3R-IN3 – Systèmes d'exploitation
TP1

Xavier Hilaire
x.hilaire@esiee.fr

19 mars 2025

1 Recherche de fichiers dupliqués

Pour cette partie, vous devrez en premier exécuter la ligne de commande suivante :

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

Elle devrait créer une arborescence de fichiers `fichiers-tp` directement rattaché à votre répertoire par défaut. Cette arborescence vous servira à tester la suite du TP.

1.1 La commande read

Examinez ce que fait la fonction read du shell sur l'exemple suivant, en saisissant une ligne au clavier après la ligne read :

```
read var
echo $?
echo $var
```

Faites la même chose, mais au lieu de saisir une ligne, fermez l'entrée standard avec la combinaison de touche Ctrl+D. Conclusion ?

Sol.: La variable spéciale '?' reflète le code d'erreur renvoyé par read. Si ce code est positif, cela veut dire que la commande a échoué. Cela est vrai pour n'importe quelle autre commande (voir cours). Si la lecture s'est bien passée, le code sera nul, et var contiendra ce qui a été lu.

1.2 Script de formatage

Créez un nouveau script `~/bin/formater` (avec gedit, nano, emacs, ou l'éditeur texte de votre choix), et copiez-y le code suivant :

```
#!/bin/sh
#

while read line; do
    echo "J'ai lu: $line"
done
```

Nous n'avons pas encore vu l'instruction `while` en cours, mais le code parle de lui-même... En utilisant `chmod`, modifiez les droits de `~/bin/formater` pour que vous puissiez exécuter votre script [1]. Puis observez l'effet de `ls ~ | formater`

Que fait `formater` ?

Sol.:

```
chmod u+x formater
export PATH="$PATH:$HOME/bin"
ls ~ | formater
```

Le script permet simplement de lire l'entrée standard ligne par ligne.

1.3 Find et printf

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, on en sort d'ailleurs en appuyant sur 'q')

Par ailleurs, la commande `printf` de Linux fait la même chose que son homonyme en langage C, à cette différence près que tous ses arguments étant passés depuis le shell, ce sont des chaînes de caractères. Exemple :

```
[hilairex@pc5352a TP]$ printf "%d %o %x %s \t %d %u\n\n" 51 51 51 "du texte" -1 -1
51 63 33 du texte   -1 18446744073709551615

[hilairex@pc5352a TP]$
```

Remarquez que les caractères `'\n'` et `'\t'` sont bien interprétés correctement.

Sachant cela, modifiez votre script formater pour qu'il réécrive la sortie produite par find sous la forme : chemin d'accès complet + 1 tabulation + nom du fichier seul. Voici par exemple les 6 premières lignes que j'ai obtenues sur ma machine (la commande head -n 6 permet de limiter aux 6 premières lignes) :

```
[hilairex@pc5352a TP]$ find ~/fichiers-tp/ | formater  | head -n 6
/home/hilairex/fichiers-tp/
/home/hilairex/fichiers-tp/doublon1 doublon1
/home/hilairex/fichiers-tp/fichier.ext fichier.ext
/home/hilairex/fichiers-tp/html html
/home/hilairex/fichiers-tp/html/lien.html lien.html
/home/hilairex/fichiers-tp/html/node128.html node128.html
[hilairex@pc5352a TP]$
```

Sol.:

```
#!/bin/sh
#

while read fic; do
     printf "${fic}\t${fic##*/}\n"
done
```

1. Si vous avez fait le TD jusqu'au bout, vous avez normalement créé le répertoire `~/bin` et étendu votre variable `PATH` pour qu'elle l'inclue

1.4 Sort et uniq

La commande `sort` trie son entrée standard ligne par ligne, par ordre alphabétique. Consultez sa page de manuel, en particulier l'option `-k`. Ce que ne dit pas forcément le manuel, c'est que le délimiteur par défaut de sort est la tabulation – celle justement utilisée à la question précédente...

Par ailleurs, la commande `uniq` repère les lignes répétées de son entrée standard, à condition qu'elles soient adjacentes. Consultez sa page manuel.

Sachant cela, étendez la ligne de commande précédente pour n'afficher que les fichiers dont le nom se rencontre plus d'une fois. Voici la sortie que j'obtiens sur ma machine de travail :

```
[hilairex@pc5352a TP]$ find ~/fichiers-tp/ | ./formater  | ...
/home/hilairex/fichiers-tp/doublon1 doublon1
/home/hilairex/fichiers-tp/repertoire/doublon1 doublon1
/home/hilairex/fichiers-tp/fichier.ext fichier.ext
/home/hilairex/fichiers-tp/repertoire/fichier.ext fichier.ext
/home/hilairex/fichiers-tp/repertoire/rep2/fichier.ext fichier.ext
/home/hilairex/fichiers-tp/html/repertoire repertoire
/home/hilairex/fichiers-tp/repertoire repertoire
[hilairex@pc5352a TP]$
```

Recopiez et créez un lien d'un fichier quelconque vers un autre répertoire. Il devra être trouvé par votre commande.

