esiea
ECOLE D'INGENIEURS
DU MONDE NUMERIQUE

SYSTÈME

Linux : les commandes de base

By Rexy

Table des matières

1Introduction....3
2Le SHELL : l'interpréteur de commandes....3
2.1Les variables d'environnements....3
2.2Les caractères de substitutions (« jokers »)....3
3Les commandes....3
3.1Syntaxe d'une commande....3
3.2L'aide en ligne....4
3.3Historique des commandes....4
3.4L'environnement de travail....4
Quel est mon terminal de rattachement ?....4
La date et l'heure courante....4
Effacer l'écran du terminal....5
3.5Les commandes d'identification....5
Qui suis je ?....5
Qui d'autre est connecté sur le même système ?....5
Identité d'une machine....5
3.6Le déplacement dans l'arborescence....5
Se situer....5
Chemins relatif et absolu....5
Se déplacer....6
3.7Les Commandes de localisation....6
Rechercher dans une arborescence....6
Localiser un exécutable....7
Afficher les lignes contenant un motif donné....7
3.8Les commandes de visualisation....7
Afficher le contenu du répertoire 7
Format de fichier....7
Contenu d'un fichier....8
Visualiser un fichier....8
Tri du contenu d'un fichier....8
3.9Les commandes de manipulation de fichiers....8
Créer un fichier....8
Concaténer des fichiers....9
Copie de fichiers....9
Déplacer, renommer un fichier....10
Supprimer un fichier 10
3.10Opérations sur les répertoires....10
Créer des répertoires....10
Supprimer un répertoire « vide »....10
Supprimer un répertoire «non-vide »....11
3.11Alias des commandes....11
3.12Conclusion....11

1 Introduction

Le but de ce document est d'une part de vous présenter l'environnement de travail spécifique au monde Linux, d'autre part de passer en revue les commandes essentielles du système d'exploitation.
L'apprentissage de ces commandes permet à l'utilisateur de se connecter sur un terminal Linux, de gérer ses ressources en matière de fichiers, d'identifier la station, le terminal et les utilisateurs connectés sur la station.

2 Le SHELL : l'interpréteur de commandes

Le Shell se comporte comme une boucle infinie.

Début_de_Boucle
    affichage du prompt ($) d'attente du clavier
    lecture d'une commande (RETURN)
    analyse syntaxique (découpe en mots)
    substitution (caractères spéciaux)
    exécution de la commande
Fin_de_Boucle

La sortie de la boucle (entraînant la fermeture se session) s'effectue par : « ***exit*** » , « <***CTRL*> + *D*** » ou « ***logout*** ».

2.1 Les variables d'environnements

Tout utilisateur qui se connecte sur un système Linux bénéficie de variables qui permettent de configurer l'environnement de ce dernier.
Ces variables sont appelées « variables d'environnement ».
Les commandes « ***printenv*** » et « ***set*** » permettent d'afficher les variables définies.
Quelques exemples :

```
$PWD
$OLDPWD
$PS1
$HOME
$PATH
$LOGNAME
$SHELL
```

2.2 Les caractères de substitutions (« jokers »)

Le caractère **« * »** remplace une suite quelconque de caractères. Je recherche tous les fichiers commençant par la lettre « i » dans le répertoire /etc :

```
$ ls /etc/i*
```

Le caractère « **?** » remplace un caractère quelconque et un seul. Je recherche tous les fichiers ayant un « - » en deuxième caractère dans le répertoire /etc :

```
$ ls /etc/?i*
```

Le caractère « **[ ]** » permet de spécifier explicitement une suite de caractères. Je recherche tous les fichiers ayant « x », « y » ou « z » en deuxième lettre dans le répertoire /etc :

```
ls /etc/ ?[x-z]*
```

3 Les commandes

Il ne sera abordé ici qu'un nombre limité de commandes, dire « de base », car nécessaire à la manipulation des ressources élémentaires (fichiers, répertoires).

3.1 Syntaxe d'une commande

Le système d'exploitation fait la distinction entre les minuscules et les majuscules : il est sensible sensible à la casse. Les commandes s'écrivent toutes en minuscules, de même que la plupart des options

La syntaxe générale d'une commande UNIX est : nom_commande [-option(s)] [argument(s)]

- nom_commande : en minuscules.
- La présence des crochets correspond à une convention d'écriture qui signifie que les options et les paramètres sont optionnels.
- La validation est faite par la touche <Entrée>.
- L'espace est utilisé comme séparateur des expressions régulières, entre le nom et les options, entre les options et le(s) argument(s).
- Les options (quand il y en a…) peuvent être concaténées (exemple : *ls -l -i fichier* s'écrit *ls -li fichier*).
- Les arguments peuvent être facultatifs ou obligatoires selon les commandes. Certaines commandes n'en ont pas (exemple : *pwd*). Le nombre d'arguments concernés par la commande n'est pas limité.

