Commandes utiles du terminal
Aide-mémoire

MP2I Lycée Pierre de Fermat

Vocabulaire

- **Répertoire courant** : Dossier où le terminal se trouve actuellement. Les commandes lancées prendront effet à partir de ce dossier.
- **Chemin** : Suite de noms de dossiers, de la forme `dossierA/dossierB/dossierC/...` pouvant éventuellement finir par un nom de fichier (auquel cas c'est un chemin **vers** le fichier).
- **Chemin absolu/relatif** : Un chemin commençant par `/` démarre à partir de la racine de l'ordinateur, on dit qu'il est absolu. Sinon, le chemin part du répertoire courant et on dit qu'il est relatif.
- **Home** : Le répertoire par défaut, ou "home", est le répertoire où se situe le terminal à son lancement. Le symbole est un raccourci pour le home dans un terminal.

Préliminaire : message d'aide d'une commande

Cette fiche regroupe quelques commandes pratiques à connaître dans un terminal. Chaque commande est présentée de manière non-exhaustive, en en donnant l'utilisation de base et quelques arguments optionnels pratiques. La plupart des commandes ont un mode "aide" qui vous affiche un message expliquant toutes les manières de les utiliser. Ce mode s'active généralement avec l'option `--help` ou bien `-h`. Par exemple, la commande `mkdir` sert à créer des dossiers, et si l'on tape `mkdir --help` dans le terminal, on obtient le message d'aide suivant :

```
Usage: mkdir [OPTION]... DIRECTORY...
Create the DIRECTORY(ies), if they do not already exist.

Mandatory arguments to long options are mandatory for short options too.
  -m, --mode=MODE   set file mode (as in chmod), not a=rwx - umask
  -p, --parents     no error if existing, make parent directories as needed
  -v, --verbose     print a message for each created directory
  -Z                   set SELinux security context of each created directory
                         to the default type
      --context[=CTX]  like -Z, or if CTX is specified then set the SELinux
                         or SMACK security context to CTX
      --help     display this help and exit
      --version  output version information and exit
```

On voit alors que l'option `--parents`, aussi utilisable avec `-p`, permet de créer les dossiers intermédiaires s'ils n'existent pas. Donc, la commande :

```
mkdir bla/truc --parents
```

crée le dossier `bla` s'il n'existe pas déjà, puis le sous-dossier `bla/truc` s'il n'existe pas. La commande `mkdir bla/truc` seule n'aurait pas fonctionné !

Dans un terminal, la commande `man mkdir` affiche une version plus détaillée du message d'aide.

Se repérer dans l'arborescence

pwd La commande `pwd` (comme **P**rint **W**orking **D**irectory) affiche le chemin absolu du répertoire courant.

ls La commande `ls` affiche le contenu du répertoire courant.
Options utiles :

- `--all` / `-a` : Affiche les fichiers et dossiers cachés, c'est à dire dont le nom commence par un point.
- `--recursive` / `-R` : Affiche récursivement tous les sous-dossiers et leur contenu.
- `--size` / `-s` : affiche la taille de chaque fichier.

cd La commande `cd` (comme **C**hange **D**irectory) permet changer le répertoire courant. On peut préciser un répertoire ou même un chemin entier, et utiliser `..` pour signifier "en arrière". Par exemple, la commande

```
cd ../../bla/truc
```

revient de deux étages en arrière dans l'arborescence, puis descend vers le dossier `bla/`, puis vers le dossier `truc/`.

Taper `cd` sans argument change le répertoire courant en le répertoire par défaut, le "home".

Création et suppressions de fichiers

touch La commande `touch` permet de créer de nouveaux **fichiers** :

```
touch fichier1 fichier2 ...
```

On peut spécifier un chemin complet vers le fichier à créer, mais ce chemin doit être valide : la commande touch ne créera pas les dossiers s'ils n'existent pas.

mkdir La commande `mkdir` permet de créer de nouveaux **dossiers** :

```
mkdir dossier1 dossier2 ...
```

Comme pour `touch`, on peut spécifier des chemins complets, mais ils doivent être valides.
Options utiles :

- `--recursive` ou `-R` : crée les dossiers intermédiaires s'ils n'existent pas.

rm La commande **rm** (comme **R**e**M**ove) sert à supprimer des fichiers :

```
rm fichier1 fichier2 ...
```

Options utiles :

