3R-IN3 – Systèmes d'exploitation
TD 2

Xavier Hilaire
x.hilaire@esiee.fr

19 mars 2025

1 Séquence entière

La commande `test x -le y` permet de tester si l'entier `x` est plus petit ou égal à `y`. Et la commande `test x = y` teste si les chaînes de charactères `x` et `y` sont identiques. Sachant cela, écrivez un script `sequence.sh [-c t] debut fin` qui écrit tous les nombres entiers entre `debut` et `fin` inclus, un nombre par ligne. Si l'option `-c` est utilisée, alors le script devra afficher les nombres `debut`, `debut+t`, `debut+2*t`, ..., sans jamais dépasser `fin`.

Sol.:

```
#!/bin/sh

if test "$1" = "-c"; then
    t=$2
    shift 2
else
    t=1
fi

i=$1
while test $i -le $2; do
    printf "%d\n" $i
    i=$((i+t))
done
```

2 Mauvaises extensions

Exécutez la commande suivante dans un terminal :

```
curl https://perso.esiee.fr/~hilairex/3R-IN3/fichiers-TD2.tgz | tar xvpzf - -C /tmp
```

puis placez-vous dans `/tmp/fichiers-TD2`. Ce répertoire contient des fichiers d'images qui sont toutes au format JPEG. L'extension des noms de fichiers est parfois correcte (.jpg), mais il arrive aussi qu'elle soit fausse (.tif, .fig, .Z, ...), ou inexistante.

Ecrire un script qui corrige ce problème dans le cas général. Il recevra en unique paramètre un répertoire, censé contenir les fichiers à corriger (`/tmp/fichiers-TD2` dans le cas présent). Puis, pour chaque fichier de ce répertoire, écrira la ligne de commande nécessaire (sans la lancer) pour :

— renommer le fichier en `.jpg` si l'extension n'est pas la bonne
— l'ajouter s'il n'a pas d'extension

S'il n'a rien à faire car l'extension est la bonne, il l'écrira aussi.

Aide : pour comparer deux chaînes de caractères entre elles, on peut utiliser la commande `test` avec l'opérateur `=`. Et le `!` marque la négation (dans la commande `test` elle-même comme dans une condition en Bash). Comparez, par exemple, les codes de sortie de

```
test abc = abc
test ! xyz = abc
```

Exemple d'utilisation :

```
[xavier@localhost TD]$ ./rennomer.sh /tmp/fichiers-TD2
Je traite /tmp/fichiers-TD2/crabe 072.png -> mv /tmp/fichiers-TD2/crabe 072.png /tmp/fichiers-TD2/crabe 072.jpg
Je traite /tmp/fichiers-TD2/image_0030.tif -> mv /tmp/fichiers-TD2/image_0030.tif /tmp/fichiers-TD2/image_0030.jpg
Je traite /tmp/fichiers-TD2/image_0031.JFIF -> mv /tmp/fichiers-TD2/image_0031.JFIF /tmp/fichiers-TD2/image_0031.jpg
Je traite /tmp/fichiers-TD2/image_0032.jpg -> est OK, rien à lancer
Je traite /tmp/fichiers-TD2/image_0033.jpg -> est OK, rien à lancer
Je traite /tmp/fichiers-TD2/image_0067.jpg -> est OK, rien à lancer
Je traite /tmp/fichiers-TD2/image_0071.jpg -> est OK, rien à lancer
Je traite /tmp/fichiers-TD2/inconnu.jpg -> est OK, rien à lancer
Je traite /tmp/fichiers-TD2/insecte  .bmp -> mv /tmp/fichiers-TD2/insecte  .bmp /tmp/fichiers-TD2/insecte  .jpg
Je traite /tmp/fichiers-TD2/Libell -> mv /tmp/fichiers-TD2/Libell /tmp/fichiers-TD2/Libell.jpg
Je traite /tmp/fichiers-TD2/Une Libellile 010a.Z -> mv /tmp/fichiers-TD2/Une Libellile 010a.Z /tmp/fichiers-TD2/Une Libellile 010a.jpg
[xavier@localhost TD]$
```

