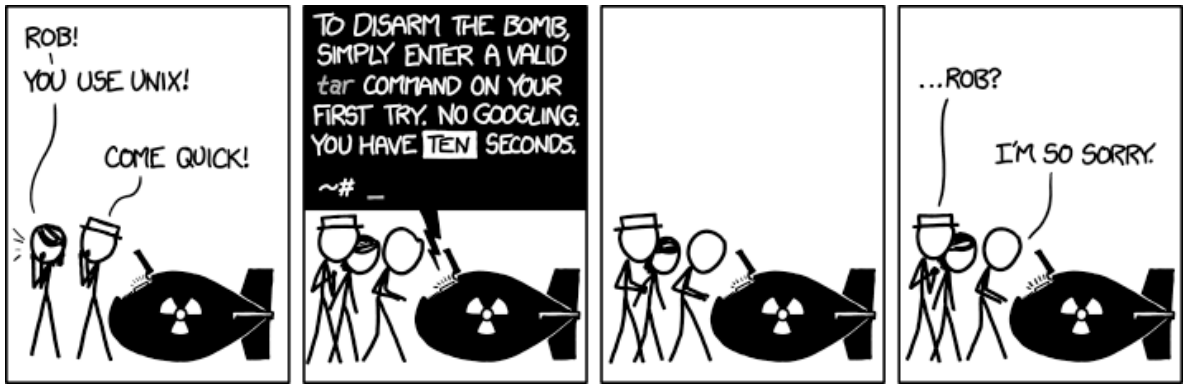
TD nº 1 - Commandes UNIX



Exercice 1. *Navigation*

Dans cet exercice nous allons nous intéresser aux commandes permettant de manipuler l'arborescence de fichiers du système.

`ls` (*list*) permet de lister le contenu d'un répertoire. Par défaut elle liste le contenu du répertoire courant, mais on peut lui passer en argument le chemin du répertoire à observer (*e.g.* `ls /usr/bin`). Il existe de nombreuses options pour modifier la présentation du résultat, parmi lesquelles `-a` qui affiche tous les fichiers y compris les fichiers « cachés » (ceux dont le nom commence par « . »), `-l` qui affiche des informations plus détaillées (taille des fichier, propriétaire, droits d'accès,...), `-R` qui parcourt les répertoires récursivement.

`cd` (*change directory*) change le répertoire courant. Si la commande est lancée sans argument, c'est le répertoire personnel de l'utilisateur (*home*) qui devient le répertoire courant. L'argument est le nouveau répertoire dans lequel il faut se placer, qui peut être donné de manière absolue ou relative. La notation `..` désigne le répertoire parent d'un répertoire (on peut utiliser cette notation de manière répétée, ainsi `../..` désigne le parent du parent).

`pwd` (*print working directory*) affiche le répertoire courant.

`mkdir` (*make directory*) sert à créer un répertoire dont le nom est donné en argument.

`touch` permet de créer un fichier vide (le nom du fichier à créer est passé en argument).

`mv` (*move*) permet de déplacer un fichier ou répertoire. Cette commande prend deux arguments : le fichier source et sa destination. C'est également cette commande qui permet de renommer un fichier en le « déplaçant » vers un autre nom (*e.g.* `mv titi.txt toto.txt`).

`cp` (*copy*) copie un fichier ou un répertoire d'un emplacement vers un autre. La syntaxe est similaire à celle de la commande `mv` (source puis cible). L'option `-R` permet d'effectuer une copie *récursive* dans le cas où la source est un répertoire.

`rm` (*remove*) supprime le ou les fichiers ou répertoires passés en argument. Pour supprimer un répertoire il faut ajouter l'option `-r` qui indique que la suppression doit-être effectuée récursivement.

`file` donne des informations sur le contenu d'un fichier (le format des données, l'encodage, etc.).

1. Placez-vous dans votre répertoire personnel, créez un répertoire nommé `archiDU` et déplacez-vous dans ce répertoire.

R.

```
$ cd
$ mkdir archiDU
$ cd archiDU
```

2. Dans le répertoire `archiDU`, créez un fichier `vid.txt`. Renommez ensuite ce fichier en `vide.txt`.

R.

```
$ touch vid.txt
$ mv vid.txt vide.txt
```

3. Copiez le fichier `vide.txt` deux fois en donnant le nom que vous voulez aux copies.

R.

```
$ cp vide.txt copie1.txt
$ cp vide.txt copie2.txt
```

4. Tout en restant dans le répertoire `archiDU`, créez un répertoire `test` dans votre dossier personnel. Déplacez ensuite tous les fichiers de `archiDU` dans le nouveau répertoire `test`.

R.

```
$ mkdir ../test
$ mv * ../test
```

5. Déplacez le répertoire `test` (et ce qu'il contient) dans `archiDU`. Affichez le contenu de `archiDU` d'abord de manière simple puis de manière à voir tous les fichiers contenus dans les sous-répertoires.

R.

```
$ mv ../test .
$ ls archiDU
$ ls -R
```

6. Supprimez le répertoire `test` et tout ce qu'il contient.

R.

```
$ rm -r ./test
```

Exercice 2. *Quelques autres outils*

Nous allons maintenant effectuer quelques actions un peu plus complexes en utilisant des fonctions *Unix* un peu plus élaborées.

