# Aide-mémoire des commandes Linux

## Table des matières



Caractères spéciaux :

"/" : séparation entre les noms de répertoires / fichiers

".." : le répertoire du niveau juste au dessus

"." : le répertoire courant

"~marie" : le répertoire par défaut de chez marie

"~" : le répertoire par défaut chez moi ($HOME)

L'utilisation de la touche "Ctrl-C" n'est pas un « copié collé », mais a pour but de d'annuler une commande en cours (*Cancel*).

Expressions régulières :

"*" : peut être remplacé par n'importe quoi

ex : "f*" : tout ce qui commence par « f »

"f*.jpg" : tout ce qui commence par « f » et qui termine par « .jpg »

Commandes rapides (pour more/less/man):

"/" : rechercher "n" : aller au suivant

"espace", "entrée" : aller à la page/ligne suivante

"q" : quitter

Ligne de commande :

";" : remplace la touche « entrée » pour exécuter plusieurs commandes sur une seule ligne (ex: « mkdir rep ; mkdir rep/rep1 »)

"cmd1 | cmd2" : « pipe » : tube de communication qui exécute la commande « cmd2 » sur le résultat de « cmd1 » (ex: « ls -lt | less »)

"cmd &" : lance la commande « cmd » et redonne la main aussitôt. « jobs » permet de lister les programmes lancés en arrière plan. « bg » et « fg » permettent de basculer en arrière ou avant plan.

"cmd > fic" : rediriger le résultat de la commande « cmd » directement dans un nouveau fichier « fic » (ex: « sort fic1 > tri1 »). (« >> » redirige dans un fichier mais en ajoutant à la suite ; « >! » écrase le fichier s'il existait déjà).

"tabulation" : remplace le reste de la ligne par ce qui existe ou s'il existe plusieurs possibilité les montre (avec une 2ème tabulation)

Aide :

"cmd -h" ou "cmd --help " : Affiche l'aide de la commande

Vocabulaire :

« prompt/console/xterm/shell » : fenêtre où sont tapés les commandes

« input/entrée standard » : ce qui est tapé par l'utilisateur

« output/sortie standard » : ce qui est affiché par les programmes

# Répertoires

## pwd

Affiche le chemin absolu du répertoire courant (*Print Working Directory*).

```
$ pwd
/home/nicolas
```

## cd [répertoire]

Change de répertoire (*Change Directory*). Va dans `répertoire` ou dans le répertoire de l'utilisateur s'il n'y a pas d'argument.
Si "`-`" est indiqué en argument, déplace dans le répertoire précédent (temporellement).

```
$ pwd
/home/nicolas
$ cd /var/tmp
$ pwd
/var/tmp
$ cd ..
$ pwd
/var
$ cd
$ pwd
/home/nicolas
$ cd -
/var
$ pwd
/var
```

## ls [fichier ...]

Liste le contenu des répertoires ou le nom des fichiers passés en arguments (liste le répertoire courant si pas d'argument).
`-l` : affichage détaillé (*long*)
`-a` : affichage aussi des fichiers cachés dont le nom commence par un point (*all*)
`-t` : trie l'affichage suivant la date de modification des fichiers (*time*)
`-r` : inverse le tri d'affichage (*reverse*)
`-R` : affichage du contenu de tous les sous répertoires (*recursive*)

```
$ ls
fic1  fic2  rep1
$ ls -a
.  ..  .bash_history  .bash_profile  .bashrc  .viminfo  fic1  fic2
rep1
$ ls -l
total 8
```

```
-rw-r--r--    1 nicolas  users           0 Dec  6 11:48 fic1
-rw-r--r--    1 nicolas  users         868 Dec  6 11:48 fic2
drwxr-xr-x    3 nicolas  users        4096 Dec  6 11:48 rep1
$ ls -R
.:
fic1  fic2  rep1

./rep1:
ficA  repA

./rep1/repA:
$ ls -ltr
total 8
-rw-r--r--    1 nicolas  users           0 Dec  6 11:48 fic1
drwxr-xr-x    3 nicolas  users        4096 Dec  6 11:48 rep1
-rw-r--r--    1 nicolas  users         868 Dec  6 11:48 fic2
$ ls -l rep1
total 4
-rw-r--r--    1 nicolas  users           0 Dec  6 11:48 ficA
drwxr-xr-x    2 nicolas  users        4096 Dec  6 11:42 repA
```