3.2 L'aide en ligne

L'utilisation du « ***man »*** est le réflexe à avoir dès qu'un problème de syntaxe survient.
Les commandes complémentaires « ***whatis »*** et « **apropos »** vous informent sur toutes les sections du man concernant cette commande. Les sections correspondent aux thématiques suivantes :

Nro	Thématique
1	Commandes générales
2	Appels système
3	Fonction des bibliothèques standards du C
4	Fichiers spéciaux (pilotes de périphériques généralement situés dans « /dev »)
5	Format de fichier
6	Jeux et économiseurs d'écran
7	Divers
8	Commande et d'administration système et DAEMON

Exemple : « printf » est une commande Linux, mais aussi une fonction C de la bibliothèque « stdio.h ». « ***man printf »*** et « ***man 3 printf »*** fourniront les informations relatives aux 2 thèmes.

3.3 Historique des commandes

Tout système UNIX offre à l'utilisateur des moyens d'afficher les dernières commandes passées sur un terminal donné sous telle identité.
Il existe d'une part les touches de clavier (flèche haut sur LINUX), d'autre part une commande ou des alias standards de cette dernière : « ***fc -l »*** (accès au fichier historique), ou « ***history »***.

3.4 L'environnement de travail

Quel est mon terminal de rattachement ?

Dans le cas où la bannière d'accueil ne renseigne pas sur l'identité du terminal sur lequel on est connecté, la commande « ***tty »*** répond à la question.

```
$ tty
/dev/pts/0
```

De plus en plus souvent, l'utilisateur travaille en environnement graphique. C'est ainsi que les terminaux sont émulés (pts=pseudo terminaux)
Le mode « historique » des terminaux UNIX peut être retrouvé via les séquences de touches « ***<CTRL>*** + ***<ALT>*** + ***<F2 à F6>*** » (F1 pour revenir en mode graphique).

La date et l'heure courante

La commande date renvoie une date complète au format anglais.
En fonction de la valeur de la variable « LANG », le résultat peut être différent (français, anglais...).

```
$ date
```

```
lun jun 9 08:43:51 CET 2008
```

Effacer l'écran du terminal

La commande « ***clear*** » ou « ***<CTRL>*** + ***l*** » permet d'effacer l'écran du terminal sur lequel vous êtes.

3.5 Les commandes d'identification

Qui suis je ?

La commande « ***whoami*** » répond à cette question.
Dans le cas où l'invite de commandes (prompt) ne renseigne pas sur l'identité de l'utilisateur connecté, ce dernier lance la première commande citée.

```
$ whoami
 user1
```

La commande ***id*** fournit plus d'information (groupe d'appartenance)

```
$ id
uid=501(user1) gid=600(groupe1) groupes=600(groupe1)
```

Qui d'autre est connecté sur le même système ?

Dans la mesure où plusieurs sessions sont ouvertes sur la même station, il est intéressant de savoir qui est connecté, ne serait-ce que pour communiquer avec lui par l'envoi de messages. La commande « ***who*** » répond à ce besoin. Exemple avec un utilisateur « rexy » connecté en mode graphique, l'utilisateur connecté via le réseau (ssh) et « root » connecté sur le 2ème terminal texte.

```
$ who
sysadmin      pts/0        Feb 04 15:06    (160.192.12.135)
rexy          :0           Feb 04 12:05
root          tty2         Feb 04 13:06
```

Identité d'une machine

La commande « ***hostname*** » va nous renseigner sur le nom de la machine.

```
$ hostname
 Linux
```

Elle permet également d'attribuer un nom temporaire à une machine (il faut être « root »).

```
# hostname MaMachine
# hostname
 MaMAchine
```

La commande « ***uname –n*** » fournit le même renseignement.
Cette même commande, avec l'option -a, fournit plusieurs informations système.

```
$ uname -n
  MaMachine
$ uname -a
Linux MaMachine 3.8.13.4-desktop-1.mga3 #1 SMP Thu Jul 4 13:56:21 UTC 2013 x86_64 x86_64 x86_64
GNU/Linux
```

3.6 Le déplacement dans l'arborescence

Se situer

Avant de se déplacer, il faut identifier son positionnement dans l'arborescence.
La commande « ***pwd*** » (print working directory) indique le répertoire courant en chemin absolu.

```
/home/rexy
```

Chemins relatif et absolu.

