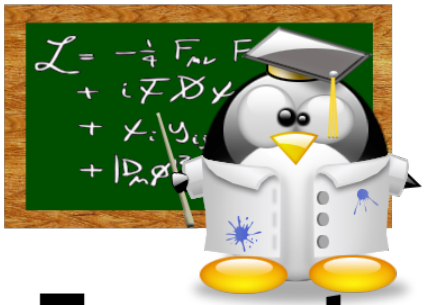


Formatux

Commandes pour utilisateurs Linux

Table des matières

🎓 Objectifs

Apprendre aux futurs administrateurs à :

☑ se **déplacer** dans l'arborescence du système ;
☑ **créer** un fichier texte, **afficher** son contenu et le **modifier** ;
☑ utiliser les commandes les plus utiles de Linux.

Chapitre 1. Généralités

Les systèmes Linux actuels possèdent des utilitaires graphiques dédiés au travail d'un administrateur. Toutefois, il est important d'être capable d'utiliser l'interface en mode ligne de commandes et cela pour plusieurs raisons :

- La majorité des commandes du système sont communes à toutes les distributions Linux, ce qui n'est pas le cas des outils graphiques.
- Il peut arriver que le système ne démarre plus correctement mais qu'un interpréteur de commandes de secours reste accessible.
- L'administration à distance se fait en ligne de commandes avec un terminal SSH.
- Afin de préserver les ressources du serveur, l'interface graphique n'est soit pas installée, soit lancée à la demande.
- L'administration se fait par des scripts.

L'apprentissage de ces commandes permet à l'administrateur de se connecter à un terminal Linux, de gérer ses ressources, ses fichiers, d'identifier la station, le terminal et les utilisateurs connectés, etc.

1.1. Les utilisateurs

L'utilisateur du système Linux est défini (dans le fichier `/etc/passwd`) par :

- un **nom de connexion**, plus communément appelé « login », ne contenant pas d'espace ;
- un identifiant numérique : **UID** (User Identifier) ;
- un identifiant de groupe : **GID** (Group Identifier) ;
- un **mot de passe**, qui sera chiffré avant d'être stocké ;
- un **interpréteur de commandes**, un shell, qui peut être différent d'un utilisateur à l'autre ;
- un **répertoire de connexion**, le *home directory* ;
- une **invite de commande**, ou *prompt* de connexion, qui sera symbolisée par un `#` pour les administrateurs et un `$` pour les autres utilisateurs.

En fonction de la politique de sécurité mise en œuvre sur le système, le mot de passe devra comporter un certain nombre de caractères et respecter des exigences de complexité.

Parmi les interpréteurs de commandes existants, le **Bourne Again Shell** (`/bin/bash`) est celui qui est le plus fréquemment utilisé. Il est affecté par défaut aux nouveaux utilisateurs. Pour diverses raisons, des utilisateurs avancés de Linux choisiront des interpréteurs de commandes alternatifs parmi le Korn Shell (`ksh`), le C Shell (`csh`), etc.

Le répertoire de connexion de l'utilisateur est par convention stocké dans le répertoire `/home` du poste de travail. Il contiendra les données personnelles de l'utilisateur. Par défaut, à la connexion,

le répertoire de connexion est sélectionné comme répertoire courant.

Une installation type poste de travail (avec interface graphique) démarre cette interface sur le terminal 1. Linux étant multi-utilisateurs, il est possible de connecter plusieurs utilisateurs plusieurs fois, sur des **terminaux physiques** (TTY) ou **virtuels** (PTS) différents. Les terminaux virtuels sont disponibles au sein d'un environnement graphique. Un utilisateur bascule d'un terminal physique à l'autre à l'aide des touches `Alt`+`Fx` depuis la ligne de commandes ou à l'aide des touches `Ctrl`+`Alt`+`Fx`.

1.2. Le shell

Une fois que l'utilisateur est connecté sur une console, le shell affiche l'invite de commandes (**`prompt`**). Il se comporte ensuite comme une boucle infinie, à chaque saisie d'instruction :

- affichage de l'invite de commande ;
- lecture de la commande ;
- analyse de la syntaxe ;
- substitution des caractères spéciaux ;
- exécution de la commande ;
- affichage de l'invite de commande ;
- etc.

La séquence de touche `Ctrl`+`C` permet d'interrompre une commande en cours d'exécution.

L'utilisation d'une commande respecte généralement cette séquence :

Séquence d'une commande

```
commande [option(s)] [arguments(s)]
```

Le nom de la commande est **toujours en minuscules**.

Un espace sépare chaque élément.

Les **options courtes** commencent par un tiret (`-l`), alors que les **options longues** commencent par deux tirets (`--list`). Un double tiret (`--`) indique la fin de la liste d'options. Il est possible de regrouper certaines options courtes :

Options courtes

```
$ ls -l -i -a
```

est équivalent à :

Regroupement d'options

```
$ ls -lia
```

Il peut bien entendu y avoir plusieurs arguments après une option :

Arguments d'une commande

```
$ ls -lia /etc /home /var
```

Dans la littérature, le terme « option » est équivalent au terme « paramètre », plus utilisé dans le domaine de la programmation. Le côté optionnel d'une option ou d'un argument est symbolisé en le mettant entre crochets `[` et `]`. Lorsque plusieurs options sont possibles, une barre verticale appelée « pipe » les sépare `[a|e|i]`.

Chapitre 2. Les commandes générales

2.1. Les commandes man et whatis

Il est impossible pour un administrateur, quel que soit son niveau, de connaître toutes les commandes et options dans les moindres détails. Une commande a été spécialement conçue pour accéder en ligne de commandes à un ensemble d'aides, sous forme d'un manuel : la commande `man` (« le man est ton ami »).

Ce manuel est divisé en 8 sections, regroupant les informations par thèmes, la section par défaut étant la section 1 :

1. Commandes utilisateurs ;
2. Appels système ;
3. Fonctions de bibliothèque C ;
4. Périphériques et fichiers spéciaux ;
5. Formats de fichiers ;
6. Jeux ;
7. Divers ;
8. Outils d'administration système et démons.

Des informations sur chaque section sont accessibles en saisissant `man x intro`, `x` indiquant le numéro de section.

La commande :

Syntaxe de la commande man

```
$ man passwd
```

informera l'administrateur sur la commande `passwd`, ses options, etc. Alors qu'un :

Syntaxe de la commande man avec section

```
$ man 5 passwd
```

l'informera sur les fichiers en relations avec la commande.

Toutes les pages du manuel ne sont pas traduites de l'anglais. Elles sont toutefois généralement très précises et fournissent toutes les informations utiles. La syntaxe utilisée et le découpage peuvent dérouter l'administrateur débutant, mais avec de la pratique, l'administrateur y retrouvera rapidement l'information qu'il recherche.