## mkdir <répertoire ...>

Créé les répertoires (*MaKe DIRectory*) passés en arguments.

```
$ mkdir rep1
$ ls -l
total 4
drwxr-xr-x    2 nicolas  users        4096 Dec  6 11:41 rep1
$ mkdir rep1/repA rep2
$ ls -lR
.:
total 8
drwxr-xr-x    3 nicolas  users        4096 Dec  6 11:42 rep1
drwxr-xr-x    2 nicolas  users        4096 Dec  6 11:42 rep2

./rep1:
total 4
drwxr-xr-x    2 nicolas  users        4096 Dec  6 11:42 repA

./rep1/repA:
total 0

./rep2:
total 0
```

## rmdir <répertoire ...>

Supprime les répertoires (*ReMove DIRectory*) passés en arguments s'ils sont vides.

```
$ rmdir rep1 rep2
rmdir: `rep1': Directory not empty
$ ls -l
total 4
drwxr-xr-x    3 nicolas  users        4096 Dec  6 11:42 rep1
```

# Fichiers

## cp <source ...> <destination>

Copie (*CoPy*) les fichiers `source` vers `destination`.
`-i` : demande confirmation avant écrasement (interactive)
`-f` : écrase sans demander confirmation (force)
`-R` ou `-r` : copie aussi les répertoires (recursive)

```
$ ls -l
total 8
-rw-r--r--    1 nicolas  users           0 Dec  6 11:48 fic1
-rw-r--r--    1 nicolas  users         868 Dec  6 11:48 fic2
drwxr-xr-x    3 nicolas  users        4096 Dec  6 11:48 rep1
$ cp fic1 fic3
$ cp fic2 fic3
$ cp -i fic2 fic3
cp: overwrite `fic3'? o
$ cp rep1 rep2
cp: omitting directory `rep1'
$ cp -r rep1 rep2
$ cp rep1/fic11 .
$ ls -l
total 16
-rw-r--r--    1 nicolas  users           0 Dec  6 11:48 fic1
-rw-r--r--    1 nicolas  users           0 Dec  6 11:48 fic11
-rw-r--r--    1 nicolas  users         868 Dec  6 11:48 fic2
-rw-r--r--    1 nicolas  users         868 Dec  6 14:08 fic3
drwxr-xr-x    3 nicolas  users        4096 Dec  6 11:48 rep1
```

Attention : sans l'option `-R` (ou `-r`), la commande cp ne pourra pas copier les répertoires ; il est nécessaire qu'elle travaille en "récursif" pour parcourir l'arborescence de fichiers sous le répertoire, et ainsi pouvoir copier les fichiers sous jacents.

Le '.' signifie « ici, et sous le même nom que celui d'origine ».

## mv <source ...> <destination>

Renomme/déplace (*MoVe*) les fichiers `source` vers `destination`.
`-i` : demande confirmation avant écrasement (interactive)
`-f` : écrase sans demander confirmation (force)

Il s'agit d'un déplacement lorsque la destination est un répertoire déjà existant : le déplacement est fait dedans.

```
$ ls
fic1  fic2  fic3  rep1  rep2
$ ls rep2
ficA  repA
```

```
$ mv fic* rep2
$ ls
rep1  rep2
$ ls rep2
fic1  fic2  fic3  ficA  repA
$ mv rep1 rep2
$ ls
rep2
$ ls rep2
fic1  fic2  fic3  ficA  rep1  repA
```

## rm <fichier ...>

Supprime (*ReMove*) les fichiers passés en arguments.
`-i` : demande confirmation avant suppression (interactive)
`-f` : supprime sans demander confirmation (force)
`-R` : supprime aussi les répertoires (recursive)

```
$ ls -R .
.:
rep2

./rep2:
fic1  fic2  fic3  ficA  rep1  repA

./rep2/rep1:
ficA  repA

./rep2/rep1/repA:

./rep2/repA:
$ rm rep2/fic1
$ ls rep2
fic2  fic3  ficA  rep1  repA
$ rm rep2/rep1
rm: `rep2/rep1' is a directory
$ rm -r rep2/rep1
$ ls rep2
fic2  fic3  ficA  repA
```

## find <chemin(s)> <critère(s)> <action(s)>

Recherche de fichiers multi-critères :
- recherche récursive dans le(s) répertoire(s) indiqué(s) (`chemin(s)`)
- les principaux critères (`critère(s)`) sont :
  `-name '<motif>'`
  `-size <[+|-]taille>`
  `-mtime <[+|-]date>`
  `-user <nom|UID>`
  `-newer <fichier référence>`

```
$ find /home -name 'rep*'
```

```
/home/nicolas/rep1
/home/nicolas/rep2
/home/nicolas/rep2/rep1
```

