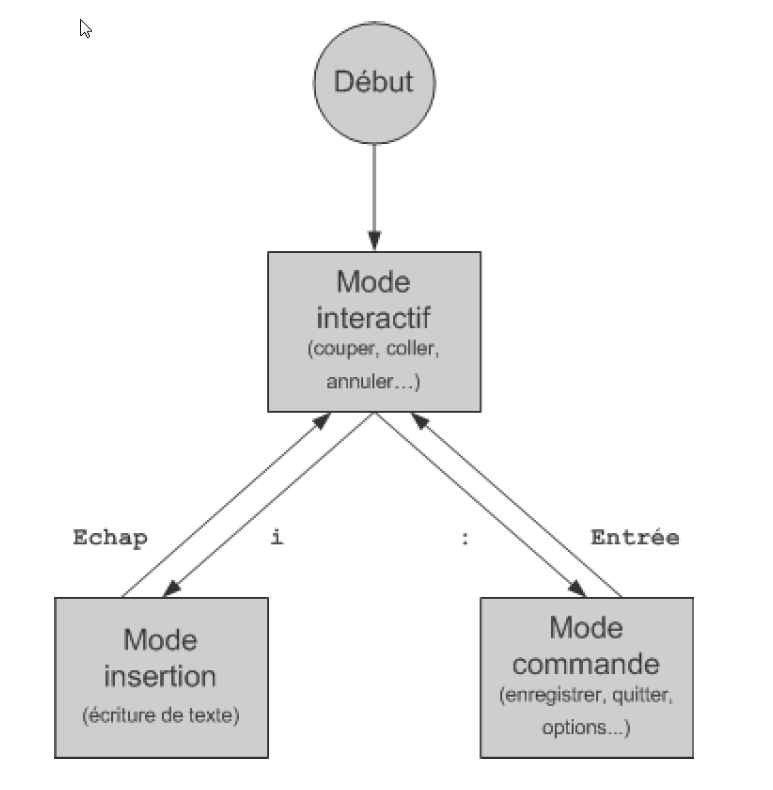
ou sur deux ordinateurs différents. Il est particulièrement utile pour effectuer des

Sauvegardes

1. **Réseaux**
   1. /etc/hosts : permet de lier un nom de domaine à une adresse ip. Si on veut lier une adresse ip à un nom de domaine, il suffit de rajouter la ligne dans ce fichier. exemple : 92.243.25.239 siteduzero.com
   2. Iptables : parefeu. L’ordre des règles dans le parefeu est important
      1. iptables -L : afficher les règles. iptables -L --line-numbers : affiche les numéros des règles
      2. iptables -F : Attention ! Réinitialise toutes les règles iptables !
      3. -A chain : ajoute une règle en fin de liste pour la chain indiquée (INPUT ou OUTPUT, par exemple).
         1. iptables -A (chain) -p (protocole) --dport (port) -j (décision)
         2. iptables -A INPUT -p tcp --dport ssh -j ACCEPT
      4. -D chain rulenum : supprime la règle no rulenum pour la chain indiquée.
      5. -I chain rulenum : insère une règle au milieu de la liste à la position indiquée par rulenum. Si vous n'indiquez pas de position rulenum, la règle sera insérée en premier, tout en haut dans la liste.
      6. -R chain rulenum : remplace la règle no rulenum dans la chain indiquée.
      7. -L : liste les règles (nous l'avons déjà vu).
      8. -F chain : vide toutes les règles de la chain indiquée. Cela revient à supprimer toutes les règles une par une pour cette chain.
      9. -P chain regle : modifie la règle par défaut pour la chain. Cela permet de dire, par exemple, que par défaut tous les ports sont fermés, sauf ceux que l'on a indiqués dans les règles.
      10. iptables -P INPUT DROP : refuse toute les connexions par défaut
      11. 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.
2. **Compiler un programme depuis les sources**
   1. Pour compiler des programmes, vous aurez besoin avant toute chose d'installer les outils de compilation. Pour cela, rien de plus simple, il su‑t d'installer le paquet build-essential : apt-get install build-essential
   2. télécharger les sources du programme sur le web (souvent archivées au format
   3. .tar.gz) ;
   4. décompresser l'archive (tar zxvf archive.tar.gz) ;
   5. exécuter ./configure et résoudre les problèmes ;
   6. exécuter make pour compiler ;
   7. exécuter sudo make install pour installer le programme.
3. **Vim :** possède 3 modes
   1. Mode interactif : c'est le mode par défaut par lequel vous commencez. En lançant