Pour se déplacer sans difficulté au sein de l'arborescence ou organisation générale des fichiers sous UNIX, deux définitions de chemins sont à connaître.

Le nom absolu est le chemin d'accès à un fichier en partant de la racine et en décrivant les différents répertoires traversés.
Le nom relatif désigne la position du fichier par rapport au répertoire courant (celui où nous sommes au temps « t »).
Toute commande UNIX accepte en arguments aussi bien le premier type de chemin que le deuxième.

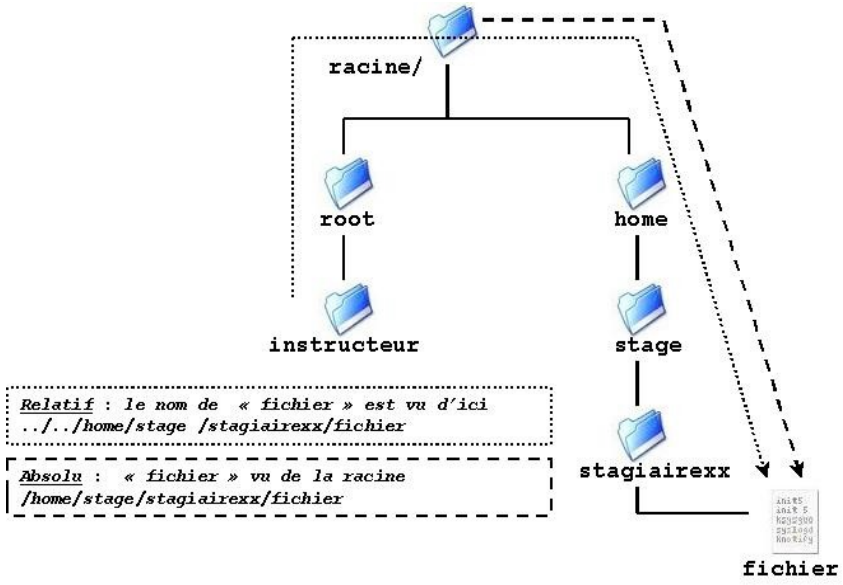

figure 3.1

La figure ci-dessus est un exemple d'organisation des fichiers du système. Hormis la racine, chaque niveau est séparé du suivant par un "/" (slash).

Dans le cas d'un chemin relatif, l'emploi de ".." désigne le répertoire de niveau supérieur (ou répertoire parent). Il peut être utilisé autant de fois que nécessaire pour remonter dans l'arborescence avant de redescendre celle-ci en désignant les répertoires traversés (voir figure 3.1).

Se déplacer

Pour déplacer le répertoire courant vers le répertoire cible, la commande « ***cd »*** (change directory) est employée.
Syntaxe : cd [répertoire_cible].

```
$ cd  /etc/sysconfig
```

Emploi d'un chemin absolu pour atteindre un répertoire.

```
$ cd  ../../var/log
```

Emploi d'un chemin relatif pour atteindre un autre répertoire.
Options :

- revenir à son répertoire de connexion (home directory) à partir de n'importe quel endroit

```
$ cd
```

- Se déplacer dans le répertoire "Document" depuis le répertoire courant.

```
$ cd rep
```

ou (le répertoire « . » correspond au répertoire courant)

```
$ cd ./rep
```

- remonter d'un niveau dans l'arborescence

```
$ cd  ..
```

- revenir au répertoire où l'on était précédemment

```
cd -
pwd
```

3.7 Les Commandes de localisation

Rechercher dans une arborescence

La commande « ***find »*** permet de rechercher un fichier dans l'arborescence Linux. Il est possible de rechercher suivant différents critères, nom, type, permission …
Syntaxe : find [chemin …] [expression]
options : -name, -type, -perm, -user, -group, -exec, -size

L'exemple suivant recherche les fichiers commençant par ifcfg depuis le répertoire /etc/sysconfig

```
$ find /etc/sysconfig/ –name ifcfg*
```

Localiser un exécutable

La commande « ***whereis »*** identifie l'emplacement d'un fichier de type exécutable.

```
$ whereis find
find: /bin/find /usr/bin/find
```

