TD : Programmation sous Linux — semaine 1

9 février 2018

Objectif(s)

- ✯ S'approprier l'environnement UNIX/Linux en utilisant diverses commandes consoles.
- ✯ Apprendre à écrire un script bash.

Exercice 1 – Préliminaires

Tout au long de ce TD nous allons utilisez de nombreuses commandes UNIX. Faites vous un "carnet de l'ingénieure" dans lequel vous listerez ces commandes afin de ne pas les oublier et documenterez toutes difficultés rencontrées.

1. Ouvrez un terminal à l'aide du raccourci `CTRL+ALT+T` ou via le menu graphique.
2. La console permet de naviger dans les répertoires et fichiers de votre système. Nous allons effectuer, en console, les étapes suivantes :
   — Créer un répertoire `ISUP` avec `mkdir ISUP`
   — Rentrer dans ce répertoire avec `cd ISUP`
   — Créer un autre répertoire `informatique` et puis `td1` à l'intérieur de ce dernier.
   — Retourner dans le répertoire parent avec `cd ..` et lancer la commande `ls` ; quelle est sa fonction ?
   — Taper maintenant `cd ~`. Où êtes-vous ? Que signifie le `~` ?
   — Retourner dans `td1`.
3. Vous êtes normalement dans le répertoire `~/ISUP/informatique/td1`, vérifiez que c'est bien le cas en tapant `pwd`. Taper la commande `gedit` qui lancera l'éditeur de texte Gedit. Créer et sauvegarder un fichier dans ce même répertoire comprenant plusieurs lignes avec, par exemple, la liste de vos musiques préférées.
4. Retourner dans la console. Vous remarquez que, si vous n'avez fermé Gedit, vous ne pouvez plus utiliser la console. En effet, celle-ci se bloque jusqu'à ce que le programme lancé (ici Gedit) se termine. Fermer Gedit dans la console à l'aide de l'interruption `CTRL + C`. Maintenant, relancez le avec la commande `gedit &`, ce caractère `&` à la fin de toute commande permet de lancer un programme via la console et que celle-ci soit toujours utilisable (le programme est lancé en tâche de fond).
5. À partir de maintenant, toutes les commandes ne seront plus décrites en détail à chaque fois. Cependant, deux moyens pour vous aidez dans votre quête :
   — Adopter un esprit *shonen* et utiliser la commande `man` qui permet de se renseigner sur une autre commande. Par exemple si vous tapez `man cd` vous aurez des informations sur `cd`. Par contre, vous aurez vraiment *toutes* les informations (d'où l'esprit *shonen*).
   — Et sinon il vous reste le vénérable et honorable Google, qui avec une question bien posée (pas toujours gagné), peut vous aidez. Une technique est de chercher des exemples d'utilisations (taper "examples cd command linux"). Notamment, pour l'informatique, le site *stackoverflow.com*.
6. Maintenant, on va voir si vous maitrisez la recherche d'information. À l'aide la commande `wc` compter le nombre de mots de votre fichier. Ensuite, trouver les lignes qui contiennent la lettre "e" (commande `grep`). Allez, premier niveau terminé.

Exercice 2 – Le cycle de la vie : créer, lister, bouger et effacer

1. Familiarisez-vous avec les options des commandes `ls`, `cp`, `mv` et `rm`. Créez un sous-répertoire dans `td1` et quelques fichiers dans le répertoire. Déplacez-les dans le répertoire père (`td1`), puis les effacer tous.
2. Afin de ne pas supprimer un fichier par erreur, il est plus prudent d'utiliser les commandes `rm`, `mv` et `cp` avec l'option `-i` ("interactive"). Créer s'il n'existe pas déjà le fichier `.bashrc` et y définir des alias pour ces trois commandes incluant l'option -i (commande `alias`). Au passage, renseignez-vous sur le fichier `.bashrc`.
3. Saisir les commandes suivantes en alternance avec la commande `pwd` :
   1. `ls *`, `ls .`, `ls ..`, `ls ../..`, `ls ~`, `ls /usr`,
      `ls /usr/*`, `ls /u*r/*`, `ls /u*/*`, `ls /u*/l*/lib[b-e]*`.
   2. `cd`, `cd .`, `cd ..`, `cd ../..`, `cd /`, `cd ~` et enfin `cd ~login` (où *login* représente un nom d'utilisateur valide).
   3. Entre autres, que signifie l'étoile `*` ? Et le chemin `/` ?

