TD séance n° 8 et 9
Redirections et Traitements par lots

4 Exercices

Nous allons donc faire quelques exercices pour mettre en pratique l'utilisation des redirections et voir ainsi la puissance de ce mécanisme. Et nous finirons par un exercice montrant la puissance du scripting Shell à savoir faire du traitement par lot sur un ensemble de fichiers (où sur les données qu'ils contiennent).

Exercice n° 1:

Pour les différentes questions de cet exercice, vous donnerez la commande et indiquerez ce que contient le fichier après l'exécution.

- Redirigez le résultat de la commande « `date` » dans le fichier `sortie.txt`

```
$ date > sortie.txt
```

- Ajoutez au fichier `sortie.txt` créé précédemment le résultat de la commande « `ls -l` »

```
$ ls -l >> sortie.txt
```

- Rediriger le résultat des erreurs produites par la commande « `ls *.com` » dans le fichier `erreur.txt`

```
$ ls *.com 2> erreur.txt
```

Exercice n° 2:

Faire un programme `msg_err.sh` qui écrit un premier message « Message normal » vers la sortie standard et un deuxième message « Message d'erreur » vers la sortie standard d'erreur.

Vérifiez que votre programme fait bien ce qu'il faut.

```
#!/bin/bash
echo "Message normal"
echo "Mon message d'erreur" 1>&2
Pour le vérifier : ./msg_err.sh 2> fichier_erreurs.txt
```

Exercice n° 3:

A l'aide d'un enchainement de commandes avec des redirections, compter le nombre de dossiers présents dans le dossier courant. Une fois que vous aurez trouvé la bonne solution, faites en un script `count_dir.sh`.

```
ls -l | grep "^d" | wc -l
```

Modifier ce programme `count_dir.sh` pour passer en paramètre le nom du dossier dans lequel faire ce calcul (c'est une manière d'automatiser ce que l'on vous a demandé de faire au TD3 quand vous avez du compter le nombre de dossier à la racine du système de fichiers).

```
# !/bin/bash
ls -l $1 | grep "^d" | wc -l
```

Puis pour lancer le programme:

```
count_dir.sh /
```

Exercice n° 4:

Ecrire un script simple `tri_photos.sh` qui va prendre `2` arguments : le nom d'un dossier contenant des photos `jpg` (vous utiliserez le dossier contenant une cinquantaine de photos que vous avez récupérées dans le fichier de ressource pour ce TD) et le nom d'un dossier dans lequel il va recopier ces photos en les triant et les renommant.

Voici quelques précisions pour créer votre script. Suivez bien ces étapes pour simplifier votre développement

1. Vous commencerez par récupérer la date de numérisation d'une image. Cette information est stockée dans le fichier image lui-même. Vous pouvez récupérer cette information grâce à la commande `exif` en spécifiant la valeur de tag `0x0132`. Faites déjà un test en lançant la commande dans votre interprète de commande avant d'inclure celle-ci-ci dans un script.

```
val=$(exif --tag=0x0132 "nom du fichier" | grep "Value:")
Il ne faut pas oublier les guillemet autour du nom dans le cas où celui-ci contient
des espaces.
```

TD séance n° 8 et 9
Redirections et Traitements par lots

2. A partir de cette extraction de l'information, commencez maintenant votre script en stockant dans des variables la date (sous la forme année-mois-jour), l'heure (sous la forme heure-minute-seconde) et l'année. Dans une première version de ce script, vous récupérerez le nom du fichier sur lequel faire le traitement en paramètre et vous afficherez le résultat sous la forme :

```
$ tri_photo.sh 00722_door_1680x1050.jpg
La photo du fichier "00722_door_1680x1050.jpg" a été prise en 2006 (précisement
le 2006-03-16) à 21-12-08
```

```
#!/bin/bash

val=`exif --tag=0x0132 "$1" | grep "Value:"`

date=`echo $val | cut -d ' ' -f 2 | tr ':' '-'`
time=`echo $val | cut -d ' ' -f 3 | tr ':' '-'`
year=`echo $date | cut -d '-' -f 1`

echo "La photo du fichier \"$1\" a été prise en $year (précisement le $date) à $time
```

3. Modifiez le résultat obtenu précédemment en n'affichant plus le message (mettez-le en commentaire) mais remplacez le par la création d'un dossier correspondant à l'année de prise de la photo. Ce dossier devra être créé dans le dossier de destination spécifié en deuxième paramètre de votre script.

```
$ tri_photo.sh 00722_door_1680x1050.jpg dest_dir
```

```
# echo "La photo du fichier \"$1\" a été prise en $year (précisement le $date) à $time

mkdir –p "$2/$year"
# Mettez bien le chemin "$2/$year" entre guillemets dans le cas ou le nom de dossier
spécifié est un nom avec un espace
```

4. Ajoutez une copie du fichier dans ce nouveau dossier en la renommant avec un nom du type année-mois-jour_heyre-minute-seconde.jpg.

```
# Créer une copie du fichier en changeant son nom
cp $1 "$2/$year/${date}_$hour.jpg"
```