Vim, vous êtes donc en mode interactif. Dans ce mode, vous ne pouvez pas écrire de texte. Le mode interactif est un mode puissant qui permet de se déplacer dans le texte, de supprimer une ligne, copier-coller du texte, rejoindre une ligne précise, annuler ses actions, etc. Chaque action peut être déclenchée en appuyant sur une touche du clavier (par exemple, on appuie sur u pour annuler la dernière action).

* + 1. w : se déplacer de mot en mot
    2. x : effacer des lettres. Par exemple, si vous tapez 4x (4 puis x), vous supprimerez les quatre prochaines lettres en partant du curseur.
    3. dd : supprimer une ligne
    4. dw : supprimer un mot. Vous pouvez aussi supprimer les trois prochains mots en tapant 3dw. Notez que le 3 peut être placé entre le d et le w ; cela revient au même : d3w (qui peut se lire delete 3 words ).
    5. En tapant d0, vous supprimez du curseur jusqu'au début de la ligne.
    6. En tapant d$, vous supprimez du curseur jusqu'à la fin de la ligne.
    7. yy : copier une ligne en mémoire
    8. p : coller
    9. r : remplacer une lettre
    10. u : annuler les modifications
    11. G : sauter à la ligne numéro X. Vous pouvez par exemple directement sauter à la ligne no 7 en tapant 7G (attention, c'est un G majuscule, donc pensez à laisser la touche Maj appuyée).
    12. / : rechercher un mot. Pour passer à la prochaine occurrence du mot, plus bas dans le fichier (s'il apparaît plusieurs fois), appuyez sur n. Pour rechercher en arrière, appuyez sur N (Maj + n).
  1. Mode insertion : celui-là, c'est celui que vous connaissez ! Vous tapez du texte et ce dernier s'insère à l'endroit où se trouve le curseur. Pour entrer dans ce mode, il existe plusieurs possibilités. L'une des plus courantes est d'appuyer sur la touche i (insertion). Pour en sortir, il faut appuyer sur la touche Echap.
     1. :w : enregistrer le fichier
     2. :q : quitter
     3. :q! : forcer la fermeture sans enregistrer
     4. :wq : enregistrer puis quitter
     5. :s : rechercher et remplacer du texte.
        1. :s/ancien/nouveau : remplace la première occurrence de la ligne où se trouve le curseur ;
        2. :s/ancien/nouveau/g : remplace toutes les occurrences de la ligne où se trouve le curseur ;
        3. :#,#s/ancien/nouveau/g : remplace toutes les occurrences dans les lignes no # à # du fichier ;
        4. :%s/ancien/nouveau/g : remplace toutes les occurrences dans tout le fichier. C'est peut-être ce que vous utiliserez le plus fréquemment.
     6. :r : fusion de fichiers. Avec :r, vous pouvez insérer un fichier à la position du curseur. Vous devez indiquer le nom du chier à insérer, par exemple : :r autrefichier. L'autocomplétion avec Tab fonctionne là aussi, donc pas besoin d'écrire le nom du chier en entier !
  2. Mode commande : ce mode permet de lancer des commandes telles que : quitter, enregistrer, etc. Vous pouvez aussi l'utiliser pour activer des options de Vim (comme la coloration syntaxique, l'affichage du numéro des lignes. . .). Vous pouvez même envoyer des commandes au shell (la console) telles que ls, locate, cp, etc. Pour activer ce mode, vous devez être en mode interactif et appuyer sur la touche deux points. Vous validerez la commande avec la touche Entrée et reviendrez alors au mode interactif.