La navigation dans le manuel se fait avec les flèches ↑ et ↓. Le manuel se quitte en appuyant sur la touche `q`.

La commande `whatis` permet de faire une recherche par mot clef au sein des pages de manuel :

Syntaxe de la commande whatis

```
$ whatis clear
```

2.2. La commande shutdown

La commande `shutdown` permet de **stopper électriquement**, immédiatement ou après un certain laps de temps, un serveur Linux.

Syntaxe de la commande shutdown

```
[root]# shutdown [-h] [-r] heure [message]
```

L'heure d'arrêt est à indiquer au format `hh:mm` pour une heure précise, ou `+mm` pour un délai en minutes.

Pour forcer un arrêt immédiat, le mot `now` remplacera l'heure. Dans ce cas, le message optionnel n'est pas envoyé aux autres utilisateurs du système.

- Exemples :

Exemples de la commande shutdown

```
[root]# shutdown -h 0:30 "Arrêt du serveur à 0h30"
[root]# shutdown -r +5
```

- Options :

Table 1. Options de la commande shutdown

Options	Observations
`-h`	Arrête le système électriquement
`-r`	Redémarre le système

2.3. La commande history

La commande `history` permet d'afficher l'historique des commandes qui ont été saisies par l'utilisateur.

Les commandes sont mémorisées dans le fichier `.bash_history` du répertoire de connexion de l'utilisateur.

Exemple de commande history

```
$ history
  147 man ls
  148 man history
```

Table 2. Options de la commande history

Options	Commentaires
`-w`	L'option `-w` permet d'y copier l'historique de la session en cours.
`-c`	L'option `-c` effacera l'historique de la session en cours (mais pas le contenu du fichier `.bash_history`).

- Manipuler l'historique :

Pour manipuler l'historique, des commandes permettent depuis l'invite de commandes de :

Touches	**Fonction**
`!!`	Rappeler la dernière commande passée.
`!n`	Rappeler la commande par son numéro dans la liste.
`!string`	Rappeler la commande la plus récente commençant par la chaîne de caractères.
`[↑]`	Remonter l'historique des commandes.
`[↓]`	Redescendre l'historique des commandes.

2.4. L'auto-complétion

L'auto-complétion est également d'une aide précieuse.

- Elle permet de compléter les commandes, les chemins saisis ou les noms de fichiers.
- Un appui sur la touche `TAB` complète la saisie dans le cas d'une seule solution.
- Sinon, il faudra faire un deuxième appui pour obtenir la liste des possibilités.

Si un double appui sur la touche `TAB` ne provoque aucune réaction de la part du système, c'est qu'il n'existe aucune solution à la complétion en cours.

Chapitre 3. Affichage et identification

3.1. La commande clear

La commande `clear` permet d'effacer le contenu de l'écran du terminal. En réalité, pour être plus précis, elle permet de décaler l'affichage de sorte que l'invite de commandes se retrouve en haut de l'écran sur la première ligne.

Dans un terminal, l'affichage sera définitivement masqué tandis que dans une interface graphique, un ascenseur permettra de remonter dans l'historique du terminal virtuel.

3.2. La commande echo

La commande `echo` permet d'afficher une chaîne de caractères.

Cette commande est plus particulièrement utilisée dans les scripts d'administration pour informer l'utilisateur pendant l'exécution.

L'option `-n` permet de ne pas revenir à la ligne après avoir affiché le texte (ce qui est le comportement par défaut de la commande).

Pour diverses raisons, le développeur du script peut être amené à utiliser des séquences spéciales (commençant par un caractère `\`). Dans ce cas, l'option `-e` sera stipulée, permettant l'interprétation des séquences.

Parmi les séquences fréquemment utilisées, nous citerons :

Table 3. Séquences spéciales de la commande echo

Séquence	Résultat
`\a`	Émet un bip sonore
`\b`	Retour en arrière
`\n`	Ajoute un saut de ligne
`\t`	Ajoute une tabulation horizontale
`\v`	Ajoute une tabulation verticale

3.3. La commande date

La commande `date` permet d'afficher la date et l'heure. La commande respecte la syntaxe suivante :

Syntaxe de la commande date

```
date [-d AAAAMMJJ] [format]
```

Exemples :

```
$ date
mer. Avril 17 16:46:53 CEST 2013
$ date -d 20150729 +%j
210
```

Dans ce dernier exemple, l'option `-d` affiche une date donnée. L'option `+%j` formate cette date pour n'afficher que le quantième.

Le format d'une date peut changer suivant la valeur de la langue définie dans la variable d'environnement `$LANG`.

L'affichage de la date peut suivre les formats suivants :

Table 4. Formats de la commande date

Option	Format
`+%A`	Nom complet du jour
`+%B`	Nom complet du mois
`+%c`	Affichage complet de la date
`+%d`	Numéro du jour
`+%F`	Date au format `AAAA-MM-JJ`
`+%G`	Année
`+%H`	Heure
`+%j`	Quantième du jour
`+%m`	Numéro du mois
`+%M`	Minute
`+%R`	Heure au format `hh:mm`
`+%s`	Secondes depuis le 1er janvier 1970
`+%T`	Heure au format `hh:mm:ss`
`+%u`	Jour de la semaine (`1` pour lundi)
`+%V`	Numéro de la semaine
`+%x`	Date au format `JJ/MM/AAAA`

La commande `date` permet également de modifier la date et l'heure système. Dans ce cas, l'option `-s` sera utilisée.

```
[root]# date -s "2013-04-17 10:19"
jeu. Avril 17 10:19:00 CEST 2013
```

Le format à respecter pour l'argument suivant l'option `-s` est celui-ci :

```
date -s "[AA]AA-MM-JJ hh:mm:[ss]"
```

3.4. Les commandes id, who et whoami

La commande `id` affiche le nom de l'utilisateur courant et ses groupes ou ceux d'un utilisateur, si le login de celui-ci est fourni comme argument.

```
$ id util1
uid=501(util1) gid=501(group1) groups=501(group1),502(group2)
```

Les options `-g`, `-G`, `-n` et `-u` affichent respectivement le GID du groupe principal, les GID des groupes secondaires, les noms au lieu des identifiants numériques et l'UID de l'utilisateur.

La commande `whoami` affiche le login de l'utilisateur courant.

La commande `who` seule affiche le nom des utilisateurs connectés :

```
$ who
stagiaire   tty1    2014-09-15 10:30
root        pts/0   2014-09-15 10:31
```

Linux étant multi-utilisateurs, il est probable que plusieurs sessions soient ouvertes sur la même station, que ce soit physiquement ou à travers le réseau. Il est intéressant de savoir quels utilisateurs sont connectés, ne serait-ce que pour communiquer avec eux par l'envoi de messages.

tty
: représente un terminal.

pts/
: représente une console virtuelle sous environnement graphique.