## locate

Recherche de fichiers suivant leur nom ; cette commande est plus rapide que la commande `find` car elle utilise une base de données des fichiers présents sur le système (voir la commande `updatedb`). Cependant, si la base de données des fichiers n'est pas à jour, le résultat de la recherche n'affiche pas tous les fichiers existants.

```
$ locate ficA.txt
/home/nicolas/rep2/ficA
```

# Traitement de fichiers et filtres

*Pour cet excercice, créez un fichier nommé « fictexte » contenant le texte ci dessous de Savinien*

## cat <fichier ...>

Affiche le contenu des fichiers texte passés en arguments.

```
$ cat fictexte
Je vis que la Terre ayant besoin de la lumière,
de la chaleur, et de l'influence de ce grand feu,
elle se tourne autour de lui pour recevoir
également en toutes ses parties cette vertu qui
la conserve.
                Savinien de CYRANO DE BERGERAC
```

L'utilisation de la commande « cat » avec plusieurs fichiers les affichera tous à la suite (concaténation).

```
$ cat fic1 fic2 > fic1+2
```

## tac <fichier ...>

Affiche le contenu inversé (de la dernière ligne à la première) des fichiers texte passés en arguments.

```
$ tac fictexte
                Savinien de CYRANO DE BERGERAC
la conserve.
également en toutes ses parties cette vertu qui
elle se tourne autour de lui pour recevoir
de la chaleur, et de l'influence de ce grand feu,
Je vis que la Terre ayant besoin de la lumière,
```

## nl <fichier ...>

Affiche le contenu des fichiers texte passés en arguments en numérotant les lignes.

```
$ nl fictexte
     1  Je vis que la Terre ayant besoin de la lumière,
     2  de la chaleur, et de l'influence de ce grand feu,
     3  elle se tourne autour de lui pour recevoir
     4  également en toutes ses parties cette vertu qui
     5  la conserve.
     6                  Savinien de CYRANO DE BERGERAC
```

## head -<num> <fichier ...>

Affiche le contenu des fichiers texte passés en arguments en limitant le nombre de lignes aux num premières.

```
$ head -3 fictexte
     Je vis que la Terre ayant besoin de la lumière,
     de la chaleur, et de l'influence de ce grand feu,
     elle se tourne autour de lui pour recevoir
```

## tail -/+<num> <fichier ...>

Affiche le contenu des fichiers texte passés en arguments en limitant le nombre de lignes aux num dernières. Avec +, affiche à partir de la nième ligne.

```
$ tail -3 fictexte
     également en toutes ses parties cette vertu qui
     la conserve.
                       Savinien de CYRANO DE BERGERAC
```

## more <fichier ...>

Affiche page par page le contenu des fichiers texte passés en arguments.

Pour afficher le contenu d'un fichier :

```
$ more ~/.bashrc
export prompt="%S %m %/ %s%% "
export path = ( . /usr/local/bin /usr/local/lang /usr/local/X11/bin /usr/ucb
/usr/bin /usr/etc /usr/openwin/bin /usr/local/wp/wpbin $path)
...
#setenv VISUAL vi
#
# setting your pager, uncomment one of these
#setenv PAGER less
#setenv PAGER more

--More--(24%)
```

Dans un tube, pour visualiser le résultat d'une commande :

```
$ ps -ef | more
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Oct23 ?        00:00:19 init [3]
root         2     1  0 Oct23 ?        00:00:00 [keventd]
....
root       484     1  0 Oct23 ?        00:00:03 /usr/sbin/sshd
root       493     1  0 Oct23 ?        00:16:07 syslogd -m 0
root       497     1  0 Oct23 ?        00:00:00 klogd -x
rpc        507     1  0 Oct23 ?        00:00:00 portmap
rpcuser    526     1  0 Oct23 ?        00:00:00 rpc.statd
ldap       576     1  0 Oct23 ?        04:21:33 /usr/sbin/slapd -u ldap
-h ldap:
```

```
--More--
```

## less <fichier ...>

Affiche page par page le contenu des fichiers texte passés en arguments (similaire à « more »).

```
$ less ~/.tcshrc

Quelques commandes spécifiques à less :
« N » pour aller au précédent après une recherche
« < », « > » pour aller au début ou la fin du fichier
L'utilisation des flèches est possible.
```