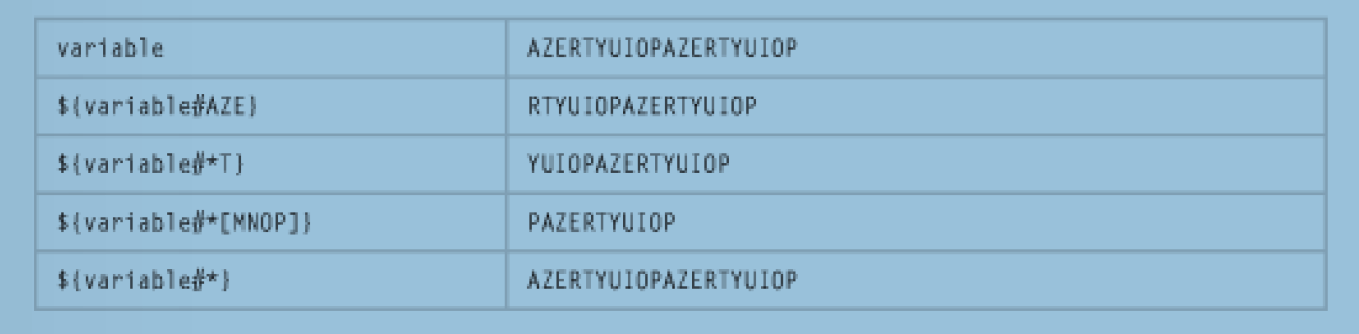


* 1. Activer les options
     1. On peut les activer par set option, elle seront perdu au prochain démarrage de la machine
     2. Mettre les options dans un fichier de configuration dans le dossier personnel. Commencer par copier un fichier template :