Afficher les lignes contenant un motif donné

La commande « ***grep »*** répond à cette question.
Syntaxe : grep [options] motif fichier

```
$ grep root /etc/passwd
```

Rechercher un motif dans tous les fichiers à partir d'un répertoire donné :

```
$ grep -R ifconfig /etc/sysconfig/
```

3.8 Les commandes de visualisation

Afficher le contenu du répertoire

La commande « ***ls »*** suivie d'un nom de répertoire, affiche le contenu de ce dernier.
Syntaxe : ls [-options] nom_répertoire
Parmi les nombreuses options dont dispose la commande « ***ls »***, on peut citer :

- **-l** affichage des caractéristiques du fichier : type, permissions, propriétaires, taille en octets (par défaut), date de dernière modification (par défaut) ;
- **-i** affichage du numéro d'inode (référence propre du fichier au sein du système de gestion) ;
- **-s** affichage de la taille en blocs de données du fichier ;
- **-a** affichage de tous les fichiers, notamment les fichiers cachés dont le nom commence par le caractère point « . » ;
- **-d** affiche des informations sur le répertoire lui-même et non sur ce qu'il contient.

Affiche le contenu du répertoire courant uniquement par les noms de fichiers.

```
$ ls
fichier1 truc fic2 rep
```

Affiche les informations des fichiers contenus par le répertoire « ***essai »*** situé sous le répertoire « ***home »*** (chemin absolu).

```
$ ls  -l  /home/essai
-rwxr-x---   1   prof   instruc     660  Feb 21 2001  test
drwxr-xr-x-  25  prof   instruc    4096  Feb 21 2001  REPX
```

Affiche les informations du seul répertoire désigné.

```
$ ls  -ld  /home/essai
drwxr-xr-x-  3  prof instruc  1024  Feb 21 2001  /home/essai
```

Format de fichier

La première colonne des informations renvoyée par la commande « ***ls –l »*** affiche le type de fichier selon l'angle de vue du système d'exploitation (cf. cours sur le système de fichiers).
Pour avoir une idée plus précise sur le format de données d'un fichier, il faut utiliser la commande « ***file »*** qui interprète le « nombre magique » affecté à tout format.
Syntaxe : file nom_fichier.

```
$ file fichier repertoire
fichier:     ascii text
repertoire:  directory
```

Contenu d'un fichier

Il existe plusieurs commandes pour afficher le contenu d'un fichier ASCII selon sa taille.
Pour un petit fichier :
Syntaxe : cat nom_fichier

```
$ cat petit
ceci est
un petit fichier
texte
```

Pour des fichiers plus importants à afficher page par page :
Syntaxe : more nom_fichier

- pour faire défiler ligne à ligne, touche « <**Entrée**> »
- pour faire défiler page à page, touche « <***Espace***> »

Visualiser un fichier

On peut être amené à vouloir visualiser certaines parties précises d'un fichier.
Pour afficher les premières lignes d'un fichier, la commande « ***head »*** suivie du nombre de lignes souhaitées est totalement adaptée.
Syntaxe : head –nombre-de-lignes fichier

```
$ head -15 /etc/passwd.
```

De la même façon la commande « ***tail »*** affichera le nombre de lignes souhaitées en fin de fichier.
Syntaxe : tail –nombre-de-lignes fichier

```
tail -15 /var/log/messages
```

La commande « ***tailf*** » (identique à « ***tail -f*** ») permet d'afficher la fin d'un fichier de manière continue. Cela peut être pratique par exemple pour garder un œil sur un journal d'évènements

```
tailf /var/log/security.log
```

Tri du contenu d'un fichier

La commande « ***sort »*** permet de trier, selon certains critères, le contenu des fichiers de type ordinaire.
Syntaxe : sort [-options] fichier
La commande trie le fichier au niveau de l'affichage à l'écran. Pour réaliser le tri réellement dans le fichier, il faut utiliser l'option ***–o*** suivi du nom de fichier en sortie.

```
$ sort /etc/passwd –o /etc/passwd
```

La première partie de la commande trie au niveau de l'affichage écran, la deuxième partie écrit en sortie dans le fichier précisé à la suite de l'option.

