Antisèche de commandes Shell

Afficher un message sur la sortie standard : echo. Exemple :

```
nlouvet@rrunch:~/tctest$ echo "Toto fait du vélo !"
Toto fait du vélo !
```

Afficher le chemin absolu du répertoire courant : pwd (pour *print working directory*). Exemple :

```
nlouvet@rrunch:~/tctest$ pwd
/home/nlouvet/tctest
```

Changer de répertoire courant : cd (pour *change directory*)
`cd destination`
↪ le répertoire courant devient le répertoire `destination` (pour peu que le chemin `destination` existe). Exemples :

- `cd cours/`
  ↪ On se déplace vers le sous-répertoire `cd cours/` du répertoire courant. Si par exemple le répertoire courant est `/home/nlouvet/`, le répertoire courant devient `/home/nlouvet/cours/`.
- `cd /etc`
  ↪ Le répertoire courant devient le répertoire `/etc/`.
- `cd ~` ou simplement `cd`
  ↪ Le répertoire courant devient le répertoire personnel de l'utilisateur qui entre la commande.

Lister le contenu d'un répertoire : ls (pour *list*)
`ls directory`
↪ liste sur la sortie standard le contenu (fichiers, sous-répertoires) du répertoire dont le chemin est `directory`. L'option **-l** (pour pour *long*) permet d'afficher des informations longues (détaillées) sur le contenu du répertoire. Exemples :

- `ls /etc/`
  ↪ liste le contenu du répertoire `/cours/`.
- `ls`
  ↪ liste le contenu du répertoire courant.

Copie de fichiers : cp (pour *copy*)
`cp source destination`
↪ copie le fichier `source` vers `destination` (`source` et `destination` sont des chemins). Exemples :

- `cp brouillon.txt notes.txt`
  ↪ Copie le fichier `brouillon.txt` vers le fichier `notes.txt` ; si `notes.txt` existait déjà, alors son contenu est écrasé, sinon il est créé.
- `cp /var/log/kern.log ./`
  ↪ Copie le fichier `kern.log`, situé dans le répertoire `/var/log/`, vers le répertoire courant (chemin `./`).
- Copie d'un répertoire avec l'option **-r** (pour *recursive*) :
  `cp -r /home/nlouvet/cours /tmp/`
  ↪ Copie le répertoire `/home/nlouvet/cours` (tous les fichiers et les sous-répertoires) vers `/tmp/`. On se retrouve avec un répertoire `/tmp/cours/`.

Déplacement ou renommage de fichiers : mv (pour *move*)
`mv source destination`
↪ déplace le fichier `source` de façon à ce qu'il deviennne le fichier destination (`source` et `destination` sont des chemins). Exemples :

- `mv brouillon.txt notes.txt`
  ↪ Renomme le fichier `brouillon.txt` en `notes.txt`.
- `mv notes.txt /tmp/`
  ↪ Déplace le fichier `notes.txt` vers le répertoire `/tmp/`.
- `mv notes.txt /tmp/brouillon.txt`
  ↪ Déplace le fichier `notes.txt` vers le répertoire `/tmp/`, en le renommant `brouillon.txt` ; de façon équivalente, renomme `notes.txt` en `/tmp/brouillon.txt`.

Envoyer un fichier vers la sortie standard : cat (pour *concatenate*, car on peut concaténer des fichiers avec)
`cat fichier`
↪ Envoie le contenu de `fichier` (il s'agit d'un chemin) vers la sortie standard. Exemple :

```
nlouvet@rrunch:~$ cat notes.txt
toto junior  4/20
toto senior  5/20
titi        15/20
tutu        16/20
```