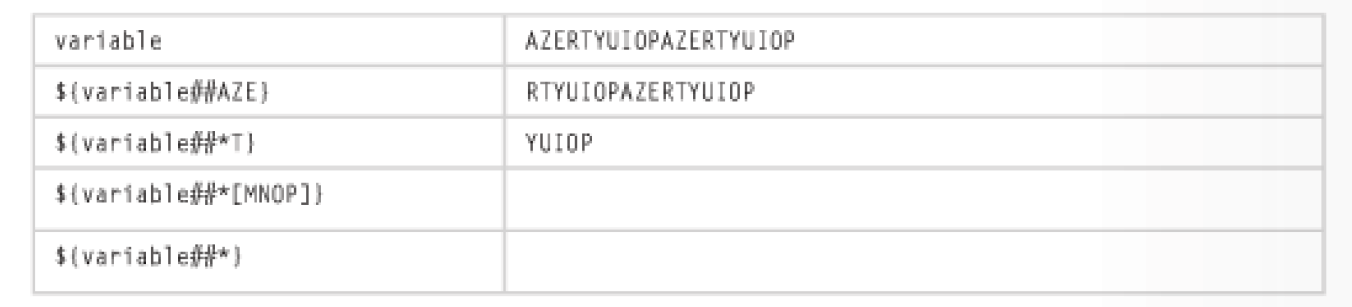
cp /etc/vim/vimrc ~/.vimrc

* + 1. syntax : activer la coloration syntaxique : syntax on

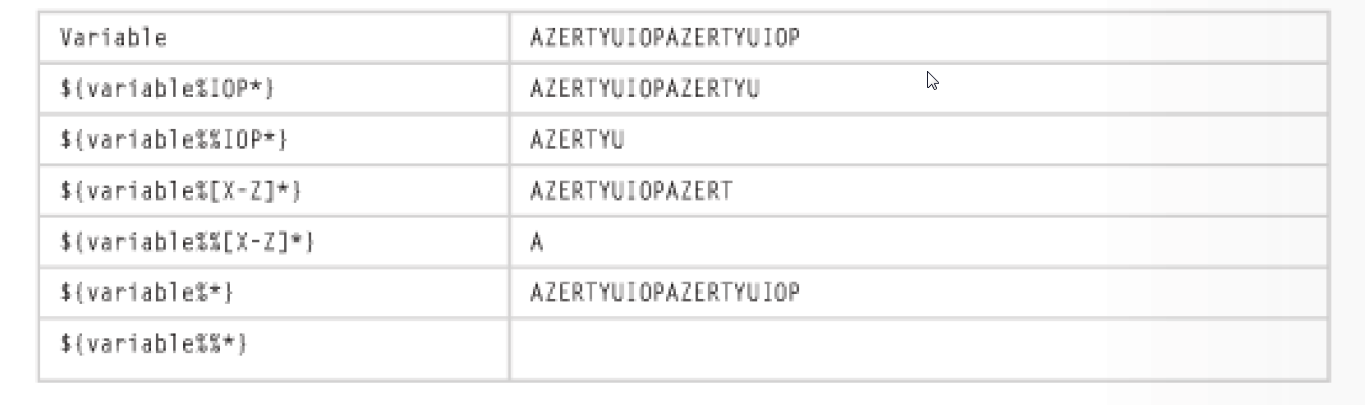
1. **Introduction aux scripts shell**
   1. Au début de chaque shell, mettre cette ligne : #!/bin/bash. /bin/bash peut être remplacé par /bin/sh si vous souhaitez coder pour sh, /bin/ksh pour ksh, etc. Bien que non indispensable, cette ligne permet de s'assurer que le script est bien exécuté avec le bon shell. 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 !
   2. # : commenter une ligne
   3. –x : lancer le script en débogage. Exemple : bash -x essai.sh
   4. Export nom\_variable : permet de mettre nom\_variable dans tout les environnement (deviant accessible d’ailleurs)
   5. ${#variable} : désigne la longueur de la variable
   6. ${variable} : est une généralisation de $variable au cas où on veut l’utiliser avec autre chose.
   7. Toute variable possède un nom et une valeur : message='Bonjour tout le monde'. Il ne faut pas mettre d’espace devant =
   8. Sous-chaine : ${chaine :debut :fin}. Exemple ${message :2 :5}
   9. Si vous voulez insérer une apostrophe dans la valeur de la variable, il faut
   10. la faire précéder d'un antislash \. Exemple : message='Bonjour c\'est moi'
   11. La commande "shift" : La commande interne "shift" permet quant à elle de décaler les paramètres. La valeur du 1er paramètre ($1) est remplacée par la valeur du 2nd paramètre ($2), celle du 2nd paramètre ($2) par celle du 3ème paramètre ($3), etc... On peut indiquer en argument (shift n) le nombre de pas (position) dont il faut décaler les paramètres.
   12. ${variable#motif} : correspond à la valeur de la variable de laquelle on ôte le préfixe ‘motif’ plus petite.



* 1. ${variable##motif} : sert à éliminer le plus long préfixe ‘motif’.



* 1. ${variable%motif} : correspond à la valeur de la variable de laquelle on ôte le suffixe ‘motif’ le plus petit.
  2. ${variable%%motif} : sert à éliminer le plus long suffixe ‘motif’.



* 1. ${variable/motif/remplacement} : remplacer la première occurrence du motif par remplacement.
  2. ${variable/motif/remplacement} : remplacer toutes les occurrences du motif par remplacement.
  3. ${variable :=contenu} : si la variable est vide ou n’existe pas, elle reçoit contenu comme valeur.
  4. echo : afficher une variable. echo $message
     1. Les simples quotes ' '. Avec de simples quotes, la variable n'est pas analysée et le $ est a‑ché tel quel. message='Bonjour tout le monde', echo 'Le message est : $message', le message affiché est : Le message est : $message
     2. Les doubles quotes " " : les doubles quotes demandent à bash d'analyser le contenu du message. S'il trouve des symboles spéciaux (comme des variables), il les interprète. echo "Le message est : $message". Le message est : Bonjour tout le monde.
     3. Les back quotes ' ' : Un peu particulières, les back quotes demandent à bash d'exécuter ce qui se trouve à l'intérieur. message='pwd', echo "Vous êtes dans le dossier $message". Le message affiché est : Vous êtes dans le dossier /home/mateo21/bin
  5. read : demander une saisie. read nomvariable
     1. Affecter simultanément une valeur à plusieurs variables. Read nom prenom
     2. -p : afficher un message de prompt. read -p 'Entrez votre nom : ' nom
     3. -n : limiter le nombre de caractères. read -p 'Entrez votre login (5 caractères max) : ' -n 5 nom
     4. -t : limiter le temps autorisé pour saisir un message
     5. -s : ne pas afficher le texte saisi
  6. Effectuer des opérations mathématiques. Avec la commande let. let "a = 5"

, let "b = 2", let "c = a + b", echo $c

* 1. Les calculs arithmitiques peuvent aussi se faire avec l’opérateur $((operations))
  2. La base d’un nombre par défaut est 10. Si on veut ecrire un nombre avec une autre base : variable=base#nombre. Exemple : masque=2#000110
  3. Les variables d'environnement : Les variables d'environnement sont des variables que l'on peut utiliser dans n'importe quel programme. On parle aussi parfois de variables globales. Pour afficher ces variables, il suffit de taper env.
     1. On appelle les variable d’environnement dans un shell avec $nom
  4. Les variables des paramètres
     1. Comme toutes les commandes, vos scripts bash peuvent eux aussi accepter des paramètres. Ainsi, on pourrait appeler notre script comme ceci : ./variables.sh param1 param2 param3. Le problème, c'est que nous n'avons toujours pas vu comment récupérer ces paramètres dans notre script. Pourtant, c'est très simple à réaliser !