## wc <fichier ...>

Affiche le nombre de lignes, de mots et de caractères (*Word Count*) contenus dans les fichiers passés en arguments.
`-l` : affiche uniquement le nombre de lignes (line)
`-w` : affiche uniquement le nombre de mots (word)
`-c` : affiche uniquement le nombre de caractères (character)

```
$ wc rep1/fic*
    569    2805   19935 /rep1/ficA
    594    2957   20910 /rep1/ficB
$ ls | wc -l
      5
```

## join <fichier1> <fichier2>

Effectue une jointure (dans le sens d'une base de données relationnelle) entre deux fichiers texte.

```
$ cat fic1
1       nicolas
3       franck
5       gerard
8      stef
9      willy
$ cat fic2
1       formateur
3       directeur
5       patron
8       commercial
9       secretaire
$ join fic1 fic2
1 nicolas formateur
3 gerard directeur
5 franck patron
8 alain commercial
9 soraya secretaire
```

## paste <fichier1> <fichier2> <...>

Fusionne ligne par ligne les fichiers passés en argument.

```
$ cat fic1
1       nicolas
5       franck
3       gerard
12      stef
$ cat fic2
5       patron
8       commercial
3       directeur
12      administrateur
$ paste fic1 fic2
1       nicolas 5       patron
5       franck  8       commercial
3       gerard  3       directeur
12      stef    12      administrateur
```

## diff <fichier1> <fichier2>

Compare ligne par ligne les fichiers passés en argument et affiche les différences.

```
$ cat fic1
1       nicolas
5       franck
3       gerard
$ cat fic11
1       nicolas
5       franck
2       gerard
12      stef
$ diff fic1 fic11
3c3,4
< 3   gerard
---
> 2   gerard
> 12   stef
```

## gzip <fichier>

Compresse le fichier `fichier` au format GNU Zip ; par défaut, le fichier compressé est nommé `fichier.gz` et l'original est supprimé.
Voir la commande `gunzip` pour décompresser le fichier généré.

```
$ ls -l
total 1144
-rw-r--r--    1 nicolas  users     1166532 Dec  6 16:21 fichier
$ gzip fichier
$ ls -l
total 140
-rw-r--r--    1 nicolas  users      137644 Dec  6 16:21 fichier.gz

$ gunzip fichier
$ ls -l
total 1144
```

```
-rw-r--r--    1 nicolas  users     1166532 Dec  6 16:21 fichier
```

Idem pour « zip » à la place de gzip (donnera une extension .zip, et unzip pour décompresser)

## grep <regexp> [fichier ...]

Affiche uniquement les lignes, des fichiers passés en argument, correspondantes à l'expression rationnelle (ou expression régulière) regexp.
`-v` : inverse le résultat de la commande (affiche seulement les lignes ne correspondant pas à regexp)
`-c` : retourne le nombre de correspondances
`-n` : affiche les numéros des lignes correspondantes
`-l` : affiche les noms des fichiers contenant des lignes correpondant à regexp
`-i` : ne tient pas compte de la casse des caractères

```
$ grep BERGERAC fic*
/home/nicolas/fictexte
```

## awk <instr ...> [fichier ...]

Applique les instructions awk sur les fichiers passés en arguments.
`-f instr.awk` : utilise les instructions awk contenues dans le fichier `instr.awk`

```
$ awk '{print "ligne "NR" : le nom est "$2}' fic1
ligne 1 : le nom est nicolas
ligne 2 : le nom est franck
ligne 3 : le nom est gerard
```

## cut -d<délimiteur> -f<champ(s)> [fichier]

Affiche les champs spécifiés avec l'option `-f` et séparés par le délimiteur indiqué après l'option `-d`, ou affiche les colonnes de caractères indiquées après l'option `-c`.

Pour afficher la 3ème et 6ème colonnes du fichier `/etc/passwd` :

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
nobody:x:65534:65534:nobody:/home:/bin/sh

$ cut -d":" -f3,6 /etc/passwd
0:/root
1:/usr/sbin
2:/bin
3:/dev
65534:/home
```

### `sort -rmf [fichier]`

Trie les ligne d'un fichier texte. L'option '-m' mélange les fichiers en les triant (chaque fichier d'entrée doit déjà être trié individuellement). L'option '-r' trie dans l'ordre inverse. L'option '-f' considère les minuscules comme des majuscules.

```
$ cat fic1
1       nicolas
5       franck
3       gerard
12      stef
$ sort fic1
1       nicolas
3       gerard
5       franck
12      stef
```

# Documentation

---

### `man [section] <argument>`

Affiche la page de manuel électronique dont le nom est `argument` (se trouvant dans la section du manuel éventuellement spécifié).
`-k` : retourne le nom des pages de manuel contenant `argument`

```
$ man grep

NAME
       grep, egrep, fgrep - print lines matching a pattern

SYNOPSIS
       grep [options] PATTERN [FILE...]
       grep [options] [-e PATTERN | -f FILE] [FILE...]