Sol.:

```
#!/bin/sh

for fichier in "$1"/*; do
    echo -n "Je traite $fichier -> "
    if test "${fichier%%.*}" = "$fichier"; then # le fichier n'a pas d'extension
        echo mv "$fichier" "${fichier}.jpg"
    else
        # le fichier a une extension
        if test "${fichier##*.}" = jpg; then # c'est la bonne
            echo "est OK, rien à lancer"
        else
            echo mv "${fichier}" "${fichier%%.*}.jpg"
        fi
    fi
done
```

Une solution un peu plus élégante consiste à transformer les tests (if) en case, qui est bien adapté à ce problème, étant donné i) qu'il permet de confronter directement les noms de fichiers à des motifs, et ii) qu'il examine les cas dans l'ordre où on les lui présente :

```
#!/bin/sh
#

for fichier in "$1"/*; do
    echo -n "Je traite $fichier -> "
    case "$fichier" in
         *.jpg) # il est d'extension .jpg
                echo "est OK, rien à lancer"
                ;;
         *.*) # il a une extension, mais ce n'est pas .jpg
              echo mv "${fichier}" "${fichier%%.*}.jpg"
              ;;
```

```
        *) # il n'a pas d'extension du tout
           echo mv "$fichier" "${fichier}.jpg"
           ;;
    esac
done
```

Une dernière solution consiste à ne pas utiliser la résolution des noms de fichiers dans la ligne de la boucle for, mais de lui préférer un utilitaire comme find (ou ls). Cette dernière présentera l'avantage :

- de garantir qu'on ne traite que des fichiers avec le prédicat `-type f` de find, les autres entrées, (répertoires, liens symboliques, tubes, sockets, ...) n'étant pas examinés ;
- de garantir qu'on n'examine que le répertoire, et pas ses sous-répertoires avec le prédicat `-maxdepth 1` de find ;
- de briser une éventuelle limite du shell quant à la taille de sa ligne de commande : que se passe t'il si le répertoire à traiter contient des centaines de milliers de fichiers ? Leurs noms peuvent-ils tous être substitués sur la ligne de commande lorsque le shell doit exécuter le for ?

La mise en œuvre de find est toutefois plus délicate, car contrairement à la résolution des noms de fichiers, qui produits ses effets directement sur la ligne de commande, find produit les siens sur la sortie standard, à raison de 1 fichier par ligne.

On doit donc relire chaque ligne produite par find, et la traiter, à l'instar de ce qui a été fait au TP1 pour la recherche de doublons. Ce qui implique le remplacement de la boucle for par une boucle while, et aboutit au résultat suivant :

```
#!/bin/sh
#

find "$1" -maxdepth 1 -type f -print | while read fichier; do
    echo -n "Je traite $fichier -> "
    case "$fichier" in
*.jpg) # il est d'extension .jpg
    echo "est OK, rien à lancer"
       ;;
*.*) # il a une extension, mais ce n'est pas .jpg
    echo mv "${fichier}" "${fichier%%.*}.jpg"
    ;;
*) # il n'a pas d'extension du tout
    echo mv "$fichier" "${fichier}.jpg"
    ;;
    esac
done
```

Un point remarquable ici est que le while, à droite du tube, est démarré dans un sous-shell : il y a bien deux processus qui coopèrent, find qui produit ses fichiers trouvés, et la boucle while, qui est exécutée dans un sous-shell et non dans le shell courant.

Cette séparation aurait pu conduire à une difficulté supplémentaire si la boucle while avait du affecter des variables que le shell qui a lance le script aurait été amené à consulter ensuite : l'indépendance des processus fait que ces variables ne pouvaient pas être transmises directement. Le seul moyen de les passer aurait été par soit par fichier, soit par la sortie standard et l'utilisation d'une quote inversée.

