

Memento ligne de commandes, shell bash

Thème architectures matérielles et systèmes d'exploitation

Première NSI, Lycée du Parc

Table des matières

Crédits	1
1 Un premier exemple de commande	2
2 Naviguer dans l'arborescence du système de fichiers	3
3 Copier, supprimer, déplacer un fichier	4
4 Expansion des noms de fichiers et <i>globbing</i>	5
5 Gestion des droits sur les fichiers	6

Crédits

Memento directement inspiré des livres [La ligne de commande par l'exemple](#) de Vincent Fourmond et [Parlez-vous Shell ?](#) de Thomas Hugel.

Dans ce memento, nous présentons des commandes du *shell* **BASH** sous licence libre, qui est le *shell* par défaut sur la plupart des distributions du système d'exploitation libre **Linux**. On distinguera parfois fichiers et répertoires mais on rappelle que les répertoires sont juste des fichiers spéciaux, qui contiennent d'autres fichiers. Un memento en ligne est disponible sur <https://juliend.github.io/linux-cheatsheet/>.

Conseils pratiques : pour faciliter la saisie des commandes avec le clavier, **BASH** offre quelques raccourcis clavier bien pratiques :

- la touche de tabulation permet d'appeler la complétion automatique qui propose de compléter la commande avec les choix possibles (fichiers ou commandes existants). Par exemple si on saisit `pw`, l'appui sur la touche de tabulation nous propose plusieurs commandes commençant par ce préfixe :

```
junier@fredportable:~$ pw
pwck      pwconv   pwd      pwdx     pwgen    pwunconv
```

- les flèches de direction Haut et Bas permettent de naviguer dans l'historique des commandes.
- la plupart des commandes du *shell* sont dotées d'une documentation accessible depuis l'interpréteur avec la commande `man`. Par exemple pour afficher l'aide de la commande `ls`, on écrira `man ls`.

1 Un premier exemple de commande

Une commande *shell* est constituée du nom de la commande suivi d'un ou plusieurs arguments. Des options précédées peuvent modifier le comportement de la commande :

```
nom_commande -option1 -option2 ... arg1 arg2 arg3 ...
```

Ainsi, la commande `ls` permet d'afficher des informations sur répertoire ou un fichier :

- Sans argument, ni option `ls` liste le contenu du répertoire courant :

```
junier@fredportable:~/sandbox$ ls
fichier1 fichier2 fichier3 fichier4 rep1 rep2
```

- Avec l'option `-l` elle affiche des informations détaillées sur chacun des fichiers contenus dans le répertoire :

```
junier@fredportable:~/sandbox$ ls -l
total 8
-rw-rw-r-- 1 junier junier  0 août  16 21:43 fichier1
.....
drwxrwxr-x 2 junier junier 4096 août  16 21:44 rep1
drwxrwxr-x 2 junier junier 4096 août  16 21:44 rep2
```

- L'option `-a` affiche les fichiers (ou répertoires cachés) et l'option `-h` convertit les tailles de fichiers (en octets par défaut) en des multiples plus lisibles. On peut écrire `ls -l -a -h` ou regrouper les options `ls -lah`. L'ordre des options n'a pas d'importance :

```
junier@fredportable:~/sandbox$ ls -lah
total 16K
drwxrwxr-x  4 junier junier 4,0K août  16 21:49 .
drwxr-xr-x 50 junier junier 4,0K août  16 21:43 ..
-rw-rw-r--  1 junier junier  0 août  16 21:49 .cache_cache
-rw-rw-r--  1 junier junier  0 août  16 21:43 fichier1
.....
drwxrwxr-x  2 junier junier 4,0K août  16 21:44 rep1
drwxrwxr-x  2 junier junier 4,0K août  16 22:10 rep2
```

- Si on passe un répertoire en argument à la commande, elle affiche son contenu et si on souhaite une information globale sur le répertoire, on passe l'option `-d` :

```
junier@fredportable:~/sandbox$ ls -l rep1
total 0
-rw-rw-r-- 1 junier junier 0 août  16 21:44 photo1.jpg
-rw-rw-r-- 1 junier junier 0 août  16 21:44 photo2.jpg
junier@fredportable:~/sandbox$ ls -ld rep1
drwxrwxr-x 2 junier junier 4096 août  16 21:44 rep1
```

- On peut afficher l'aide détaillée de `ls` avec l'option longue (double tiret) `--help`

```
junier@fredportable:~/sandbox$ ls --help
Utilisation : ls [OPTION]... [FICHER]...
Afficher des renseignements sur les FICHIERS (du répertoire actuel par défaut).
```

Remarque : L'aide s'obtient avec l'option `-help` ou avec `man nom_commande`.