L'option « `-r` » affiche en plus le niveau d'exécution (voir chapitre « démarrage »).

Chapitre 4. Arborescence de fichiers

Sous Linux, l'arborescence des fichiers se présente sous la forme d'un arbre inversé, appelé **arborescence hiérarchique unique**, dont la racine est le répertoire « `/` ».

Le **répertoire courant** est le répertoire où se trouve l'utilisateur.

Le **répertoire de connexion** est le répertoire de travail associé à l'utilisateur. Les répertoires de connexion sont, en standard, stockés dans le répertoire `/home`.

À la connexion de l'utilisateur, le répertoire courant est le répertoire de connexion.

Un **chemin absolu** référence un fichier depuis la racine en parcourant l'arborescence complète jusqu'au niveau du fichier :

- `/home/groupeA/alice/monfichier`

Le **chemin relatif** référence ce même fichier en parcourant l'arborescence complète depuis le répertoire courant :

- `../alice/monfichier`

Dans l'exemple précèdent, les "`..`" font référence au répertoire parent du répertoire actuel.

Un répertoire, même s'il est vide, contiendra obligatoirement au minimum **deux références** :

« `.` »
: référence sur lui-même.

« `..` »
: référence le répertoire parent du répertoire actuel.

Un chemin relatif peut ainsi commencer par « `./` » ou par « `../` ». Lorsque le chemin relatif fait référence à un sous dossier ou à un fichier du répertoire courant, alors le « `./` » est souvent omis. Mentionner le premier « `./` » de l'arborescence ne sera réellement requis que pour lancer un fichier exécutable.

Les erreurs dans les chemins peuvent être la cause de nombreux problèmes : création de dossier ou de fichiers aux mauvais endroits, suppressions involontaires, etc. Il est donc fortement recommandé d'utiliser l'auto-complétion (cf. 2.2) lors des saisies de chemin.

Figure 1. Notre arborescence exemple

Dans l'exemple ci-dessus, nous cherchons à donner l'emplacement du fichier monfichier depuis le répertoire de bob.

- Par un **chemin absolu**, le répertoire courant importe peu. Nous commençons par la racine, pour descendre successivement dans les répertoires "`home`", "`groupeA`", "`alice`" et enfin le fichier "`monfichier`" : **`/home/groupeA/alice/monfichier`**.
- Par un **chemin relatif**, notre point de départ étant le répertoire courant "`bob`", nous remontons d'un niveau par "`..`" (soit dans le répertoire `groupeA`), puis nous descendons dans le répertoire "`alice`", et enfin le fichier "`monfichier`" : **`../alice/monfichier`**.

4.1. La commande pwd

La commande `pwd` (Print Working Directory) affiche le chemin absolu du répertoire courant.

```
$ pwd
/home/stagiaire
```

Pour se déplacer à l'aide d'un chemin relatif, il faut impérativement connaître son positionnement dans l'arborescence.

Selon l'interpréteur de commandes, l'invite de commandes peut également afficher le nom du répertoire courant.

4.2. La commande cd

La commande `cd` (Change Directory) permet de changer le répertoire courant, autrement dit, de se déplacer dans l'arborescence.

```
$ cd /tmp
$ pwd
/tmp
$ cd ../
$ pwd
/
$ cd
$ pwd
/home/stagiaire
```

Comme vous pouvez le constater dans le dernier exemple ci-dessus, la commande `cd` sans argument permet de repositionner le répertoire courant sur le répertoire de connexion (*home directory*).

4.3. La commande ls

La commande `ls` affiche le contenu d'un répertoire.

Syntaxe de la commande ls

```
ls [-a] [-i] [-l] [repertoire1] [repertoire2] [···]
```

Exemple :

```
$ ls /home
.   ..  stagiaire
```

Les options principales de la commande `ls` sont :

Table 5. Options principales de la commande ls

Option	Information
`-a`	Affiche tous les fichiers, même ceux cachés. Les fichiers cachés sous Linux sont ceux qui commencent par un "`.`".
`-i`	Affiche les numéros d'inode.
`-l`	Affiche sous forme de liste verticale la liste des fichiers avec des informations supplémentaires formatées par colonnes.

La commande ls offre toutefois de très nombreuses options (voir le man) :

Table 6. Options complémentaires de la commande ls

Option	Information
`-d`	Affiche les informations d'un répertoire au lieu de lister son contenu.
`-g`	Affiche les UID et GID plutôt que les noms des propriétaires.

Option	Information
`-h`	Affiche les tailles de fichiers dans le format le plus adapté (octet, kilo-octet, méga-octet, giga-octet, ...). `h` pour Human Readable.
`-s`	Affiche la taille en octets (sauf si option `k`).
`-A`	Affiche tous les fichiers du répertoire sauf "`.`" et "`..`".
`-R`	Affiche récursivement le contenu des sous répertoires.
`-F`	Affiche le type des fichiers. Imprime un `/` pour un répertoire, `*` pour les exécutables, `@` pour un lien symbolique, et rien pour un fichier texte.
`-X`	Trier les fichiers en fonction de leurs extensions.

- Description des colonnes :

```
$ ls -lia /home
78489 drwx------ 4 stagiaire users 4096 25 oct. 08:10 stagiaire
```

Table 7. Description des colonnes du résultat généré par la commande ls

Valeur	Information.
`78489`	Numéro d'inode.
`drwx------`	Type de fichier (`d`) et droits (`rwx------`).
`4`	Nombre de sous-répertoires ("`.`" et "`..`" inclus). Pour un fichier de type lien physique : nombre de liens physiques.
`stagiaire`	Utilisateur propriétaire.
`users`	Groupe propriétaire.
`4096`	Taille en octets.
`25 oct. 08:10`	Date de dernière modification.
`stagiaire`	Nom du fichier (ou du répertoire).

Des **alias** sont fréquemment positionnés au sein des distributions courantes.

