ESIEE année 2008 – 2009
IN3R21 – Systèmes d'exploitation

TP n°2
Recherche de fichiers dupliqués

RAPPEL IMPORTANT : les comptes-rendus (au format Word97 ou texte simple) sont à remettre par e-mail avec accusé de réception à x.hilaire@esiee.fr avant la diffusion par e-mail de la correction de ce TP (cf. échéancier du polycopié distribué en cours).

Le but du TP est d'écrire un script qui détecte les fichiers doublons sur disque. Vous aurez besoin des fichiers utilisés lors du TP n°. Si vous avez supprimé ces fichiers : `tar –xvpzf ~hilairex/public_html/IN3R21-TPx-fichiers.tgz –C $HOME`

1) Considérez l'exemple de ligne de commande suivant :

```
fic=/usr/bin/ldd
```

Sans utiliser de commande externe, comment pouvez-vous faire afficher le chemin (`/usr/bin`) et le nom du fichier proprement dit (`ldd`) de `fic` séparément ?

2) Créez un nouveau script `~/bin/formater` (avec emacs), et copiez-y le code shell suivant :

```
#!/bin/sh
#
read line
while [ "$line" ]; do
    echo "J'ai lu: $line"
    read line
done
```

Posez les droits nécessaires pour que ce fichier puisse être exécuté à partir du shell ; puis observez `l'effet de cat /etc/passwd | formater`

Que fait ce script ?

3) La commande `find` liste récursivement les fichiers à partir d'un ou plusieurs répertoires passés en arguments.

Observez le résultat produit par `find ~/fichiers-tp /etc /var | less` (la dernière commande ayant pour effet de vous permettre de consulter le résultat page par page)

Modifiez le script précédent pour qu'il réécrive la liste des fichiers trouvés par `find` sous la forme *`nom / chemin`*. Par exemple :

```
$ find ~/fichiers-tp | formater
fichiers-tp / /Users/xavier
doublon1 / /Users/xavier/fichiers-tp
fichier.ext / /Users/xavier/fichiers-tp
html / /Users/xavier/fichiers-tp
…
```

Quel est l'intérêt d'utiliser la séquence « / » (espace-shlash-espace) comme délimiteur ?

4) Créez un deuxième script `~/bin/filtrer` qui affiche les lignes **consécutives** de l'entrée standard sous la forme gauche / droite, pour lesquelles le premier champ est identique à celui de la ligne immédiatement précédente. **L'entrée standard sera supposée triée**. Exemple :

```
$ cat double
un / ignorer
deux / rep1
deux / rep2
deux / rep3
trois / passer
quatre / xx
cinq / yyy
cinq /  /
six / fini
$ cat double | filtrer
deux / rep2
deux / rep3
cinq /  /
$
```

5) Créez un troisième script `~/bin/doublons` avec emacs, et recopiez-y le code shell suivant :

```
#!/bin/sh
#
echo "$# arguments ont ete passes au script"
echo "arg0 = $0"
echo "arg1 = $1"
echo "arg2 = $2"
echo '$* vaut' $*
echo $* | wc
```

Observez le comportement de ce nouveau script en lui passant 2, puis 3, 4,… arguments.

A présent, observez l'effet de la commande sort sur un fichier texte, par exemple : `sort /etc/passwd` ou `cat /etc/passwd | sort`.

Modifiez le fichier `doublons` pour qu'il appelle **`find`** sur les arguments qu'il a reçus, et écrive

sur la sortie standard les noms des fichiers qui ont été listés plus d'une fois.

6) On souhaiterait que `doublons` n'inclue par défaut que les fichiers listés par `find` qui correspondent à des fichiers ordinaires. Mais on voudrait aussi laisser à l'utilisateur la possibilité d'inclure en plus des fichiers ordinaires les liens symboliques et les répertoires s'il le souhaite, à l'aide de deux options L, et d :

Un appel à `doublons –L répertoires…` doit inclure aussi les liens symboliques
Un appel à `doublons –d répertoires…` doit inclure aussi les répertoires

Les options sont cumulatives et peuvent être passées dans n'importe quel ordre : par exemple, un appel à `doublons –d –L rep` équivaut à `doublons –L –d rep` et doit inclure à la fois les liens symboliques et les répertoires. L'utilitaire `find` admet un prédicat `–type` qui peut être combiné par des OU et ET logiques et qui répond précisément à votre besoin. Par exemple :
`find . –type d –or –type l`

filtrera les entrées trouver pour ne laisser passer que les répertoires et liens symboliques (consultez la page manuel de `find` si nécessaire). Modifiez votre script pour qu'il se comporte de la manière qui vient d'être dite. De plus, si l'utilisateur donne une option invalide, le script devra afficher un message d'erreur et terminer avec un code d'erreur > 0. Sinon, il devra terminer avec le code d'erreur de `find`.

7) On souhaiterait utiliser un critère supplémentaire pour considérer que deux fichiers sont des doublons : leurs tailles doivent être identiques. Listez le fichier `/etc/passwd` avec **`ls`**, et repérez sa taille. Observez maintenant l'effet des commandes suivantes :

```
stat -c "%s" /etc/passwd
v=$(stat -c "%s" /etc/passwd)
echo $v
```

Comment pouvez-vous modifier `formater,` et éventuellement `filtrer,` pour intégrer ce nouveau critère ?

8) Ecrivez un script ou une fonction (au choix) `trouver` prenant d'éventuelles options et un répertoire en arguments et produisant exactement le même résultat que l'appel à **`find`** modifié à la question 6), mais sans utiliser **`find`** (indication : `test -d` , `test -f,` et `test -h`).
Le script devra tester le nombre d'arguments transmis : si ce nombre ne vaut pas un, il affichera un message d'erreur et terminera avec le code de retour 1. Sinon, il affichera les fichiers trouvés et terminera avec le code de retour 0.