2 Naviguer dans l'arborescence du système de fichiers

1. La commande `tree` permet d'afficher l'arborescence du système de fichiers à partir du répertoire passé en argument (le répertoire courant par défaut). Ci-dessous, une partie de l'affichage de l'arborescence d'un *système de fichiers* (répertoires seulement) depuis le *répertoire racine* `/`. Le *chemin absolu* du répertoire `grub` est `/boot/grub`.

```
junier@fredportable:/$ tree -L 2 -d
.
├── bin -> usr/bin
├── boot
│   ├── efi
│   └── grub
├── cdrom
├── dev
│   ├── block
│   ├── bsg
│   └── bus
```

Figure 1: arborescence racine

2. La commande `pwd` pour *print work directory* permet d'afficher le *répertoire courant dit de travail*. Le symbole tilde `~` est un raccourci pour désigner le répertoire personnel de l'utilisateur, en général `/home/utilisateur`.

```
junier@fredportable:~/sandbox$ pwd
/home/junier/sandbox
```

3. La commande `tree` sans argument permet alors d'afficher toute l'arborescence depuis le répertoire courant qui est représenté par un `..`. Le *chemin relatif* du fichier `photo1.jpg` par rapport au répertoire courant `sandbox` est `./rep1/photo1.jpg` ou `rep1/photo1.jpg` et son chemin absolu par rapport au répertoire racine s'obtient en le faisant précéder par le chemin absolu de `sandbox`, c'est donc `/home/junier/sandbox/rep1/photo1.jpg`.

```
junier@fredportable:~/sandbox$ tree
.
├── fichier1
├── fichier2
├── fichier3
├── fichier4
├── rep1
│   ├── photo1.jpg
│   └── photo2.jpg
├── rep2
│   ├── son1.jpg
│   └── son2.jpg
```

Figure 2: arborescence relative

4. La commande `cd` pour *change directory* permet de changer de répertoire courant.
 - Sans argument ou avec `cd ~` elle ramène l'utilisateur dans son répertoire personnel `/home/utilisateur`. `cd -` ramène dans le répertoire précédent

```
junier@fredportable:~/sandbox$ cd
junier@fredportable:~$ pwd
/home/junier
junier@fredportable:~$ cd -
/home/junier/sandbox
```

- `cd ..` permet de remonter dans le répertoire parent :

```
junier@fredportable:~/sandbox$ pwd
/home/junier/sandbox
junier@fredportable:~/sandbox$ cd ..
junier@fredportable:~$ pwd
/home/junier
```

- On peut fournir à `cd` un chemin absolu ou relatif mais il faut que le chemin soit uniquement constitué de répertoires !

```
junier@fredportable:~/sandbox$ cd /home/junier/sandbox/rep1
junier@fredportable:~/sandbox/rep1$ cd ..
junier@fredportable:~/sandbox$ cd rep1
junier@fredportable:~/sandbox/rep1$ cd photo1.jpg
bash: cd: photo1.jpg: N'est pas un dossier
```

3 Copier, supprimer, déplacer un fichier

1. La commande `mv` pour move sert à déplacer ou renommer des fichiers ou des répertoires. Elle prend deux arguments *source* et *cible* : si la *cible* est un répertoire, alors la *cible* est copiée dedans sinon elle est renommée.

```
junier@fredportable:~/sandbox$ ls
fichier1 fichier2 fichier3 fichier4 rep1 rep2 rep3
junier@fredportable:~/sandbox$ mv fichier1 fichier1-copie
junier@fredportable:~/sandbox$ ls
fichier1-copie fichier2 fichier3 fichier4 rep1 rep2 rep3
junier@fredportable:~/sandbox$ ls rep1
photo1.jpg photo2.jpg
junier@fredportable:~/sandbox$ mv fichier1-copie rep1
junier@fredportable:~/sandbox$ ls rep1
fichier1-copie photo1.jpg photo2.jpg
junier@fredportable:~/sandbox$ mv rep1 rep2
junier@fredportable:~/sandbox$ ls rep2
rep1 son1.jpg son2.jpg
```