C'est le cas de l'alias `ll` :

Alias de la commande ls -l

```
alias ll='ls -l --color=auto'
```

La commande `ls` dispose de nombreuses options dont voici quelques exemples avancés d'utilisations :

- Lister les fichiers de `/etc` par ordre de dernière modification :

```
$ ls -ltr /etc
total 1332
-rw-r--r--.  1 root root    662 29 aout   2007 logrotate.conf
-rw-r--r--.  1 root root    272 17 nov.   2009 mailcap
-rw-------.  1 root root    122 12 janv.  2010 securetty
...
-rw-r--r--.  2 root root     85 18 nov.  17:04 resolv.conf
-rw-r--r--.  1 root root     44 18 nov.  17:04 adjtime
-rw-r--r--.  1 root root    283 18 nov.  17:05 mtab
```

- Lister les fichiers de `/var` plus gros qu'un méga-octet mais moins qu'un giga-octets :

```
[root]# ls -Rlh /var | grep [0-9]M
...
-rw-r--r--. 1 apache apache 1,2M 10 nov.  13:02 XB RiyazBdIt.ttf
-rw-r--r--. 1 apache apache 1,2M 10 nov.  13:02 XB RiyazBd.ttf
-rw-r--r--. 1 apache apache 1,1M 10 nov.  13:02 XB RiyazIt.ttf
...
```

- Afficher les droits sur un dossier :

Pour connaître les droits sur un dossier, dans notre exemple `/etc`, la commande suivante ne conviendrait pas :

```
$ ls -l /etc
total 1332
-rw-r--r--.  1 root root     44 18 nov.  17:04 adjtime
-rw-r--r--.  1 root root   1512 12 janv.  2010 aliases
-rw-r--r--.  1 root root  12288 17 nov.  17:41 aliases.db
drwxr-xr-x.  2 root root   4096 17 nov.  17:48 alternatives
...
```

puisque cette dernière liste par défaut le contenu du dossier et non le contenant.

Pour ce faire, il faut utiliser l'option `-d` :

```
$ ls -ld /etc
drwxr-xr-x. 69 root root 4096 18 nov.  17:05 /etc
```

- Lister les fichiers par taille :

```
$ ls -lhS
```

- Afficher la date de modification au format “timestamp” :

```
$ ls -l --time-style="+%Y-%m-%d $newline%m-%d %H:%M" /
total 12378
dr-xr-xr-x. 2 root root 4096 2014-11-23 11-23 03:13 bin
dr-xr-xr-x. 5 root root 1024 2014-11-23 11-23 05:29 boot
```

- Ajouter le “trailing slash” à la fin des dossiers :

Par défaut, la commande `ls` n’affiche pas le dernier slash d’un dossier.

Dans certains cas, comme pour des scripts par exemple, il est utile de les afficher :

```
$ ls -dF /etc
/etc/
```

- Masquer certaines extensions :

```
$ ls /etc --hide=*.conf
```

4.4. La commande mkdir

La commande `mkdir` crée un répertoire ou une arborescence de répertoire.

Syntaxe de la commande mkdir

```
mkdir [-p] repertoire [repertoire] [···]
```

Exemple :

```
$ mkdir /home/stagiaire/travail
```

Le répertoire « `stagiaire` » devra exister pour créer le répertoire « `travail` ».

Sinon, l’option « `-p` » devra être utilisée. L’option « `-p` » crée les répertoires parents s’ils n’existent pas.

Il est vivement déconseillé de donner des noms de commandes UNIX comme noms de répertoires ou fichiers.

4.5. La commande touch

La commande `touch` modifie l'horodatage d'un fichier ou crée un fichier vide si le fichier n'existe pas.

Syntaxe de la commande touch

```
touch [-t date] fichier
```

Exemple :

```
$ touch /home/stagiaire/fichier
```

Option	Information
`-t date`	Modifie la date de dernière modification du fichier avec la date précisée. Date au format : `[AAAA]MMJJhhmm[ss]`

La commande touch est utilisée en priorité pour créer un fichier vide, mais elle peut avoir un intérêt dans le cadre de sauvegarde incrémentale ou différentielle. En effet, le fait d'exécuter un touch sur un fichier aura pour seul effet de forcer sa sauvegarde lors de la sauvegarde suivante.

4.6. La commande rmdir

La commande `rmdir` supprime un répertoire vide.

Exemple :

```
$ rmdir /home/stagiaire/travail
```

Option	Information
`-p`	Supprime le ou les répertoire(s) parent(s) à la condition qu'ils soient vides.

Pour supprimer à la fois un répertoire non-vide et son contenu, il faudra utiliser la commande `rm`.

4.7. La commande rm

La commande `rm` supprime un fichier ou un répertoire.

Syntaxe de la commande rm

```
rm [-f] [-r] fichier [fichier] […]
```

ATTENTION !!! Toute suppression de fichier ou de répertoire est définitive.

Table 8. Options de la commande rm

Options	Information
`-f`	Ne demande pas de confirmation de la suppression.
`-i`	Demande de confirmation de la suppression.
`-r`	Supprime récursivement les sous-répertoires.

La commande `rm` en elle-même ne demande pas de confirmation lors de la suppression de fichiers. Ce comportement est propre à la distribution RedHat/CentOS.

La commande `rm` est ici un alias de la commande `rm -i`. Ne soyez pas surpris sur une autre distribution, type Debian par exemple, de ne pas obtenir de demande de confirmation.

La suppression d'un dossier à l'aide de la commande `rm`, que ce dossier soit vide ou non, nécessitera l'ajout de l'option `-r`.

La fin des options est signalée au shell par un double tiret "`--`".

Dans l'exemple :

```
$ >-dur-dur # Creer un fichier vide appelé -dur-dur
$ rm -f -- -dur-dur
```

Le nom du fichier `-dur-dur` commence par un "`-`". Sans l'usage du "`--`" le shell aurait interprété le "`-d`" de "`-dur-dur`" comme une option.

4.8. La commande mv

La commande `mv` déplace et renomme un fichier.

Syntaxe de la commande mv

```
mv fichier [fichier …] destination
```

Exemples :

```
$ mv /home/stagiaire/fic1 /home/stagiaire/fic2
$ mv /home/stagiaire/fic1 /home/stagiaire/fic2 /tmp
```

Table 9. Options de la commande mv

Options	Information
`-f`	Ne demande pas de confirmation si écrasement du fichier de destination.
`-i`	Demande de confirmation si écrasement du fichier de destination (par défaut).

Quelques cas concrets permettront de mieux saisir les difficultés qui peuvent se présenter :

```
$ mv /home/stagiaire/fic1 /home/stagiaire/fic2
```

Permet de renommer “`fic1`” en “`fic2`”, si “`fic2`” existe déjà, il sera remplacé par “`fic1`”.

```
$ mv /home/stagiaire/fic1 /home/stagiaire/fic2 /tmp
```

Permet de déplacer “`fic1`” et “`fic2`” dans le répertoire “`/tmp`”.

```
$ mv fic1 /repexiste/fic2
```

« `fic1` » est déplacé dans « `/repexiste` » et renommé « `fic2` ».

```
$ mv fic1 fic2
```

« `fic1` » est renommé « `fic2` ».

```
$ mv fic1 /repexiste
```

Si le répertoire de destination existe, « `fic1` » est déplacé dans « `/repexiste` ».

```
$ mv fic1 /repexistepas
```

Si le répertoire de destination n’existe pas, « `fic1` » est renommé « `repexistepas` » à la racine.