En effet, des variables sont automatiquement créées :

$# : contient le nombre de paramètres ;

$\* contient la liste des paramètres passés en argument,

$\* $@ liste de tous les paramètres La signification de $\* et de $@ est identique. Cependant, lorsque "$\*" est utilisé comme argument de commande, il est équivalent

à "$1 $2 " alors que "$@" est équivalent à "$1" "$2"

$0 : contient le nom du script exécuté (ici ./variables.sh) ;

$1 : contient le premier paramètre ;

$2 : contient le second paramètre ;

. . . ;

$9 : contient le 9e paramètre.

* 1. Pour définir un tableau : tableau=('valeur0' 'valeur1' 'valeur2')
     1. Afficher une case : ${tableau[2]}
     2. Vous pouvez afficher l'ensemble du contenu du tableau d'un seul coup en utilisant ${tableau[\*]}
  2. Les conditions

if [ $1 = "Bruno" ]

then

echo "Salut Bruno !"

elif [ $1 = "Michel" ]

then

echo "Bien le bonjour Michel"

elif [ $1 = "Jean" ]

then

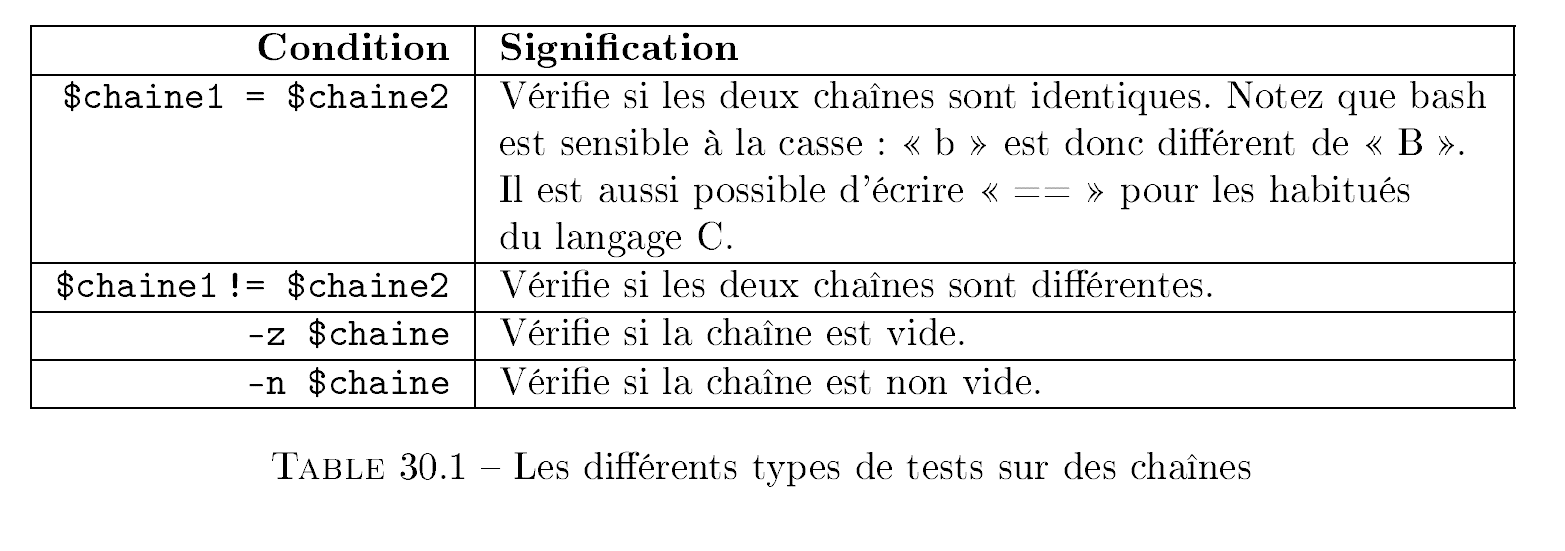
echo "Hé Jean, ça va ?"

