

# Petit catalogue de commandes Unix

Juliusz Chroboczek

15 Septembre 2010

## Introduction

Ce document contient une explication sommaire d'un petit nombre de commandes Unix. Les commandes sont groupées par fonctionnalité, et la présentation ne suit donc pas un ordre pédagogique.

## 1 Syntaxe générale des commandes

Les commandes Unix ont en général (mais pas toujours) la syntaxe suivante :

*nom* [*options*] [--] [*arguments*]

où :

- le *nom* est le nom de la commande, par exemple `ls` ;
- une *option* est typiquement un signe moins (« - ») suivi d'une lettre ; il y a zéro, une ou plusieurs options séparées par des espaces ;
- l'option `--` signifie la fin des options ;
- les *arguments* sont une liste de zéro, un ou plusieurs arguments séparés par des espaces.

Par exemple, dans la ligne de commande suivante :

```
ls -a /usr/bin/ /usr/local/bin/
```

le nom de la commande est « `ls` », il y a une seule option « `-a` », et deux arguments « `/usr/bin/` » et « `/usr/local/bin/` ».

## 2 Quelques commandes utiles

### 2.1 Commandes d'information

`man` : affiche le manuel de la commande donnée en argument. Options utiles :

- s *n* spécifie la section du manuel à consulter ;
- k « à propos », donne une liste de commandes ayant le mot donné en argument dans leur description.

`:` : ne fait rien.

`echo` : affiche ses arguments.

## 2.2 Commandes élémentaires de manipulation de fichiers

`cat` : copie les fichiers donnés en arguments (ou, à défaut, son entrée) sur sa sortie.

`more` : visualisation interactive des fichiers donnés en arguments (ou, à défaut, son entrée). Tapez `h` pour avoir de l'aide.

`less` : version améliorée de `more`.

`ls` : liste les fichiers des répertoires donnés en arguments, ou le répertoire courant. Options utiles :

- a liste tous les fichiers ;
- F affiche le type des fichiers ;
- l affichage détaillé (long) ;
- s affiche l'espace occupé par les fichiers.

`cp` : copie de fichiers : copie les premiers arguments à destination du dernier (qui peut être un fichier ou un répertoire). Options utiles :

- i interactif (demande confirmation pour chaque fichier) ;
- f force (ne demande jamais de confirmation) ;
- r copie dite *récursive* : copie un répertoire et tous les fichiers qu'il contient.

`mv` : renommage et déplacement de fichiers. La syntaxe est la même que celle de `cp`.

`ln` : création de liens symboliques.

- s crée un lien symbolique plutôt qu'un lien dur.

Les autres options sont les mêmes que celles de `cp`.

`rm` : supprime les fichiers donnés en arguments *définitivement*. Options utiles :

- i interactif ;
- f force ;
- r supprime un répertoire et tous ses répertoires.

`gzip` : compresse les fichiers donnés en arguments (ou à défaut l'entrée standard sur la sortie standard). Options utiles :

- c résultats sur la sortie standard ;
- d décompresse au lieu de compresser.

La commande `gunzip` est équivalente à `gzip -d`. La commande `zcat` (ou, sur certains systèmes, `gzcat`) est équivalente à `gzip -c -d`.

## 2.3 Commandes de manipulation de répertoires

`pwd` : affiche le répertoire courant.

`cd` : change le répertoire courant. Sans arguments, change le répertoire courant vers le répertoire *home*.

`mkdir` : crée le répertoire spécifié en argument.

`rmdir` : supprime le répertoire spécifié en argument (qui doit être vide).

## 2.4 La commande tar

La commande `tar` (*tape archive*) sert à créer des *archives*, c'est à dire des collections de fichiers. Comme son nom l'indique, elle sert pour l'archivage, mais peut aussi être utile lorsqu'il est nécessaire d'envoyer une collection de fichiers en bloc (par exemple par courrier électronique ou

lors d'un transfert de fichiers). Les applications sont le plus souvent distribuées sous forme d'une archive tar compressée par gzip.