4.9. La commande cp

La commande `cp` copie un fichier.

Syntaxe de la commande cp

```
cp fichier [fichier ⋯] destination
```

Exemple :

```
$ cp -r /home/stagiaire /tmp
```

Table 10. Options de la commande cp

Options	Information
`-i`	Demande de confirmation si écrasement (par défaut).
`-f`	Ne demande pas de confirmation si écrasement du fichier de destination.
`-p`	Conserve le propriétaire, les permissions et l'horodatage du fichier copié.
`-r`	Copie un répertoire avec ses fichiers et sous-répertoires.

Quelques cas concrets permettront de mieux saisir les difficultés qui peuvent se présenter :

```
$ cp fic1 /repexiste/fic2
```

« `fic1` » est copié dans « `/repexiste` » sous le nom « `fic2` ».

```
$ cp fic1 fic2
```

« `fic1` » est copié sous le nom « `fic2` » dans ce répertoire.

```
$ cp fic1 /repexiste
```

Si le répertoire de destination existe, « `fic1` » est copié dans « `/repexiste` ».

```
$ cp fic1 /repexistepas
```

Si le répertoire de destination n'existe pas, « `fic1` » est copié sous le nom « `repexistepas` ».

Chapitre 5. Visualisation

5.1. La commande file

La commande `file` affiche le type d'un fichier.

Syntaxe de la commande file

```
file fichier [fichiers]
```

Exemple :

```
$ file /etc/passwd /etc
/etc/passwd:    ASCII text
/etc:       directory
```

5.2. La commande more

La commande `more` affiche le contenu d'un ou de plusieurs fichiers écran par écran.

Syntaxe de la commande more

```
more fichier [fichiers]
```

Exemple :

```
$ more /etc/passwd
root:x:0:0:root:/root:/bin/bash
...
```

En utilisant la touche `ENTREE`, le déplacement se fait ligne par ligne. En utilisant la touche `ESPACE`, le déplacement se fait page par page.

5.3. La commande less

La commande `less` affiche le contenu d'un ou de plusieurs fichiers. La commande `less` est interactive et possède des commandes d'utilisation qui lui sont propres.

Syntaxe de la commande less

```
less fichiers [fichiers]
```

Les commandes propres à `less` sont :

Table 11. Commandes internes à less

Commande	Action
`h`	Aide.
`Flèches`	Monter, descendre d'une ligne ou pour aller à droite ou à gauche.
`Entrée`	Descendre d'une ligne.
`Espace`	Descendre d'une page.
`PgAR` ou `PgAV`	Monter ou descendre d'une page.
`Pos1` ou `Fin`	Se placer en début de fichier ou en fin de fichier.
`/texte`	Rechercher le texte.
`q`	Quitter la commande less.

5.4. La commande cat

La commande `cat` concatène (mettre bout à bout) le contenu de plusieurs fichiers et affiche le résultat sur la sortie standard.

Syntaxe de la commande cat

```
cat fichier [fichiers]
```

Exemple 1 - Afficher le contenu d'un fichier vers la sortie standard :

```
$ cat /etc/passwd
```

Exemple 2 - Afficher le contenu de plusieurs fichiers vers la sortie standard :

```
$ cat /etc/passwd /etc/group
```

Exemple 3 - Afficher le contenu de plusieurs fichiers et rediriger la sortie standard :

```
$ cat /etc/passwd /etc/group > utilisateursEtGroupes.txt
```

Exemple 4 - Afficher la numérotation des lignes :

```
$ cat -n /etc/passwd
 1 root:x:0:0:root:/root:/bin/bash
 2 bin:x:1:1:bin:/bin:/sbin/nologin
...
```

Exemple 5 - Affiche la numérotation des lignes non vides :

```
$ cat -b /etc/openldap/ldap.conf
1 #
2 # LDAP Defaults
3 #

4 # See ldap.conf(5) for details
5 # This file should be world readable but not world writable
```

5.5. La commande tac

La commande `tac` fait quasiment l'inverse de la commande `cat`. Elle affiche le contenu d'un fichier en commençant par la fin (ce qui est particulièrement intéressant pour la lecture des logs !).

Exemple : Afficher un fichier de logs en affichant en premier la dernière ligne :

```
[root]# tac /var/log/messages | less
```

5.6. La commande head

La commande `head` affiche le début d'un fichier.

Syntaxe de la commande head

```
head [-n x] fichier
```

Table 12. Options de la commande head

Option	Observation
`-n x`	Affiche les x premières lignes du fichier

Par défaut (sans l'option `-n`), la commande head affichera les 10 premières lignes du fichier.

5.7. La commande tail

La commande `tail` affiche la fin d'un fichier.

Syntaxe de la commande tail

```
tail [-f] [-n x] fichier
```

Table 13. Options de la commande tail

Option	Observation
`-n x`	Affiche les `x` dernières lignes du fichier
`-f`	Affiche les modifications du fichier en temps réel

Exemple :

```
$ tail -n 3 /etc/passwd
sshd:x:74:74:Privilege-separeted sshd:/var/empty /sshd:/sbin/nologin
tcpdump::x:72:72::/:/sbin/nologin
user1:x:500:500:grp1:/home/user1:/bin/bash
```

Avec l'option `-f`, la commande tail ne rend pas la main et s'exécute tant que l'utilisateur ne l'interrompt pas par la séquence `[CTRL]` + `[C]`. Cette option est très fréquemment utilisée pour suivre les fichiers journaux (les logs) en temps réel.

Sans l'option `-n`, la commande tail affiche les 10 dernières lignes du fichier.

5.8. La commande sort

La commande `sort` trie les lignes d'un fichier.

Elle permet d'ordonner, ranger dans un ordre donné, le résultat d'une commande ou le contenu d'un fichier, selon un ordre numérique, alphabétique, par ordre de grandeur (Ko, Mo, Go) ou dans l'ordre inverse.

Syntaxe de la commande sort

```
sort [-kx] [-n] [-o fichier] [-ty] fichier
```

Exemple :

```
$ sort -k3 -t: -n /etc/passwd
root:x:0:0:root:/root:/bin/bash
adm:x:3:4:adm:/var/adm/:/sbin/nologin
```

Table 14. Options de la commande sort

Option	Observation
`-kx`	Précise la colonne `x` sur laquelle se fera le tri
`-n`	Demande un tri numérique
`-o fichier`	Enregistre le tri dans le fichier précisé
`-ty`	Précise le caractère séparateur de champs `y`
`-r`	Inverse l'ordre du résultat

La commande `sort` ne trie le fichier qu'à l'affichage écran. Le fichier n'est pas modifié par le tri. Pour enregistrer le tri, il faut utiliser l'option `-o` ou une redirection de sortie `>`.