5. Enfin nous allons apporter une dernière modification à notre script pour qu'il ne travaille pas sur un fichier mais sur tous les fichiers du dossier passé en premier paramètre. Pour effectuer cette opération, vous allez avoir besoin d'appliquer le traitement spécifiez jusqu'à présent sur l'ensemble des fichiers d'un dossier. Nous n'avons pas vu les structures de contrôle du Shell, mais cela consiste à faire une boucle sur l'ensemble des fichiers. Voici la syntaxe à utiliser pour réaliser cette opération :

```
for file in $(ls $1)
do
    ... # la variable file contiendra le nom de fichier suivant à chaque itération
done
```

```
#!/bin/bash

for file in $(ls $1)
do
    val=`exif --tag=0x0132 "$file" | grep "Value:"`

    date=`echo $val | cut -d ' ' -f 2 | tr ':' '-'`
    time=`echo $val | cut -d ' ' -f 3 | tr ':' '-'`
    year=`echo $date | cut -d '-' -f 1`

    mkdir –p "$2/$year"
    cp "$file" "$2/$year/${date}_$hour.jpg"
done
```

TD séance n° 8 et 9
Redirections et Traitements par lots

Et voilà un script qui vous fera économiser des heures de manipulations à la souris et ainsi effectuer pour vous une suite d'opérations répétitives qui peut facilement être automatisées. Vous pourrez aussi vous rendre compte dans le cadre du cours Environnement Informatique 2 que la programmation Shell simplifie ce type de traitements par lots ou sur le contenu de fichier par rapport à beaucoup de langages de programmation (vous obligeant à des opérations intermédiaires pour accéder au contenu d'un dossier ou d'un fichier).

Exercice n° 5:

Créez le fichier `untexte.txt` à l'aide d'une redirection au lieu de le faire avec un éditeur de texte.

```
cat > untexte.txt
carotte, patate...
...
^D
(Pour dire que l'entrée du texte est terminée, on appuie sur Ctrl-D)
```

Exercice n° 6:

Ecrire un script `words.sh` qui prend un argument (un nom de fichier) et qui affiche chaque mot du fichier, à raison d'un mot par ligne, chaque mot étant suivi par son nombre d'occurrences dans le fichier. Les mots doivent apparaitre triés alphabétiquement. Par exemple, si le fichier `untexte.txt` contient les lignes suivantes :

```
carotte, patate... carotte, poireau !
haricot; poireau; carotte; patate...
patate+patate=patate
```

alors, on a le comportement suivant :

```
$ ./words.sh untexte.txt
Occurrence des mots :
3 carotte
1 haricot
5 patate
2 poireau
```

Commencez par créer le fichier untexte.txt.

Afin de vous aider, voici les étapes à suivre pour décomposer le problème et arriver à le résoudre. On utilisera pour cela intensivement les pipes pour combiner le résultat de chacune des commandes qui réaliseront successivement les étapes suivantes :

1. lire le contenu de `untexte.txt`
   ```
   cat untexte.sh
   ```
2. sur le contenu du fichier `untexte.txt` (donc le résultat de la commande précédente), remplacer les ponctuations par des retours à la ligne
   ```
   cat untexte.sh | tr '[:punct:]' '\n'
   ```
3. sur le résultat précédent, supprimer les espaces et tabulations
   ```
   cat untexte.sh | tr '[:punct:]' '\n' | tr -d '[ \t]'
   ```
4. sur le résultat précédent, ne retenir que les lignes non vides
   ```
   cat untexte.sh | tr '[:punct:]' '\n' | tr -d '[ \t]' | grep -v "^$"
   ```
5. puis appliquer au résultat un tri sur la liste des mots obtenus
   ```
   cat untexte.sh | tr '[:punct:]' '\n' | tr -d '[ \t]' | grep -v "^$" | sort
   ```
6. et enfin ne retenir qu'un seul de ces mots en cas de doublon dans la liste précédente en affichant le nombre d'occurrences.
   ```
   cat untexte.sh | tr '[:punct:]' '\n' | tr -d '[ \t]' | grep -v "^$" | sort | uniq -c
   ```

Exercice n° 7:

TD séance n° 8 et 9
Redirections et Traitements par lots

On reprend l'exercice `words.sh` pour avoir le résultat suivant :

```
$ ./words.sh untexte.txt
Nombre de mots dans ce texte: 11
Nombre de mots différents dans ce texte: 4
Occurrence des mots :
   3 carotte
   1 haricot
   5 patate
   2 poireau
```

Voici une décomposition du problème :

1. On stocke le résultat de la liste complète des mots (sans supprimer les doublons) dans une variable `list`

```
list=`cat $1 | tr '[:punct:]' '\n' | tr -d '[ \t]' | grep -v "^$" | sort`
```

2. On calcule et affiche le nombre total de mots dans le texte ainsi que le nombre unique de mots

3. echo –n "Nombre de mots dans ce texte: "

```
echo $list | tr ' ' '\n' | wc -l
echo –n "Nombre de mots différents dans ce texte: "
echo $list | tr ' ' '\n' | uniq | wc -l
```

4. On exécute ensuite le programme obtenu à l'exercice précédent

```
cat untexte.sh | tr '[:punct:]' '\n' | tr -d '[ \t]' | grep -v "^$" | sort | uniq -c
```