2. La commande `cp` permet de copier des fichiers. Elle s'utilise comme `mv`, sauf que le fichier *source* n'est pas supprimé. Par défaut `cp` ne copie que des fichiers, pour copier un répertoire et son contenu, il faut lui passer l'option `-R` pour recursive.

```
junier@fredportable:~/sandbox$ ls
fichier2 fichier3 fichier4 rep2 rep3
junier@fredportable:~/sandbox$ ls rep2
```

```

rep1 son1.jpg son2.jpg
junier@fredportable:~/sandbox$ cp fichier2 rep2
junier@fredportable:~/sandbox$ ls rep2
fichier2 rep1 son1.jpg son2.jpg
junier@fredportable:~/sandbox$ cp rep2 rep3
cp: -r non spécifié ; omission du répertoire 'rep2'
junier@fredportable:~/sandbox$ cp -R rep2 rep3
junier@fredportable:~/sandbox$ ls rep3
rep2

```

3. La commande `rm` permet de supprimer les fichiers qu'on lui passe en argument. Pour supprimer un répertoire et son contenu, il faut lui passer l'option `-R` comme pour `cp`. *Attention, `rm` ne déplace pas les fichiers vers une corbeille, ils sont supprimés définitivement !*

```

junier@fredportable:~/sandbox$ ls
fichier2 fichier3 fichier4 rep2 rep3
junier@fredportable:~/sandbox$ rm fichier2
junier@fredportable:~/sandbox$ ls
fichier3 fichier4 rep2 rep3
junier@fredportable:~/sandbox$ rm rep3
rm: impossible de supprimer 'rep3': est un dossier
junier@fredportable:~/sandbox$ rm -r rep3
junier@fredportable:~/sandbox$ ls
fichier2 fichier3 fichier4 rep2

```

4 Expansion des noms de fichiers et *globbing*

On peut agir en masse sur des fichiers grâce aux mécanismes d'expansion de la ligne de commandes : certains caractères spéciaux indiquent au *shell* qu'il peut les remplacer par des ensembles de caractères. On peut ainsi décrire des motifs (ou *pattern*) pour décrire des ensembles de noms de fichiers.

Pour les exemples, on considère un répertoire `rep3` qui contient plusieurs fichiers :

```

junier@fredportable:~/sandbox/rep3$ ls
image1.jpeg image1.jpg image1.png image2.jpeg image2.jpg image2.png image3.jpeg image3

```

- Le caractère spécial `*` représente n'importe quelle suite de caractères. Par exemple pour lister les fichiers dont le nom de base se termine par `1` et l'extension par `g` on peut écrire :

```

junier@fredportable:~/sandbox/rep3$ ls *1.*g
image1.jpeg image1.jpg image1.png

```

- Le caractère spécial `?` représente un caractère unique quelconque. Par exemple pour lister les fichiers dont le nom de base se termine par `1` et l'extension comporte trois caractères et se termine par `g` on peut écrire :

```

junier@fredportable:~/sandbox/rep3$ ls *1.??g
image1.jpg image1.png

```

- Dans un nom de fichier existant `{a..z}` représente un caractère entre `a` et `z`. Par exemple pour lister les fichiers `image1.png`, `image2.png` et `image3.png` on peut écrire :

```
junier@fredportable:~/sandbox/rep3$ ls image{1..3}.png
image1.png image2.png image3.png
```

