SYRES – TP Unix et Shell

Centrale Nantes – Option Informatique

23 novembre 2015

Quelques commandes utiles :
— `man` : obtenir de l'aide sur une commande (p.ex. : `man ls`) ;
— `|` : enchaîner des commandes (p.ex. : `cat coin.txt | grep "meuh" | head`) ;
— Programmation structurée du shell : `if, while, for`, etc.
— Filtres :
  — `read` : lire une ligne sur l'entrée standard (clavier) ;
  — `head` : ne garder que les premières lignes ;
  — `tail` : ne garder que les dernières lignes ;
  — `sort` : trier des lignes ;
  — `uniq` : traiter des lignes en doublon ;
  — `grep` : filtrer des lignes selon une expression régulière ;
  — `sed` : traitements complexes sur des lignes ;
  — `awk` : traitements complexes sur des lignes structurées (en colonnes p.ex.) ;
— Réseau :
  — `wget` : télécharger des fichiers ;
  — `curl` : transferts de données complexes suivant des protocoles réseaux ;
— Opérations sur les fichiers :
  — `cp, mv, rm` : copier, déplacer (ou renommer), effacer un fichier ;
  — `touch` : créer un fichier vide ou modifier les dates de création, modification d'un fichier ;
  — `find` : trouver les fichiers remplissant certains critères ;
  — `chmod` : changer les permissions (lecture, écriture, exécution d'un fichier ;
  — `cat` : afficher le contenu de fichiers en concaténant s'il y en a plusieurs ;
— Autres
  — `df` : obtenir des informations sur l'occupation du système de fichiers ;
  — `du` : obtenir des informations sur l'occupation de dossiers ;
  — `ps` : obtenir des informations sur les processus ;
  — `echo` : afficher une ligne de texte ;
  — `seq` : afficher une suite de nombres (p. ex. pour les boucles) ;
  — `wc` : compter le nombre de lignes, caractères ou octets ;
  — `kill` : envoyer un signal à un processus (p.ex. pour l'arrêter).

Gentille introduction

Question 0. *Choisir un endroit douillet sur le disque dur, y créer un dossier* `tp_syres`. La commande `cd` permet de se déplacer dans l'arborescence des fichiers. La commande `pwd` permet de savoir où l'on est. On rappelle que `..` représente le dossier parent, et que `.` représente le dossier actuel. Pour créer le dossier, il faut utiliser `mkdir`. *Comment faire pour l'effacer ?*

Question 1. *Créer un fichier* `coin.sh` *avec le contenu ci-dessous.* Utilisez votre éditeur de texte préféré mais ne faites pas de copier-coller du PDF.

```
#!/bin/bash

echo "coin !"
```

Question 2. *Rendre le fichier* `coin.sh` *exécutable avec la commande suivante (Posez-vous des questions : pourquoi u+x ? Dois-je le faire à chaque modification de coin.sh ?) :*

`chmod u+x coin.sh`

Question 3. *Exécuter le programme en faisant :*

`./coin.sh`

Question 4. *Modifier* `coin.sh` *selon ce qui suit, exécuter* :

```
#!/bin/bash

cris="coin meuh bzzz"
for cri in $cris
do
    echo $cri
done
```

Question 5. *Modifier* `coin.sh` *selon ce qui suit, exécuter (attention au sens particulier des guillemets ‘) :*

```
#!/bin/bash

dir="/bin"
fichiers=‘ls $dir‘
for f in $fichiers
do
    echo "Dans $dir il y a $f"
    if [ $f == "cat" ]
    then
        echo "Miaou!"
    fi
done
```

Question 6. Il y a beaucoup de fichiers dans `/bin`. Pour voir tout ce qui est affiché, et détecter un éventuel ”Miaou !”, exécuter ainsi :

`./coin.sh | more`

ou un peu plus évolué (flèches pour se déplacer et ’q’ pour quitter)

`./coin.sh | less`

ou pour ne garder que les lignes contenant ”Mia”

`./coin.sh | grep Mia` **在结果中筛选出满足RE的结果**

Question 7. *À partir de ce qui précède, si possible sans utiliser de variable, et en utilisant la commande* `seq`*, écrire un script qui affiche 10 fois ”coin !”.*

Question 8. On peut *rediriger* la sortie standard (c’est-à-dire l’écran) vers un fichier. Par exemple :

```
echo "coin" > meuh.txt
```

Écrire une commande qui copie un fichier texte sans utiliser la commande `cp`*.*

À noter au passage, qu'on peut ajouter à un fichier existant (`>>`), rediriger la sortie erreur (`2>`), ou encore rediriger l'entrée standard (`<`) pour lire des choses depuis un fichier.

Exercice 1

On s'intéresse ici à quelques tâches simples.

Question 1. On suppose qu'on dispose d'un fichier `entete.txt` qui contient un en-tête (par exemple, un rappel de la license du logiciel, ou des auteurs).

Écrire un script qui ajoute cet en-tête à tous les fichiers C contenus dans le dossier courant.

Question 2. *Écrire un script qui donne les différences (commande `diff`) entre les 50 premières lignes de deux fichiers dont les noms sont donnés en paramètres*

Question 2′. *Même question mais sans utiliser de variables ou fichiers intermédiaires.*

Question 3. Max OS X ajoute dans tous les dossiers un fichier `.DS_Store` qui mémorise la position des icônes, le choix d'une image de fond, etc. Avec un autre OS ces fichiers n'ont pas d'intérêt.

Donner une commande pour supprimer récursivement tous les fichiers `.DS_Store` *dans le dossier courant et tous les sous-dossiers accessibles (à n'importe quelle profondeur).*

Question 4. *Écrire un script qui affiche la liste des mots contenus dans un fichier donné sur l'entrée standard (par redirection). Chaque mot affiché ne doit apparaître qu'une seule fois.*

Question 5. *Écrire un script qui, dans un fichier donné sur l'entrée standard (par redirection), remplace (et affiche à l'écran) les mots de 4 lettres, ceux qui commencent par un c ou finissent par un n, par le mot « coin » et tous les autres par le mot « bla ». La ponctuation et les espaces resteront inchangées.*

Exercice 2

Le but de cet exercice est de trouver des fichiers potentiellement effaçables sur les partitions où la place commence à manquer.

Question 1. *Écrire un script qui indique si des partitions (de type ext4 p.ex.) sont pleines à plus de 88%.*

Question 2. *Modifier le script précédent pour qu'il donne la liste des fichiers « normaux » (ni un répertoire ni un périphérique) appartenant à l'utilisateur courant, qu'il peut lire et écrire et dont la taille est supérieure à 1Mo.*

Question 3. Même question avec les fichiers musicaux (sans préjuger de leur format ou de leur extension).

Exercice 3

Le but de cet exercice est de détecter les processus devant éventuellement être arrêtés car ils utilisent trop de ressources.

Question 1. *Écrire un script qui compte le nombre de processus « actifs » (non endormis) de l'utilisateur courant.*

Question 2. *Ajouter l'indication du processus qui consomme le plus le processeur. Proposer à l'utilisateur de l'arrêter. En cas de réponse positive, arrêter le processus.*

Question 3. *Même question que précédemment mais avec le processus qui consomme le plus de mémoire.*