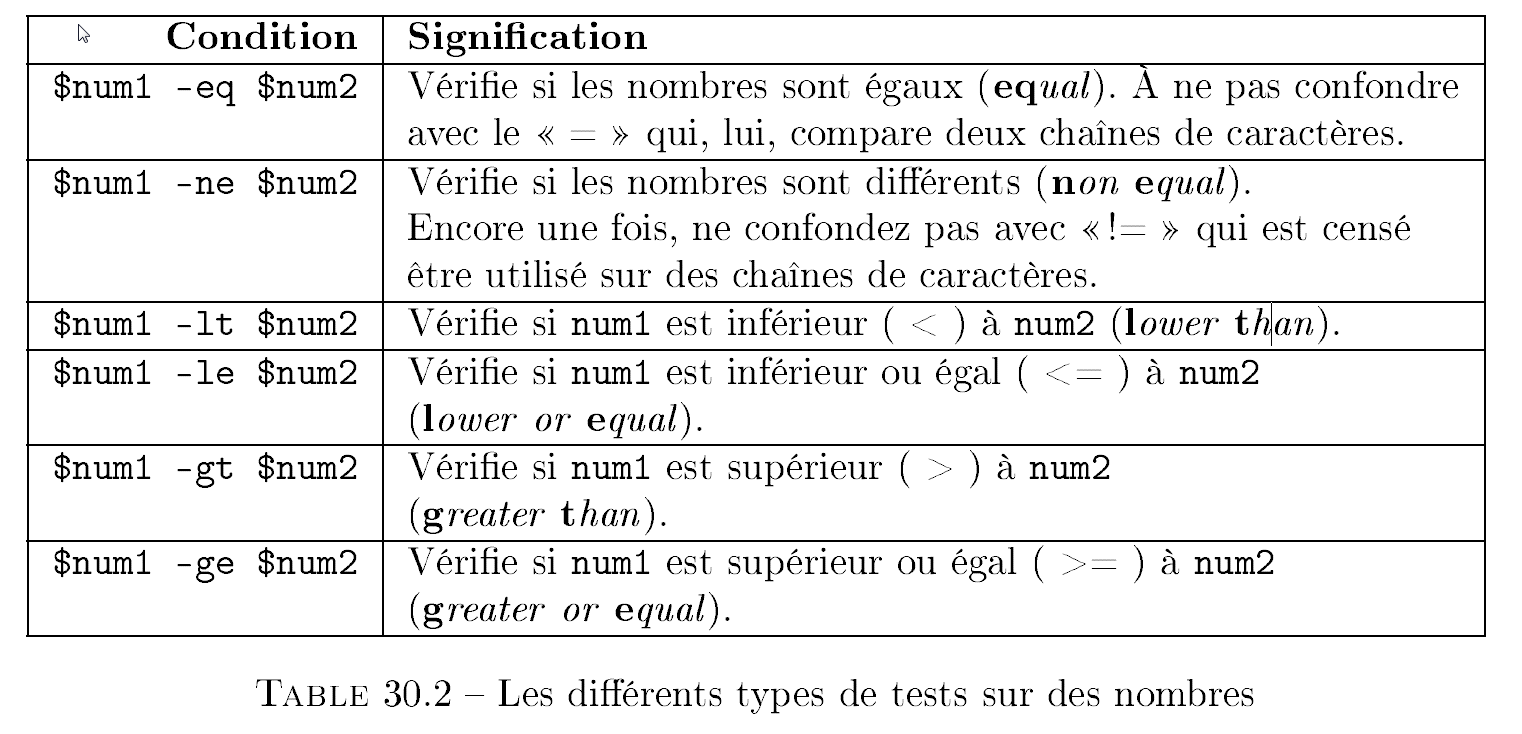
else

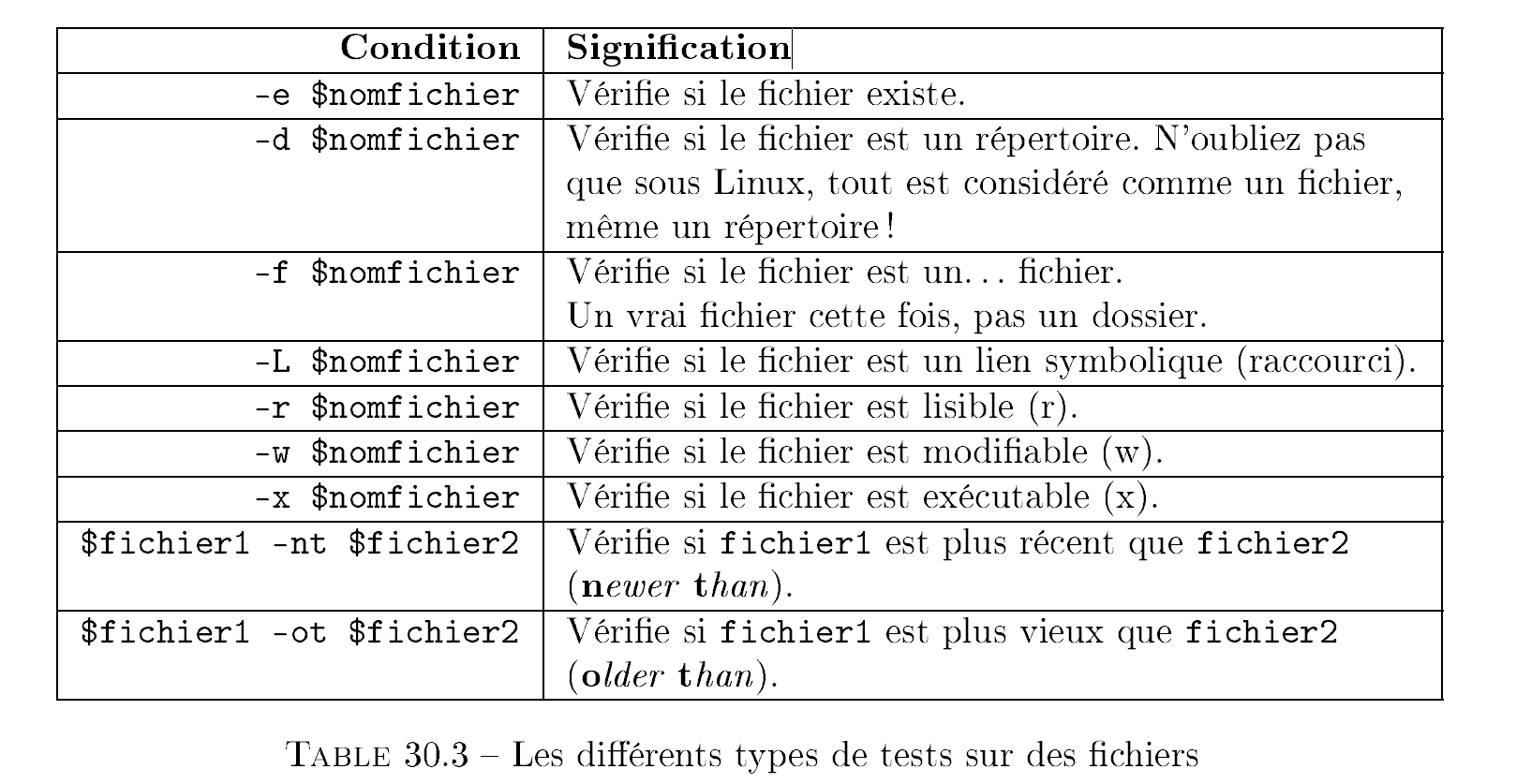
echo "J'te connais pas, ouste !"

fi

* + 1. Le mot fi (if à l'envers !) à la fin indique que le if s'arrête là.
    2. 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 ] !







* 1. Les boucle

while [ test ]

do

echo 'Action en boucle'

done

for variable in 'valeur1' 'valeur2' 'valeur3'

do

echo "La variable vaut $variable"

done

1. ls /bin/\*sh\* : permet de lister les shell existants sur notre machine
2. Lancer le un programme (helloword au démarrage du système : update-rc.d helloworld defaults
3. Supprimer lancement au démarrage : update-rc.d -f helloworld remove