Par défaut, le tri des nombres se fait selon leur caractère. Ainsi, "110" sera avant "20", qui sera lui-même avant "3". Il faut préciser l'option `-n` pour que les blocs caractères numériques soient bien triés par leur valeur.

La commande `sort` permet d'inverser l'ordre des résultats, avec l'option `-r` :

```
$ sort -k3 -t: -n -r /etc/passwd
root:x:0:0:root:/root:/bin/bash
adm:x:3:4:adm:/var/adm/:/sbin/nologin
```

Dans cet exemple, la commande `sort` rangera cette fois-ci le contenu du fichier `/etc/passwd` du plus grand uid au plus petit.

Quelques exemples avancés d'utilisation de la commande `sort` :

- Mélanger les valeurs

La commande `sort` permet également de mélanger les valeurs avec l'option `-R` :

```
$ sort -R /etc/passwd
```

- Trier des adresses IP

Un administrateur système est rapidement confronté au traitement des adresses IP issues des logs de ses services comme SMTP, VSFTP ou Apache. Ces adresses sont typiquement extraites avec la commande `cut`.

Voici un exemple avec le fichier `client-dns.txt` :

```
192.168.1.10
192.168.1.200
5.1.150.146
208.128.150.98
208.128.150.99
```

```
$ sort -nr client-dns.txt
208.128.150.99
208.128.150.98
192.168.1.200
192.168.1.10
5.1.150.146
```

- Trier des tailles de fichiers

La commande `sort` sait reconnaître les tailles de fichiers, issues de commande comme `ls` avec l'option `-h`.

Voici un exemple avec le fichier `taille.txt` :

```
1,7G
18M
69K
2,4M
1,2M
4,2G
6M
124M
12,4M
4G
```

```
[root]# sort -hr taille.txt
4,2G
4G
1,7G
124M
18M
12,4M
6M
2,4M
1,2M
69K
```

5.9. La commande wc

La commande `wc` compte le nombre de lignes, mots ou octets d'un fichier.

Syntaxe de la commande wc

```
wc [-l] [-m] [-w] fichier [fichiers]
```

Table 15. Options de la commande wc

Option	Observation
`-c`	Compte le nombre d'octets.
`-m`	Compte le nombre de caractères.
`-l`	Compte le nombre de lignes.
`-w`	Compte le nombre de mots.

Chapitre 6. Recherche

6.1. La commande find

La commande `find` recherche l'emplacement d'un fichier.

Syntaxe de la commande find

```
find repertoire [-name nom] [-type type] [-user login] [-date date]
```

Les options de la commande `find` étant très nombreuses, il est préférable de se référer au `man`.

Si le répertoire de recherche n'est pas précisé, la commande `find` cherchera à partir du répertoire courant.

Table 16. Options de la commande find

Option	Observation
`-perm permissions`	Recherche des fichiers selon leurs permissions.
`-size taille`	Recherche des fichiers selon leur taille.

6.2. L'option -exec de la commande find

Il est possible d'utiliser l'option `-exec` de la commande `find` pour exécuter une commande à chaque ligne de résultat :

```
$ find /tmp -name *.txt -exec rm -f {} \;
```

La commande précédente recherche tous les fichiers du répertoire `/tmp` nommés `*.log` et les supprime.

Comprendre l'option -exec

Dans l'exemple ci-dessus, la commande find va construire une chaîne de caractères représentant la commande à exécuter.

Si la commande find trouve trois fichiers nommés `log1.txt`, `log2.txt` et `log3.txt`, alors la commande `find` va construire la chaîne en remplaçant dans la chaîne "`rm -f {} \;`" les accolades par un des résultats de la recherche, et cela autant de fois qu'il y a de résultats.

Ce qui nous donnera :

```
rm -f /tmp/log1.txt ; rm -f /tmp/log2.txt ; rm -f /tmp/log3.txt ;
```

Le caractère "`;`" est un caractère spécial du shell qui doit être protégé par un "`\`" pour éviter son interprétation trop tôt par la commande `find` (et non plus dans le `-exec`).

6.3. La commande whereis

La commande `whereis` recherche des fichiers liés à une commande.

Syntaxe de la commande whereis

```
whereis [-b] [-m] [-s] commande
```

Exemple :

```
$ whereis -b ls
ls: /bin/ls
```

Table 17. Options de la commande whereis

Option	Observation
`-b`	Ne recherche que le fichier binaire.
`-m`	Ne recherche que les pages de manuel.
`-s`	Ne recherche que les fichiers sources.

6.4. La commande grep

La commande `grep` recherche une chaîne de caractères dans un fichier.

Syntaxe de la commande grep

```
grep [-w] [-i] [-v] "chaîne" fichier
```

Exemple :

```
$ grep -w "root:" /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

Table 18. Options de la commande grep

Option	Observation
`-i`	Ignore la casse de la chaîne de caractères recherchée.
`-v`	Inverse le résultat de la recherche.
`-w`	Recherche exactement la chaîne de caractères précisée.

La commande `grep` retourne la ligne complète comprenant la chaîne de caractères recherchée.

- Le caractère spécial `^` permet de rechercher une chaîne de caractères placée en début de ligne.
- Le caractère spécial `$` permet de rechercher une chaîne de caractères placée en fin de ligne.

```
$ grep -w "^root" /etc/passwd
```

Cette commande est très puissante et il est fortement conseillé de consulter son manuel. Elle a de nombreux dérivés,

Il est possible de rechercher une chaîne de caractères dans une arborescence de fichiers avec l'option `-R`.

```
$ grep -R "Virtual" /etc/httpd
```

6.5. Les méta-caractères

Les méta-caractères se substituent à un ou plusieurs caractères (voire à une absence de caractère) lors d'une recherche.

Ils sont combinables.

Le caractère `*` remplace une chaîne composée de plusieurs caractères quelconques. Le caractère `*` peut également représenter une absence de caractère.

```
$ find /home -name test*
/home/stagiaire/test
/home/stagiaire/test1
/home/stagiaire/test11
/home/stagiaire/tests
/home/stagiaire/test362
```

Les méta-caractères permettent des recherches plus complexes en remplaçant tout ou partie d'un mot. Il suffit de remplacer les inconnues par ces caractères spéciaux.

Le caractère "**?**" remplace un unique caractère, quel qu'il soit.

```
$ find /home -name test?
/home/stagiaire/test1
/home/stagiaire/tests
```

Les crochets "**[ ]**" permettent de spécifier les valeurs que peut prendre un unique caractère.