Exercice 3 – Je m'en fichier pas mal

Souvenez-vous que :

— `cat fichier1.txt > fichiers2.txt` va copier la sortie de la commande `cat fichier1.txt` dans le fichier `fichier2.txt`.
— `ls ~ -al | grep .bashrc` liste les fichiers du *home* et transfère le résultat vers l'entrée standard de `grep`.
— Conclusion : Il y a un fichier du côté de la pointe de > et < mais il y a une commande à droite d'un `|` qui attend quelque chose sur son entrée standard (note : certaines commandes n'attendent rien sur l'entrée standard, comme `ls` par exemple).

1. (Commande `cut`) Soit le fichier *proverbe.txt* contenant le texte suivant :

   Rien ne sert de courir
   Il faut partir à point.
   La Fontaine.

   1. Créer le fichier *proverbe.txt*.
   2. Donner la commande pour n'afficher que les 5 premiers caractères de chaque ligne.
   3. Donner la commande pour n'afficher que les 2 premiers mots de chaque ligne.
   4. Quel sera le résultat de la commande : `cut -f 2-3 -d ' '` de *proverbe.txt* ?

2. (Commandes `tail` et `sort`) Soit le fichier *pantheon.csv* contenant le texte suivant :

```
Nom ; Prenom ; Naissance
Page ; Jimmy ; 1944
Van Halen ; Eddie ; 1955
Vaughan ; Stevie Ray ; 1954
Clapton ; Eric ; 1945
Lagrene ; Birelli ; 1966
1Beck ; Jeff ; 1944
McLaughlin ; John ; 1942
Vai ; Steve ; 1960
Hendrix ; Jimi ; 1942
```

   1. Donner la commande qui permet d'obtenir le fichier *liste.csv* qui est une copie de *pantheon.csv* sans la première ligne (utiliser < et >).

2. Donner la commande pour obtenir un affichage de la liste des guitaristes par noms croissants (voir commande `cat`, et utiliser `|`).
3. Donner la commande pour obtenir un affichage de la liste par dates décroissantes

3. Aller dans votre répertoire *home* et taper la commande `ls -l -F -a`.
   — Donnez une explication rapide de chaque colonne.
   — Que signifie le point devant un fichier ou dossier ?
4. Créez un répertoire avec, dedans, le fichier *fichier.txt*. Utiliser la commande `chmod` pour (et vérifier avec `ls` quels attributs changent sur le fichier) :
   1. Mettre aucun droit d'accès à tout le monde au fichier *fichier.txt* (vérifier si ça marche en essayant de le lire avec `cat`),
   2. y mettre les droits de lecture, d'écriture et d'exécution au seul créateur du fichier,
   3. y mettre les droits de lecture seul au groupe du créateur et aux autres,
   4. y ajouter le droit d'exécution aux autres,
   5. y retirer les droits d'écriture et d'exécution au créateur.
5. Créer un fichier *mots.txt* contenant les mots suivants, un seul par ligne :

   ```
   boot book booze machine boots bungie bark aardvark broken$tuff robots
   ```

   — Écrire la commande qui affiche les lignes contenant le motif *boo*.
   — Idem mais avec en plus les numéros de ligne.
   — Le nombre de lignes où *boo* apparait.
   — Le motif qui permet de reconnaitre soit *boot* soit *boots*.
   — La commande qui affiche les lignes qui ne contiennent pas *boo*, avec leur numéro.