DESCRIPTION
       Grep  searches the named input FILEs (or standard input if no
files are named, or the file name - is given) for lines containing a
match to the given PATTERN.  By default, grep prints the matching
lines.

       In addition, two variant programs egrep and fgrep are available.
Egrep is the same as grep -E.  Fgrep is the same as grep -F.

OPTIONS
       -A NUM, --after-context=NUM
              Print NUM  lines  of  trailing  context  after  matching
```

```
lines. Places a line containing -- between contiguous groups of
matches.

       -a, --text
              Process a binary file as if it were text; this is
equivalent  to the --binary-files=text option.

       -B NUM, --before-context=NUM
              Print NUM lines of leading context before matching
lines. Places a line containing -- between contiguous groups of
matches.

       -C NUM, --context=NUM
              Print  NUM lines of output context.  Places a line
containing -- between contiguous groups of matches.
```
lines 1-39

La navigation dans une page de manuel électronique se fait de la même manière que dans « more/less ».

# Droits

## chmod <mode> <fichier ...>

Modifie les droits d'accès (*CHange MODe*) aux fichiers passés en arguments suivant le mode (notation symbolique ou octale).
-R : applique les modifications à toute l'arborescence (recursive).

```
$ ls -l
total 1148
-rw-r--r--    1 nicolas  users     1166532 Dec  6 16:21 fichier
drwxr-xr-x    2 nicolas  users        4096 Dec  6 16:52 rep
$ chmod ug+x fichier
$ chmod 754 rep
$ ls -l
total 1148
-rwxr-xr--    1 nicolas  users     1166532 Dec  6 16:21 fichier
drwxr-xr--    2 nicolas  users        4096 Dec  6 16:52 rep

'd' : signifie qu'il s'agit d'un répertoire
les « modes » d'accessibilité sont :
       'u' au propriétaire (user)
       'g' au groupe (group)
       'o' à tout les autres (others)
les autorisations sont :
       'r' lire (read)
       'w' modifier et supprimer un fichier, ou s'il s'agit d'un
       répertoire : création et suppression des fichiers dans le
       répertoire (write)
       'x' éxécuter le fichier, ou s'il s'agit d'un répertoire : y
```

rentrer via « cd » (*execute*)

# Gestion des comptes utilisateur

## passwd

Change le mot de passe du compte (mot de passe de l'utilisateur connecté).

# Communication

## ssh [login@]serveur

Se sonnecte sur un serveur distant sous le mode « Secure SHell ». Demande un mot de passe, mais permet de se connecter sans taper de mot de passe si un système de « clés » a été définie. Si c'est la première fois que vous vous connectez par ssh sur une machine, vous verrez un message qui vérifie qu'il s'agit de la bonne machine. Il suffit de répondre 'yes' pour continuer. Le login n'est pas obligatoire s'il s'agit du même que celui déjà en cours. La commande « ftp » est identique, mais non sécurisée, et les commandes éxécutées à distance sont spécifiques.

Une fois connecté, toutes les commandes tapées seront éxécutées sur le serveur.

« exit » pour quitter.

## scp [login@]serveur

Transfert depuis/sur un serveur distant sous le mode « Secure SHell » (« *Secure Copy* »).

Envoie du fichier fic.txt vers le serveur dans le répertoire « rep » et sous le nom « coincoin.txt » :

```
$ scp fic.txt login@serveur:rep/coincoin.txt
```

Rappatriement du fichier depuis le serveur dans le répertoire « rep » et sous le nom « coincoin.txt » vers le fichier nommé « fic.txt » :

```
$ scp login@serveur:rep/coincoin.txt fic.txt
```

# Divers

### `echo [argument ...]`

Affiche sur la sortie standard les chaînes de caractères passées en arguments, séparées par un espace.

### `exit`

Quitte le shell en cours.

### `logout`

Déconnecte l'utilisateur.

### `alias [nom='cmd']`

Affiche les alias définis dans l'environnement shell actuel (pas d'argument) ou en défini un nouveau. Si aucun alias n'est donné, affiche tous les alias déjà définis.

### `unalias <nom>`

Supprime l'alias de l'environnement shell actuel.

---

source : http://www.epons.org/commandes-base-linux.php

# Index lexical