```
$ find /home -name test[123]*
/home/stagiaire/test1
/home/stagiaire/test11
/home/stagiaire/test362
```

Il ne faut pas confondre les méta-caractères du shell et ceux des expressions régulières. La commande grep utilise les méta-caractères des expressions régulières.

Chapitre 7. Redirections et tubes

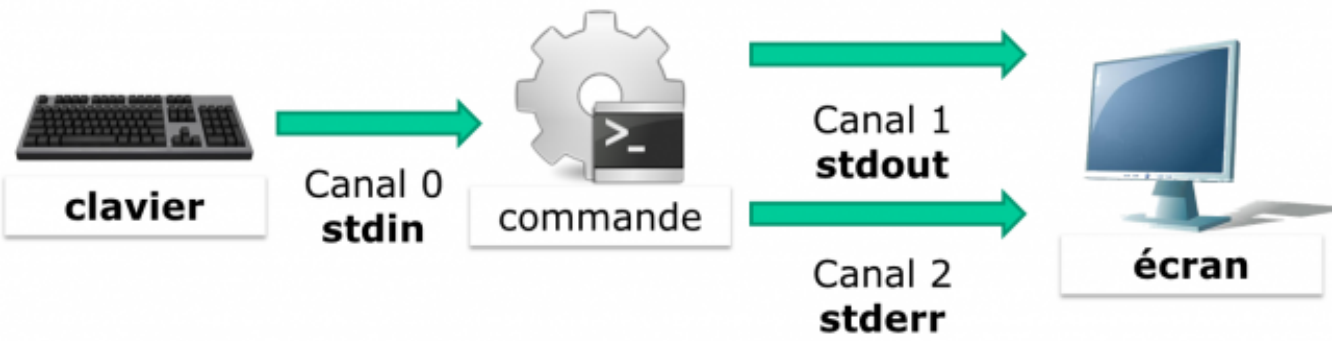
7.1. L'entrée et les sorties standards

Sur les systèmes UNIX et Linux, les flux standards sont aux nombres de trois. Ils permettent aux programmes, via la bibliothèque stdio.h de faire entrer ou sortir des informations.

Ces flux sont appelés canal X ou descripteur X de fichier.

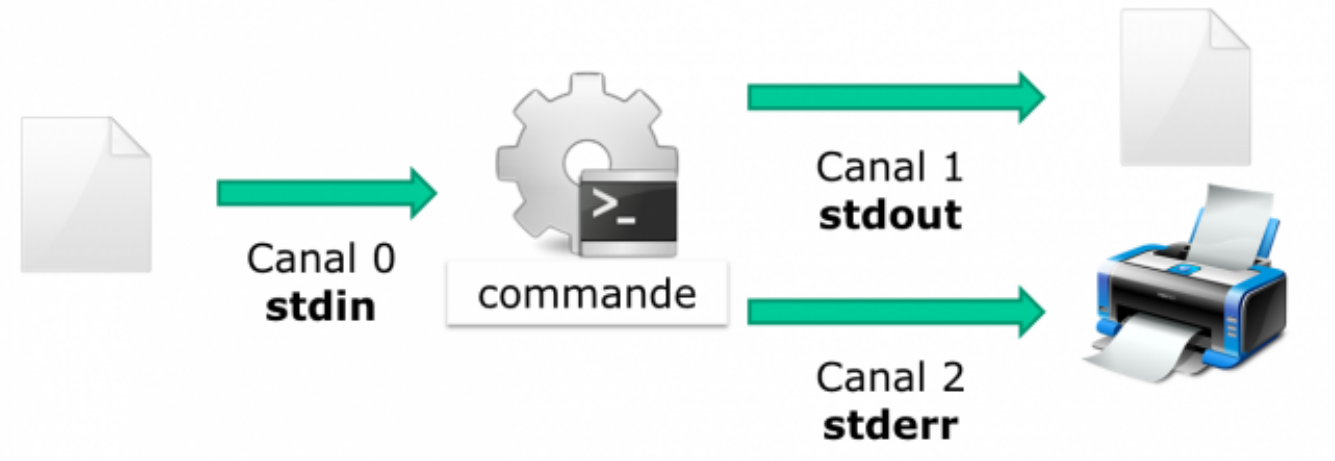
Par défaut :

- le clavier est le périphérique d'entrée pour le canal 0, appelé **stdin** ;
- l'écran est le périphérique de sortie pour les canaux 1 et 2, appelés **stdout** et **stderr**.



stderr reçoit les flux d'erreurs renvoyés par une commande. Les autres flux sont dirigés vers stdout.

Ces flux pointent vers des fichiers périphériques, mais comme tout est fichier sous UNIX, les flux d'entrées/sorties peuvent facilement être détournés vers d'autres fichiers. Ce principe fait toute la force du shell.



7.2. La redirection d'entrée

Il est possible de rediriger le flux d'entrée depuis un autre fichier avec le caractère inférieur `<` ou `<<`. La commande lira le fichier au lieu du clavier :

```
$ ftp -in serverftp << cdes-ftp.txt
```

Seules les commandes demandant une saisie au clavier pourront gérer la redirection d'entrée.

La redirection d'entrée peut également être utilisée pour simuler une interactivité avec l'utilisateur. La commande lira le flux d'entrée jusqu'à rencontrer le mot clef défini après la redirection d'entrée.

Cette fonctionnalité est utilisée pour scripter des commandes interactives :

```
$ ftp -in serverftp << FIN
user alice password
put fichier
bye
FIN
```

Le mot clef `FIN` peut être remplacé par n'importe quel mot.

```
$ ftp -in serverftp << STOP
user alice password
put fichier
bye
STOP
```

Le shell quitte la commande `ftp` lorsqu'il reçoit une ligne ne contenant que le mot clef.

La redirection de l'entrée standard est peu utilisée car la plupart des commandes acceptent un nom de fichier en argument.

La commande `wc` pourrait s'utiliser ainsi :

```
$ wc -l .bash_profile
27 .bash_profile # le nombre de lignes est suivi du nom du fichier
$ wc -l < .bash_profile
27 # le nombre de lignes est seul
```

7.3. Les redirections de sortie

Les sorties standards peuvent être redirigées vers d'autres fichiers grâce aux caractères > ou >>.

La redirection simple > écrase le contenu du fichier de sortie :

```
$ date +%F > fic_date
```

alors que la redirection double >> ajoute (concatène) au contenu du fichier de sortie.