5 Gestion des droits sur les fichiers

Considérons le contenu du répertoire ~/sandbox affiché de façon détaillée avec la commande `ls -l` :

```
junier@fredportable:~/sandbox$ ls -l
total 8
-rwxrw-r-- 1 junier junier  0 août  16 21:43 fichier3
-rw-rw-r-- 1 junier junier  0 août  16 21:43 fichier4
drwxrwxr-x 3 junier junier 4096 août  16 23:29 rep2
drwxrwxr-x 2 junier junier 4096 août  16 23:33 rep3
```

1. Les 10 premiers caractères d'une ligne représentent les droits sur le fichier (ou le répertoire) :
 - Pour `fichier3` on a `-rw-rw-r--` :
 - le premier caractère `-` indique qu'il s'agit d'un fichier
 - le premier bloc de trois caractères `rw` représente les droits pour le *propriétaire* (*u*) du fichier : *lecture* (*r*), *écriture* (*w*) et *exécution* (*x*).
 - le second bloc de trois caractères `rw-` représente les droits pour le *groupe* (*g*) du fichier : *lecture* (*r*), *écriture* (*w*) et un tiret `-` qui marque l'absence de droit d'exécution
 - le dernier bloc de trois caractères `rw-` représente les droits pour les *autres* (*o*) utilisateurs du fichier : ce sont les mêmes que pour le *groupe*.
 - Pour `rep2` on a `drwxrwxr-x` :
 - le premier caractère `d` indique qu'il s'agit d'un répertoire
 - les trois blocs de trois caractères suivants énumèrent les droits en *lecture* (*r*), *écriture* (*w*), *exécution* (*x*) des trois types d'utilisateurs du répertoire : *propriétaire*, *groupe* et *autres*.
2. Le propriétaire d'un fichier ou le superutilisateur `root` peut changer les droits d'un fichier ou d'un répertoire avec la commande `chmod` dont la syntaxe est :

```
chmod [-R] [ugoa][+==][rwx] fichier
```

- Les options entre crochets désignent :
 - `u` : le propriétaire
 - `g` : le groupe
 - `o` : les autres utilisateurs
 - `a` : tous les utilisateurs
 - `+` : ajouter le(s) droit(s)
 - `-` : enlever le(s) droit(s)
 - `=` : positionner le(s) droit(s)
 - `r` : droit de lecture
 - `w` : droit d'écriture
 - `x` : droit d'exécution
 - `-R` : récursivement (nécessaire pour agir sur un répertoire)
- Quelques exemples :
 - Fixer les droits à `-x` (écriture seule) pour tous les utilisateurs sur `fichier3` :

```
junier@fredportable:~/sandbox$ chmod a=x fichier3
```

- Donner le droit d'écriture aux autres utilisateurs sur fichier4:
junier@fredportable:~/sandbox\$ chmod o+w fichier4
- Enlever le droit de lecture à tous les utilisateurs sur fichier4 :
junier@fredportable:~/sandbox\$ chmod ugo-r fichier4
- Enlever le droit de lecture au groupe sur le répertoire rep2 :
junier@fredportable:~/sandbox\$ chmod -R g-w rep2
- Affichage des droits après les modifications précédentes :
junier@fredportable:~/sandbox\$ ls -l
total 8
---x--x--x 1 junier junier 0 août 16 21:43 fichier3
--w--w--w- 1 junier junier 0 août 16 21:43 fichier4
drwxrwxr-x 3 junier junier 4096 août 16 23:29 rep2
drwxrwxr-x 2 junier junier 4096 août 16 23:33 rep3

3. Le superutilisateur root peut modifier le propriétaire d'un fichier avec la commande `chown`. Il faut passer l'option `-R` pour modifier le propriétaire d'un répertoire.

```
junier@fredportable:~/sandbox$ sudo chown root fichier3
[sudo] Mot de passe de junier :
junier@fredportable:~/sandbox$ ls -l fichier3
---x--x--x 1 root  junier  0 août  16 21:43 fichier3
```

4. Le superutilisateur root peut modifier le groupe d'un fichier avec la commande `chgrp`. Il faut passer l'option `-R` pour modifier le groupe d'un répertoire.

```
junier@fredportable:~/sandbox$ sudo chgrp root fichier3
junier@fredportable:~/sandbox$ ls -l fichier3
---x--x--x 1 root  root    0 août  16 21:43 fichier3
```