`cat` (*catenate*) affiche le contenu de son entrée dans sa sortie (!). L'utilisation usuelle est de donner en arguments une liste de fichiers, la commande affiche alors à la suite les contenus de ces fichiers dans la sortie standard. Cependant, si la commande est lancée sans paramètres, c'est l'entrée standard (entrée au clavier) qui est affichée sur la sortie standard (affichage à l'écran).

`less` permet de visualiser le contenu d'un fichier dans le terminal. Il est possible de parcourir le fichier avec les flèches *haut* et *bas* du clavier, de rechercher un mot en appuyant sur `/` (puis `n` pour le résultat suivant et `N` pour le précédent). Appuyez sur `Q` pour quitter.

`head` et `tail` permettent d'afficher les n premières ou dernières lignes respectivement d'un fichier. Le nombre de lignes est passé en option (*e.g.* `head -10 toto.txt` affiche les 10 premières lignes du fichier `toto.txt`).

`grep` parcourt des lignes de texte et n'affiche que les listes contenant un motif particulier. Le motif est donné en premier argument. Le second argument peut être un fichier (dans ce cas le programme cherche le motif dans les lignes du fichier). S'il n'y a pas de second argument, c'est sur l'entrée standard que le motif est cherché.

`sort` trie les lignes d'un fichier dans l'ordre croissant (par défaut).

`wc` (*word count*) compte le nombre de lignes, de mots et de caractères dans le fichier passé en argument (ou l'entrée standard si aucun argument n'est donné).

Il est possible de combiner les actions de plusieurs fonctions, en redirigeant la sortie d'une commande vers l'entrée de la suivante. On utilise alors le symbole `|` entre les deux commandes. Il est également possible de rediriger le résultat d'une commande vers le contenu d'un fichier avec > ou >> (le premier écrase le contenu existant, tandis que le second ajoute le résultat à la suite du fichier) ou à l'inverse d'utiliser le contenu d'un fichier comme entrée avec <.

1. Affichez la liste des fichiers du répertoire `/usr/bin` dont le nom commence par un `t`.

R.

```
$ ls /usr/bin/t*
```

2. En utilisant la fonction `cat`, créez un fichier `poeme.txt` contenant quelques lignes de texte.

R.

```
$ cat > poeme.txt
```

Puis taper quelques lignes de texte. Quitter avec `^D`.

3. Copiez le fichier `poeme.txt` en `poesie.txt` sans utiliser la commande `cp`.

R.

```
$ cat poeme.txt > poesie.txt
```

4. En lisant le manuel de la commande `cat` (`man cat`), trouvez comment transformer le fichier `poesie.txt` pour que les lignes soient numérotées.

R.

```
$ cat -n poeme.txt > poesie.txt
```

L'option `-n` permet de numéroter les lignes affichées par `cat` (l'option `-b` numérote uniquement les lignes non vides). Attention, on ne peut pas lire et écrire dans le même fichier à la fois.

Récupérez le dictionnaire de la langue française qui se trouve sur la page du cours (`http://www.lirmm.fr/~poupet/enseignement/archidu12.php`) et sauvegardez-le dans `~/archiDU`.

5. Quels sont les deux seuls mots de la langue française qui se terminent par *mome* ?

R.

```
$ grep "mome$" td01-dict.txt
cardamome
cinnamome
```

La commande `grep` prend en argument une *expression régulière*. Le symbole `$` indique une fin de ligne (le symbole `^` sert à indiquer le début de la ligne).

6. Créez un fichier contenant tous les mots de la langue française qui se terminent par *als* suivis de tous ceux qui se terminent par *ails*. Combien y a-t-il de tels mots ?

R.

```
$ grep "als$" td01-dict.txt > mots.txt
$ grep "ails$" td01-dict.txt >> mots.txt
$ wc mots.txt
      98      98     897 mots.txt
```

La redirection avec `>` créé un nouveau fichier, la redirection avec `>>` ajoute à la fin d'un fichier existant (ou dans un nouveau fichier s'il n'existe pas). La commande `wc` indique qu'il y a 98 mots dans le fichier `mots.txt`.

7. Créez un fichier contenant tous les mots de plus de 20 caractères.

R.

```
$ grep "....................." td01-dict.txt > mots.txt
$ grep ".\{20\}" td01-dict.txt > mots.txt
```

Les deux commandes indiquées ci-dessus sont équivalentes. Dans la première, il y a 20 points. La seconde utilise une notation qui signifie ici 20 fois un caractère quelconque.

8. Affichez le contenu du dictionnaire entre les lignes 300 et 320.

R. On utilise les commandes `head` et `tail` de manière astucieuse : on demande d'abord les 320 premières lignes du fichier, puis on ne garde que les 20 dernières de ces 320 premières :

```
$ head -320 td01-dict.txt | tail -20
```

9. Quels sont les mots de 7 lettres de la forme `.om..ne` (pratique pour les mots croisés) ? Voyez-vous un moyen de trouver tous les mots contenant certaines lettres fixées (pratique pour le *scrabble*) ?

R.

```
$ grep "^.om..ne$" td01-dict.txt
```