3.9 Les commandes de manipulation de fichiers

Créer un fichier

Il n'existe pas de commande spécifique pour créer un fichier. L'utilisateur emploie d'autres commandes détournées de leur fonction d'origine : « ***touch »*** (modification des dates du fichier), « ***cat », « echo »*** (commande d'affichage de chaînes de caractères).
Pour créer un fichier vide ;
Syntaxe : touch nom_fic

```
$ touch  fic_vide
$ ls  -l  fic_vide
-rwxr-x---    1  prof   instruc        0  Sep 27 2008  fic_vide
```

Pour écrire une ligne dans un nouveau fichier ;
Syntaxe : echo "chaîne" > nom_fic

```
$ echo "ceci est un essai"  > essai
$ cat  essai
ceci est un essai
```

Le procédé consiste à rediriger (avec signe supérieur ">") l'affichage écran de la commande ***echo*** dans un fichier. Celui-ci est créé s'il n'existe pas. Si un fichier portant le même nom existait, son contenu sera

écrasé par la chaîne indiquée en argument à la commande « ***echo »***.
Pour créer un fichier texte ;

Syntaxe : Exemple 1		Exemple 2
cat > nom_fichier	ou	cat > nom_fichier <<fin
ligne 1		ligne 1
ligne 2		ligne 2
ligne n		ligne n
<Ctrl><d>		fin

La commande « **cat »** sans arguments permet d'écrire du texte sur autant de lignes que l'on veut. Si le texte affiché à l'écran doit être sauvegardé dans un fichier, il convient de l'indiquer en employant le même procédé de redirection que pour la commande « ***echo »***. La combinaison de touches « ***<Ctrl>+<d> »***, interrompt le processus et indique à la commande « ***cat »*** la fin de saisie.
Dans le deuxième exemple, c'est la chaîne de caractère « fin » qui interrompt le processus et indique à la commande la fin de la saisie.

Concaténer des fichiers

Affichage de deux fichiers à l'écran ;
Syntaxe : cat fichier1 fichier2

```
$ echo "il était une fois"  >  Fichier1
$ echo "dans un pays éloigné…"  >  Fichier2
$ cat  Fichier1  Fichier2
il était une fois
dans un pays éloigné…
```

Concaténer deux fichiers en un fichier unique résultant ;
Syntaxe : cat fichier1 fichier2 > fichier3

```
$ echo "il était une fois "  >  Fichier1
$ echo "dans un pays éloigné…"  >  Fichier2
$ cat  F1  F2  >  /tmp/histoire
$ cat  /tmp/histoire
il était une fois
dans un pays éloigné…
```

Copie de fichiers

La commande « ***cp »*** permet à la fois de dupliquer un fichier en lui changeant de nom et de copier ce fichier à un autre endroit de l'arborescence du système.

- Duplication d'un fichier ;

Syntaxe : cp fichier_source fichier_destination
Dans ce cas, les deux arguments sont deux noms de fichiers.

```
$ cp  fichier1   fichier2
$ ls  fichier*
fichier1  fichier2
```

Attention : si le deuxième argument correspond à un nom de fichier existant, ce dernier sera écrasé.

- Copie d'un (ensemble de) fichier(s) vers un répertoire ;

Syntaxe : cp fichier1 [… fichiern] nom_répertoire
Dans ce cas, le dernier argument est un nom de répertoire.

```
$ cp  pif   rep514
$ cp  pif  paf  pouf  /home/stagiaireYY
$ cp  rep_source/*  rep_dest
$ cp  -R rep1  /home/stagiaireYY
```

- Le premier exemple copie le fichier "***pif***" vers le répertoire "***rep514***", répertoire de niveau inférieur au répertoire courant.
- Le deuxième exemple montre que l'on peut copier en une commande plusieurs fichiers à un endroit donné.

- Le troisième a pour but de dupliquer le contenu de niveau immédiatement inférieur d'un répertoire (excepté les fichiers cachés pour lesquels il faut utiliser ".*") vers un autre répertoire.
- La quatrième utilise une des seules options écrites en majuscule "***-R***", dite option de récursivité, pour dupliquer l'ensemble du contenu du répertoire quel que soit le niveau.

Déplacer, renommer un fichier

La même commande « ***mv »*** est utilisée pour déplacer ou renommer.

- Déplacer un (des) fichier(s) vers un répertoire ;

Syntaxe : mv nom_fichier [… fichiern] nom_répertoire

Dans ce cas, le dernier argument est un nom de répertoire.

```
$ mv  fic_data   Rep_dat
$ mv  pif  paf  pouf  /tmp
```

- Renommer un fichier, un répertoire ;

Syntaxe : mv nom_origine nom_destination

Dans ce cas, les deux arguments sont deux noms de fichiers.

```
$ mv  ancien  nouveau
$ mv  /tmp/pif  /tmp/new_pif
```

Supprimer un fichier

La commande « ***rm »*** (remove) permet de supprimer les fichiers.