- `-r` / `--recursive` supprime récursivement le contenu des sous-dossiers
- `-v` / `--verbose` affiche les fichiers supprimés par la commande

Par exemple, la commande `rm --recursive --verbose bla`, que l'on peut abréger en `rm -rv bla`, supprime le répertoire `bla` et tous les dossiers et fichiers qu'il contient, tout en affichant chaque fichier et dossier supprimé.

cp La commande **cp** (comme **C**o**P**y) permet de copier un fichier. On spécifie en premier le fichier à copier, puis le nom de la copie à créer :

```
cp fichier_source fichier_a_creer
```

Options utiles :

- `-r` / `--recursive` permet de copier un dossier entier, et pas juste un fichier.

Contenu des fichiers

cat La commande `cat` affiche le contenu de fichiers dans le terminal :

```
cat fichier1 fichier2 ...
```

head / tail La commande `head` permet d'afficher les 10 premiers lignes de fichiers :

```
head fichier1 fichier2 ...
```

Options utiles :

- `-n` : permet de spécifier un nombre de lignes à afficher. La commande `head -n 1 bla.c truc.c` afficherait donc la première ligne des deux fichiers C.

La commande `tail` marche exactement pareil, mais affiche les **dernières** lignes.

Wildcard

Lorsqu'une commande requiert comme argument un ou plusieurs noms de fichiers, on peut utiliser des **motifs** qui permettent de sélectionner plusieurs fichiers d'un coup. Dans un motif, le symbole `*`, appelé un **joker** ou **wildcard**, peut être remplacé par n'importe quoi. Par exemple, le motif `"*.c"` signifie "n'importe quoi qui finit par `.c`". Donc, la commande

```
head -n 1 *.c
```

affiche la première ligne de **tous** les fichiers C du répertoire actuel. De même, la commande

```
rm photos/201*/*.jpg
```

supprimerait toutes les photos prises entre 2010 et 2019.

Recherche

find La commande `find` permet d'afficher certains fichiers de votre ordinateur. Par défaut, sans arguments, elle affiche l'intégralité des fichiers et dossiers du répertoire courant, récursivement (elle descend donc dans l'intégralité de l'arborescence). L'option `-name` permet de spécifier un motif à chercher. Ainsi, la commande

```
find -name "*.c"
```

affichera tous les fichiers dont le nom finit par `.c`. De la même manière, la commande

```
find -name "*ga*.c"
```

affichera tous les fichiers C dont le nom contient "ga".

Archivage

zip La commande `zip` permet de créer une archive contenant plusieurs fichiers. On spécifie d'abord le nom de l'archive à créer, puis les noms des fichiers à mettre dedans :

```
zip mon_archive.zip fichier1 fichier2...
```

Options utiles :

- `-r` : permet de spécifier aussi des dossiers, qui seront archivés récursivement.

Par exemple, si le répertoire courant contient :

```
TP1/
TP2/
TP2/exo1/code.c
...
TP2/exo15/code.c
TP3/
...
```

Pour archiver le contenu du dossier TP2, on pourrait spécifier à la main chaque fichier :

```
zip TP2.zip TP2/exo1/code.c ... TP2/exo15/code.c
```

Mais on peut simplement taper `zip -r TP2.zip TP2` et directement archiver tout le contenu du dossier `TP2`.

unzip La commande `unzip` permet de décompresser une archive, c'est à dire d'extraire les fichiers et dossiers qu'elle contient. La commande met le contenu de l'archive dans le répertoire courant.

```
unzip mon_archive.zip
```

Options utiles :

- `-d` : permet de spécifier un chemin où mettre le contenu de l'archive, plutôt que de le mettre dans le répertoire courant :
  `unzip mon_archive.zip -d autre_dossier`
- `-v` : affiche des informations sur les fichiers et dossiers extraits (taille, taux de compression, etc...)

Mode verbeux

La commande précédente a un mode permettant d'afficher plus d'informations sur son exécution. De nombreuses commandes ont cette possibilité, que l'on appelle un mode **verbose**, elle s'active généralement avec l'option `-v` ou `--verbose`.

Commandes moins utiles

cowsay La commande `cowsay` affiche un dessin ASCII d'une vache répétant un message.

```
cowsay coucou
```

Options "utiles" :

- `-f` permet de changer la forme de la vache en précisant un autre animal : `elephant`, `dragon`, ...
- `-l` affiche la liste des formes existantes pouvant être utilisées avec `-f`