Exercice 4 – La quête du script

Un aspect très intéressant de la console est qu'elle est contrôlée par un langage de programmation qui s'appelle le *bash*. Depuis le début vous écrivez des programmes en *bash* très simple. Ça peut être très utile pour automatiser des traitements.

1. On va créer un nouveau fichier dans le répertoire `td1` nommé *next.sh* qui va effectuer les actions suivantes :
   1. Aller dans le dossier parent ;
   2. Créer un dossier `td2` ;
   3. Aller dans ce dossier.

   Il suffit pour cela de séparer les différentes commandes à l'aide de l'opérateur de séquence `;`. Par exemple `mkdir Test; rm -r Test` crée et puis, successivement, supprime le répertoire *Test*. Aussi, on mettra en haut du script (le fichier finissant en `.sh`) la ligne suivante :

   ```
   #!/bin/sh
   ```

   qui permet de dire quelle variante du langage on utilise. La dernière étape est de rendre ce script exécutable et de le lancer via `.\next.sh`.
2. On voudrait améliorer ce script pour qu'il fasse les actions supplémentaires suivantes :
   — Crée automatiquement un dossier `td`N où N est le numéro du TD courant + 1.
   — Se duplique dans le nouveau dossier.
   À vous de trouvez les commandes nécessaires pour automatiser ce traitement.
3. Vous venez de rentrer de soirée, perdu votre téléphone dans le RER, et vous devez vous levez à 7h demain pour aller en examen. Le problème ? Pas de téléphone, pas de réveil. Mais grâce à vos nouvelles compétences en bash, vous trouvez la force de programmer un réveil en seulement deux lignes de code... Mais est-ce que cela marchera... vous faites quand même un test avant.
   — Créer un script qui lance une musique dans 10 secondes.
   — Créer un script qui lance une musique à 07h00 du matin (il est 01h00...).

Exercice 5 – Se ressourcer

1. Répéter un traitement un certain nombre de fois peut être réalisé avec une boucle. Soit une variable lst définie par :

   ```
   lst=(1 2 3 4)
   ```

   Vérifiez le contenu de lst (avec `echo $lst`). Expliquer la différence entre les résultats des deux commandes suivantes :
   — `for i in $lst ; do echo $i ; done`
   — `for i in "$lst" ; do echo $i ; done`
2. Créer un script `loop.sh` qui prend en argument un nombre N et qui :
   — Crée un fichier *bench.txt* ;
   — Ajoute N lignes contenant “Hello world” dans ce fichier ;
   — Supprime ce fichier.
   Pour programmer ce script, vous avez besoin de ces nouvelles notions :
   — Quand vous appellez un script avec un argument, par exemple `./loop.sh 100`, l'argument 100 est récupérable à l'intérieure du script dans une variable nommée $1 pour le premier argument et, en général, $\$N$ pour le Nème argument.
   — On ajoute dans un fichier (sans écraser ce qu'il y a déjà dedans) avec l'opérateur >>.
   — `for i in {1..4} ; ...` pour répéter un traitement 4 fois.
3. Créer un autre script qui appellera le précédent N fois avec comme arguments 1, puis 10, puis 100, ...
4. Un aspect parfois important lorsqu'on programme est de vérifier la rapidité d'exécution d'un programme. En modifiant le script précédent, mesurer et afficher le temps d'exécution de chaque itération (commande `time`).
5. Créer un script qui collecte automatiquement les résultats obtenus dans le script précédent dans un fichier. Une ligne du fichier contient `N tmp` où N est le nombre de “Hello world” et tmp est le temps d'exécution.
6. Finalement vous utilisez GNU plot (commande `plot`) pour afficher ces résultats dans un graphique.
7. Encore du temps ? Essayer de rendre ce script de *benchmark* générique pour qu'il puisse être utilisé dans d'autres projets, avec d'autres programmes, paramètres, ... Peut-être veut-on plus d'information également comme la consommation mémoire. Soyez créatif !