Syntaxe : rm nom_fichier. [… fichiern]

Parmi les options dont dispose la commande « ***rm »***, on peut citer :

- **-i** (interactif) : une question de confirmation est posée, auquel il faut répondre "y" pour oui ou "n" pour non ;
- **-f** (force) : aucune question n'est posée ;
- **-r** (récursif) : utile pour supprimer le contenu d'un répertoire.

```
$ rm  F1  F2
$ rm  -f  /tmp/histoire
```

3.10 Opérations sur les répertoires

Créer des répertoires

La commande « ***mkdir »*** (make directory) crée le répertoire indiqué en argument.

Syntaxe : mkdir [-options] nom_répertoire.

```
$ mkdir   monrep
```

Crée le répertoire « ***monrep*** » dans le répertoire courant.

```
$ mkdir  /home/essai
```

Crée le répertoire « ***essai*** » dans le répertoire « ***home*** » situé à la racine "/".

```
$ mkdir  ../rep1  ../../rep2
```

Crée le répertoire « ***rep1*** » dans le répertoire parent et le répertoire « ***rep2*** » un niveau au-dessus du répertoire parent.

```
$ mkdir -p rep1/rep11
```

Crée le répertoire « ***rep11*** » en vérifiant que le répertoire de niveau supérieur "***rep1***" existe, sinon le crée, et ceci en une seule commande.

Attention : il est déconseillé de donner des noms de commandes UNIX aux noms de répertoires ou fichiers.

Supprimer un répertoire « vide »

Le système d'exploitation distingue les répertoires vides de ceux qui ne le sont pas. Il prévoit deux commandes distinctes.

La commande « ***rmdir »*** (remove directory) supprime le répertoire « vide » indiqué en argument.

Syntaxe : rmdir nom_répertoire.

```
$ ls -a  /home/grup
.  ..
```

Vérifie que le répertoire est vide avant suppression.

```
$ rmdir  /home/grup
$ ls  -d  /home/grup
ls:  The file /home/grup does not exist.
```

Une façon de vérifier la suppression !

```
$ rmdir  /tmp
rmdir:  Cannot remove directory tmp
```

L'opération est impossible, car le répertoire « ***tmp*** » n'est pas vide !

Attention :

- Toute suppression est irréversible sous UNIX.
- Les fichiers supprimés ne sont généralement pas déposés dans un répertoire de transit permettant de les récupérer a posteriori. Il n'y a pas de poubelle (trash) excepté sous le terminal graphique de Linux.
- Il n'y a pas en général de demande de confirmation avant exécution.

Supprimer un répertoire «non-vide »

L'autre commande permettant la suppression d'un répertoire "plein" est « ***rm –r »*** (ou « ***rm –R »***). Il s'agit de la commande de suppression des fichiers ordinaires, assortie de l'option « ***-r »***.

Syntaxe : rm -r nom_répertoire.

Cette opération de suppression est récursive et généralement sans demande de confirmation.

```
$ ls  /home/prof
fic1  fic2  rep1  rep2  rep3
$ rm  -r  /home/prof
$ ls  /home/prof/rep1
ls:  The file /home/prof/rep1 does not exist.
```

3.11 Alias des commandes

Tout système UNIX prévoit un certain nombre de raccourcis de commandes ou alias. La liste des alias disponibles sur le système est obtenue par la commande « ***alias »***.

Pour configurer les alias du système, il existe plusieurs solutions :

- pour impacter tous les utilisateurs de votre machine, il faut intervenir sur le fichier « **/etc/profile.d/alias.sh »** et ajouter la ligne de déclaration de l'alias. Si le fichier alias.sh n'est pas présent dans le dossier, il vous suffit de le créer.

alias c = « clear »

- pour un utilisateur en particulier, utiliser le fichier « ***.bashrc »*** situé dans son répertoire de connexion.
- enfin pour déclarer un alias temporaire, utiliser la même commande dans un terminal.

3.12 Conclusion

Une fois la lecture de ce chapitre achevée, l'utilisateur est maintenant prêt à aborder un système d'exploitation Linux, quel qu'il soit !

Si certaines commandes n'ont pas exactement le même fonctionnement (options) suivant les distributions sur lesquelles on travaille, elles gardent toujours une syntaxe standardisée.

Si des connaissances théoriques sont importantes, rien ne remplace l'expérience. Il faut effectivement manipuler et manipuler de manière fréquente !