Pour des raisons historiques, la commande tar n'obéit pas aux conventions habituelles de passage des arguments. Une invocation de tar prend la forme quelque peu baroque suivante :

`tar aooo arguments`

où *a* est l'action à effectuer, *ooo* sont des options, et les *arguments* sont les arguments des options suivis des fichiers à archiver.

Par exemple, l'invocation

`tar cvf test.tar test.c test.h`

invoque l'action *c* (création) avec les options *v* (prolix) et *f* (nom du fichier archive). L'option *v* ne prend pas d'options, tandis que *f* en prend une, soit `test.tar`. Les fichiers à archiver sont `test.c` et `test.h`.

Actions :

- c* créer une archive ;
- t* lister le contenu d'une archive ;
- x* extraire les fichiers d'une archive.

Options :

- f* nom du fichier archive ; prend un argument ; le nom – spécifie la sortie standard ;
- v* opération prolix ; par défaut, tar est silencieux ;
- z* compresse ou décompresse une archive à la volée.

## 2.5 Commandes de manipulation de méta-données

`chown` : change le propriétaire des fichiers passés en arguments. Le nouveau propriétaire est donné par le premier argument.

`chmod` : change les permissions des fichiers passés en arguments. Le premier argument est une spécification des changements à effectuer aux permissions, de la forme

$u-p$

ou

$u+p$

où *u* est l'utilisateur concerné par le changement, soit *u* (le propriétaire), *g* (son groupe) ou *o* (les autres) ; *p* sont les permissions concernées, *r*, *w* ou *x*.

`touch` : affecte l'heure courante au temps de dernière modification du fichier donné en paramètre. Si celui-ci n'existe pas, il est créé.

`umask` : change le « masque de l'utilisateur », qui spécifie les permissions qui sont *omisées* dans les fichiers nouvellement créés. Le nouveau masque est donné en octal comme premier paramètre.

## 2.6 Commandes de traitement de texte

**wc** : compte les caractères, mots et lignes dans les fichiers texte passés en arguments ou, à défaut, dans l'entrée standard.

**grep** : recherche de l'expression régulière passée en premier argument dans les fichiers passés en arguments ou, à défaut, dans l'entrée. Options :

- f : recherche une chaîne fixe plutôt qu'une expression régulière ;
- l : liste les fichiers plutôt que les occurrences ;
- i : ignore les distinctions entre majuscules et minuscules.
- r : recherche récursivement dans des répertoires ;

**fgrep** : équivalent à **grep -f**.

**zgrep** : recherche dans des fichiers compressés avec **gzip**.

## 2.7 Commandes de gestion des processus

**jobs** : donne la liste des tâches de fond du shell courant. Options :

- l : donne aussi les numéros de processus.

**ps** : donne la liste de tous les processus. Cette commande utilise les mêmes conventions de passage d'options que **tar** (i.e. elle n'utilise pas de tiret pour indiquer les options). Options<sup>1</sup> :

- a : affiche les processus de tous les utilisateurs ;
- l : affichage détaillé ;
- gx : affiche tous les processus, même ceux qui ne sont pas « intéressants ».

**kill** : cause la terminaison d'un processus ; on lui passe en argument le numéro du processus à tuer (obtenu à l'aide de **ps**). Options :

- 1 : demande la terminaison inconditionnelle ;
- 9 : cause la terminaison inconditionnelle, sans donner au processus aucune chance de survivre ou de sauvegarder les fichiers en cours.

**top** : interface interactive à **ps** et **kill**. Tapez « h » pour avoir de l'aide.

**who, w** : donne la liste des utilisateurs connectés au système, ainsi que les processus qu'ils exécutent.

---

1. En fait, il existe deux versions de la commande **ps**. Nous décrivons ici la version dite *BSD* de cette commande ; la version *SV* a une syntaxe différente.