```
$ date +%F >> fic_date
```

Dans les deux cas, le fichier est automatiquement créé lorsqu'il n'existe pas.

La sortie d'erreur standard peut être également redirigée vers un autre fichier. Cette fois-ci, il faudra préciser le numéro du canal (qui peut être omis pour les canaux 0 et 1) :

```
$ ls -R / 2> fic_erreurs
$ ls -R / 2>> fic_erreurs
```

7.4. Exemples de redirections

Redirection de 2 sorties vers 2 fichiers :

```
$ ls -R / >> fic_ok 2>> fic_nok
```

Redirection des 2 sorties vers un fichier unique :

```
$ ls -R / >> fic_log 2>&1
```

Redirection de **stderr** vers un "puits sans fond" (`/dev/null`) :

```
$ ls -R / 2>> /dev/null
```

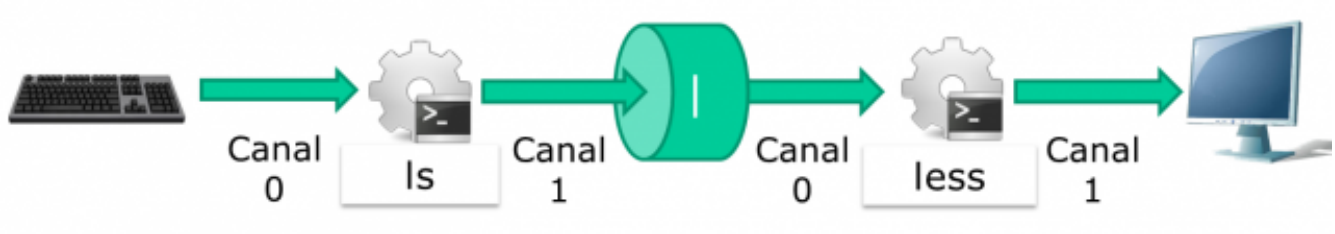
Lorsque les 2 flux de sortie sont redirigés, aucune information n'est affichée à l'écran. Pour utiliser à la fois la redirection de sortie et conserver l'affichage, il faudra utiliser la commande `tee`.

7.5. Les tubes (pipe)

Un **tube** (**pipe** en anglais) est un mécanisme permettant de relier la sortie standard d'une première

commande vers l'entrée standard d'une seconde.

Cette communication est monodirectionnelle et se fait grâce au symbole `|`. Le symbole pipe "|" est obtenu en appuyant simultanément sur les touches `AltGR+6`.



Toutes les données envoyées par la commande à gauche du tube à travers le canal de sortie standard sont envoyées au canal d'entrée standard de la commande placée à droite.

Les commandes particulièrement utilisées après un pipe sont des filtres.

- Exemples :

```
# N'afficher que le début :
$ ls -lia / | head

# N'afficher que la fin :
$ ls -lia / | tail

# Trier le résultat
$ ls -lia / | sort

# Compter le nombre de mots / caractères
$ ls -lia / | wc

# Chercher une chaîne de caractères dans le résultat :
$ ls -lia / | grep fichier
```

Chapitre 8. Points particuliers

8.1. La commande tee

La commande `tee` permet de rediriger la sortie standard d'une commande vers un fichier tout en maintenant l'affichage à l'écran.

Elle est combinée avec le pipe "`|`" pour recevoir en entrée la sortie de la commande à rediriger.

```
$ ls -lia / | tee fic
$ cat fic
```

L'option `-a` permet d'ajouter au fichier au lieu de l'écraser.

8.2. Les commandes alias et unalias

Utiliser les **alias** est un moyen pour demander au shell de se souvenir d'une commande particulière avec ses options et lui donner un nom.

Par exemple :

```
$ ll
```

remplacera la commande :

```
$ ls -l
```

La commande `alias` liste les alias de la session en cours. Des alias sont positionnés par défaut sur les distributions Linux. Ici, les alias d'un serveur CentOS :

```
$ alias
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias vi='vim'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
```

Les alias ne sont définis que de façon temporaire, le temps de la session utilisateur.

Pour une utilisation permanente, il faut les créer dans le fichier :

- `.bashrc` du répertoire de connexion de l'utilisateur ;

- `/etc/profile.d/alias.sh` pour tous les utilisateurs.

Une attention particulière doit être portée lors de l'usage d'alias qui peuvent potentiellement s'avérer dangereux ! Par exemple, un alias mis en place à l'insu de l'administrateur :

```
alias cd='rm -Rf'
```

La commande `unalias` permet de supprimer les alias.

```
$ unalias ll

# Pour supprimer tous les alias :
$ unalias -a
```

8.3. Alias et fonctions utiles

- Alias `grep`

Colorise le résultat de la commande `grep` :

```
alias grep='grep --color=auto'
```

- Fonction `mcd`

Il est fréquent de créer un dossier puis de se déplacer dedans :

```
mcd() { mkdir -p "$1"; cd "$1"; }
```

- Fonction `cls`

Se déplacer dans un dossier et lister son contenu :

```
cls() { cd "$1"; ls; }
```

- Fonction `backup`

Créer une copie de sauvegarde d'un fichier :

```
backup() { cp "$1"{,.bak}; }
```

- Fonction `extract`

Extrait tout type d'archive :

```
extract () {
  if [ -f $1 ] ; then
    case $1 in
      *.tar.bz2) tar xjf $1 ;;
      *.tar.gz) tar xzf $1 ;;
      *.bz2) bunzip2 $1 ;;
      *.rar) unrar e $1 ;;
      *.gz) gunzip $1 ;;
      *.tar) tar xf $1 ;;
      *.tbz2) tar xjf $1 ;;
      *.tgz) tar xzf $1 ;;
      *.zip) unzip $1 ;;
      *.Z) uncompress $1 ;;
      *.7z) 7z x $1 ;;
      *)
        echo "'$1' cannot be extracted via extract()" ;;
    esac
  else
    echo "'$1' is not a valid file"
  fi
}
```

- Alias `cmount`

```
alias cmount="mount | column -t"

[root]# cmount
/dev/simfs  on  /                                       type  simfs
(rw,relatime,usrquota,grpquota)
proc        on  /proc                                   type  proc
(rw,relatime)
sysfs       on  /sys                                    type  sysfs
(rw,relatime)
none        on  /dev                                    type  devtmpfs
(rw,relatime,mode=755)
none        on  /dev/pts                                type  devpts
(rw,relatime,mode=600,ptmxmode=000)
none        on  /dev/shm                                type  tmpfs
(rw,relatime)
none        on  /proc/sys/fs/binfmt_misc                type  binfmt_misc
(rw,relatime)
```

8.4. Le caractère ;

Le caractère `;` chaîne les commandes.

Les commandes s'exécuteront toutes séquentiellement dans l'ordre de saisie une fois que l'utilisateur aura appuyé sur `[ENTREE]`.

```
$ ls /; cd /home; ls -lia; cd /
```