Filtrer les lignes d'un fichier texte : grep (pour *global regular expression print*)
`grep motif fichier`
↪ Envoie toutes les lignes contenant le mot `motif`, parmi les lignes de `fichier`, sur la sortie standard. En réalité, `motif` peut être une expression régulière, pour permettre de filtrer finement les lignes que l'on souhaite obtenir. Exemple, avec le fichier `notes.txt` dont on a affiché le contenu ci-dessus :

```
nlouvet@rrunch:~$ grep "toto" notes.txt
toto junior  4/20
toto senior  5/20
```

Compter le nombre de mots, de lignes ou de caractères d'un fichier : wc (pour *word count*)
`wc [-w][-l][-c] fichier`
↪ L'option `-w` permet de compter le nombre de mots, `-l` le nombre de lignes, et `-c` le nombre de caractères. Ces options peuvent être combinées. Exemples, encore avec le même fichier :

```
nlouvet@nlbook:~$ wc note.txt
 5 10 73 note.txt
nlouvet@nlbook:~$ wc -l note.txt
5 note.txt
nlouvet@nlbook:~$ wc -c note.txt
73 note.txt
nlouvet@nlbook:~$ wc -lc note.txt
 5 73 note.txt
```

Application de filtres à l'aide du pipe |
Le pipe permet d'enchaîner plusieurs commandes, de la façon suivante :

$$\texttt{commande}_1 \,|\, \texttt{commande}_2 \,|\, \cdots \,|\, \texttt{commande}_n$$

Cela signifie que, pour tout i de 1 à $n-1$, la sortie standard de $\texttt{commande}_i$ est redirigée vers l'entrée standard de $\texttt{commande}_{i+1}$; la sortie standard de $\texttt{commande}_n$ n'est pas modifiée, donc cette commande produit son résultat sur sa sortie standard, c'est-à-dire dans le terminal. Exemples, toujours avec le même fichier :

- `cat notes.txt | grep "tutu"`
  ↪ Redirige le contenu du fichier `notes.txt` vers un filtre `grep` ; on obtient ici « `tutu  16/20` ».
- `cat notes.txt | grep "toto" | wc -l`
  ↪ Compte le nombre de lignes du fichier `notes.txt` qui comportent le mot `toto`. Cela doit donner ici 2.

Retrouver des fichiers dans une arborescence : find
`find racine -name motif`
↪ cherche, dans l'arborescence de répertoires enracinée au niveau du chemin `racine`, tous les fichiers ou répertoires dont le nom correspond au `motif`. À la place de l'option `-name`, on peut aussi utiliser `-iname` pour que la recherche ne soit pas sensible à la casse (majuscule / minuscule). Des exemples :

- `find ./ -name "notes.txt"`
  ↪ Recherche, à partir du répertoire courant, tous les fichiers nommés `notes.txt`.
- `find ~/Documents/ -iname "*.jpg"`
  ↪ Recherche, à partir du sous-répertoire `~/Documents/` (un sous-répertoire de votre répertoire personnel), tous les fichiers se terminant par `.jpg` (ou `.JPG`, ou `.Jpg`, ... car la recherche est ici insensible à la casse).

Edition de la sortie standard : sed

`sed -E 's/motif/remplacement/'`
↪ `sed` est une commande d'édition assez évoluée, dont on ne va illustrer qu'une seule fonctionnalité : la substitution (d'où le `s`). La commande indiquée remplace, sur chaque ligne reçue sur son entrée standard, la première occurrence de `motif` par `remplacement`. Des exemples :

- `echo "Tutu fait du vélo !" | sed -E 's/vélo/pédalo/'` affiche `Tutu fait du pédalo !`
- `echo "Tutu fait du vélo !" | sed -E 's/f.*t/a fait/'` affiche `Tutu a fait du vélo !`
  ↪ Le motif `f.*t` correspond à n'importe quel mot commençant par `f` et finissant par `t`.
- `echo "Tutu fait du vélo !" | sed -E 's/v.*/???/'` affiche `Tutu fait du ???`
  ↪ Le motif `v.*` correspond à `v` suivi de n'importe quel mot, et correspond donc à toute la fin de la ligne.