Sol.:

```
[hilairex@pc5352a TP]$ find ~/fichiers-tp/ | ./formater  | sort -k2 | uniq -d -f1
/home/hilairex/fichiers-tp/doublon1 doublon1
/home/hilairex/fichiers-tp/fichier.ext fichier.ext
/home/hilairex/fichiers-tp/html/repertoire repertoire
[hilairex@pc5352a TP]$
```

1.5 Script final

Finalement, créez un nouveau script `~/bin/doublons` qui affichera ces lignes en lançant find sur l'ensemble des répertoires qu'il aura reçus en arguments. Les éventuelles erreurs produites par find devront être neutralisées par le script lui-même (elles ne devront pas apparaître sur votre terminal).

Sol.:

```
#!/bin/sh
#

find "$@" 2>/dev/null | ./formater  | sort -k2 | uniq -d -f1
```

2 Redirections

Avec un éditeur de votre choix, créez le script `decompte` suivant :

```
while test "$1"; do
  echo "Le paramètre 1 vaut $1"
  shift
done
```

Essayez alors de lancer `./decompte un deux trois qutre` après avoir fait un chmod approprié sur le fichier. Quel est l'effet de shift ?

Modifier à présent le script, qui recevra au moins deux paramètres :
— Le premier sera le nom d'un fichier de sortie
— Le deuxième et les suivants seront des répertoires

Votre script devra ajouter une ligne de la forme « Il y a n fichiers dans le répertoire *machin* ». De plus, s'il rencontre des erreurs (parce que le répertoire n'existe pas ou n'est pas lisible), elles ne devront jamais apparaître à l'écran ; et en lançant votre script deux fois de suite, le fichier de sortie ne devra contenir que les informations relatives à la dernière exéuction.

Exemple d'utilisation :

```
[hilairex@pc5352a TP]$ ./decompte /tmp/sortie /etc /usr
[hilairex@pc5352a TP]$ cat /tmp/sortie
Il y a 296 fichiers dans /etc
Il y a 12 fichiers dans /usr
[hilairex@pc5352a TP]$ cat /tmp/sortie
Il y a 6 fichiers dans /home
[hilairex@pc5352a TP]$
```

Sol.:

```
#!/bin/sh
#

sortie=$1
shift
> $sortie
while test "$1"; do
    total=$(ls "$1" 2> /dev/null | wc -l)
    echo "Il y a $total fichiers dans $1" >> $sortie
    shift
done
```

3 Quote inversée

3.1 Calculette

Ecrire un script `somme` qui lit une série de nombres entiers à partir de l'entrée standard (1 nombre par ligne), puis affiche leur somme. Exemple d'utilisation :

```
[hilairex@pc5352a TP]$ ./somme
12
5
7
8
Je pense que 12+5+7+8 = 32
[hilairex@pc5352a TP]$
```

Aide :

— La commande `tr src dst` lit son entrée standard et remplace chaque caractère de la chaine `src` par celui à la même position dans `dst`. Par exemple, `tr abcde 12345` remplacera les 'a' par des '1', les 'b' par des '2', etc.

— Il existe un script de 2 lignes répondant à ce problème

Sol.:

```
#!/bin/bash
#

a=‘echo $(cat) | tr ' ' '+'‘
echo "Je pense que $a = $((a))"
```

3.2 Trafic réseau

Sans aucun argument, la commande `ifconfig` affiche des informations sur toutes les interfaces réseau de votre machine. Parmi les lignes produites, on trouve des lignes en RX packets, qui correspondent aux paquets reçus pour cette interface, et d'autres en TX packets.

Ecrire un script qui calcule la somme de tous les RX packets, et celle de tous les TX packets, et les affiche, en n'appelant `ifconfig` qu'une seule fois. Exemple d'utilisation :

```
[hilairex@pc5352a TP]$ ./packets
J'ai trouvé 3113043 RX packets, et 4228013 TX packets sur votre machine
[hilairex@pc5352a TP]$
```

Aide :

— La commande `grep chaine` filtre l'entrée standard pour ne retenir que les lignes qui contiennent la `chaine` spécifiée

— La commande `cut -c`*x*`-` ne retient que les les colonnes `x` et suivantes de l'entrée standard

— La commande `cut -f`*n* `-d' '` ne retient que le n-ième champ de l'entrée standard, en supposant que le délimiteur de champs est le caractère espace ' '

Sol.:

```
#!/bin/bash
#

ifconfig > /tmp/ifconfig
rx=$(echo $(grep "RX packet" < /tmp/ifconfig | cut -c20- | cut -f1 -d' ' ) | tr ' ' '+')
tx=$(echo $(grep "TX packet" < /tmp/ifconfig | cut -c20- | cut -f1 -d' ' ) | tr ' ' '+')
echo "J'ai trouvé $((rx)) RX packets, et $((tx)) TX packets sur votre machine"
```