3 Répertoires vides

La commande `test -d dir` permet de vérifier si `dir` est un répertoire. Et la commande `test x -ge y` de tester si l'entier `x` est plus grand ou égal à (ge = greater or equal to) `y`. Sachant cela, écrire un script shell qui fera les choses suivantes pour chacun des arguments qu'il recevra :

- Il commencera par vérifier si cet argument est un répertoire. Si ce n'est pas le cas, il affichera « ... n'est pas un répertoire, ignoré. » sur sa sortie d'erreur standard.
- Sinon, deux cas de figure possibles :
  - Soit le répertoire n'est pas vide : le script affichera alors « Le répertoire .. n'est pas vide, ignoré. » sur sa sortie d'erreur standard
  - Soit il l'est : le script affichera alors « Le répertoire ... est vide, voulez-vous le supprimer (o/n) ? » sur sa sortie standard. Si l'utilisateur répond `O` ou `o`, alors il supprime le répertoire en question ; sinon, il ne fait rien.

Le script devra sortir avec le code de retour 0 si tous les arguments passés étaient bien des répertoires, sinon avec le code 1. Exemple d'utilisation :

```
[xavier@localhost TD]$ mkdir /tmp/vide
[xavier@localhost TD]$ ./repertoires.sh /etc/passwd /etc/ /tmp/vide
/etc/passwd n'est pas un répertoire, ignoré.
Le répertoire /etc/ n'est pas vide, ignoré
Le répertoire /tmp/vide est vide, voulez-vous le supprimer (o/n) ?o
rmdir /tmp/vide
[xavier@localhost TD]$ echo $?
1
[xavier@localhost TD]$
```

Sol.:

```
#!/bin/sh
#

code=0
for rep; do
    if test -d "$rep"; then
        compte=$(find "$rep" 2> /dev/null | wc -l)
        if test $compte -ge 2; then
            echo "Le répertoire $rep n'est pas vide, ignoré"
        else
            echo -n "Le répertoire $rep est vide, voulez-vous le supprimer (o/n) ?"
            read ans
            if test "$ans" = "o" || test "$ans" = "O"; then
                echo "rmdir $rep"
            fi
        fi
    else
        echo "$rep n'est pas un répertoire, ignoré."
        code=1
    fi
done
exit $code
```

4 Find parallèle

Ecrire un script shell qui recevra un nombre quelconque d'arguments, censés être tous des répertoires. Puis, pour chacun de ces répertoires, fera en parallèle les choses suivantes :

— Il lancera une commande `find -type d` sur ce répertoire.

— La sortie standard de ce find sera redirigée vers le fichier `/tmp/find-xx.txt`, et la sortie d'erreur standard vers `/tmp/find-xx.log` où `xx` est un entier (pas forcément à 2 chiffres) correspondant au numéro de l'argument correspondant

Le script devra sortir avec un code de retour égal au nombre de commandes find qui n'ont pas elles-mêmes terminé avec un code de retour nul. Exemple de sortie :

```
[xavier@localhost TD]$ ./parafind.sh /etc /var /tmp/vide
J'attends le PID 14784
J'attends le PID 14785
J'attends le PID 14786
[xavier@localhost TD]$ echo $?
2
[xavier@localhost TD]$ ls -l /tmp/find-*
-rw-rw-r-- 1 xavier xavier   1028 18 mai   19:56 /tmp/find-1.log
-rw-rw-r-- 1 xavier xavier  90715 18 mai   19:56 /tmp/find-1.txt
-rw-rw-r-- 1 xavier xavier   9861 18 mai   19:56 /tmp/find-2.log
-rw-rw-r-- 1 xavier xavier 147288 18 mai   19:56 /tmp/find-2.txt
-rw-rw-r-- 1 xavier xavier      0 18 mai   19:56 /tmp/find-3.log
-rw-rw-r-- 1 xavier xavier     10 18 mai   19:56 /tmp/find-3.txt
[xavier@localhost TD]$
```

Sol.:

```
#!/bin/sh
#

i=1
liste=""
for rep; do
    find "$rep" > /tmp/find-${i}.txt 2> /tmp/find-${i}.log &
    liste="$liste $!"
    i=$((i+1))
done

for p in $liste; do
    echo "J'attends le PID $p"
    wait $p
    if test $? -ge 1; then
        erreurs=$((erreurs+1))
    fi
done

exit $erreurs
```