Module GNU/Linux CEFIPA : TP n°3 - Shell avancé
Corrigé

Nicolas Burrus

26 Avril 2005

Question 1

Créer un fichier `$HOME/programs` contenant la liste des fichiers de `/bin`.

Réponse :

```
$ ls /bin > ~/programs
```

Question 2

Entrez la commande `ls /bin /nowhere`. Complétez la commande pour faire disparaître le message d'erreur. Quel est le code de sortie du programme ?

Réponse :

```
$ ls /bin /nowhere 2>/dev/null
$ echo $?
1
```

Le `2>/dev/null` redirige la sortie d'erreur du `ls` dans `/dev/null`, un fichier spécial qui absorbe tout sans prendre d'espace mémoire.

Le code de sortie est différent de 0 ici, ce qui montre que `ls` a détecté un problème.

Question 3

Donnez une commande qui appelle `mkdir` pour créer le répertoire `$HOME/dir1` uniquement s'il n'existe pas.

Réponse :

```
$ ls ~/dir1 || mkdir ~/dir1
  ou
$ test -d ~/dir1 || mkdir ~/dir1
```

Question 4

Donnez la liste des fichiers dont le nom contient la chaîne "syslog" dans `/var`.

Réponse :

```
$ find /var -name '*syslog*'
```

Attention à bien mettre des quotes, sinon les étoiles sont interprétées par le shell et pas par `find`.

Question 5

Cherchez le PID du processus `init` en combinant `ps` et `grep`.

Réponse :

```
$ ps axo pid,command | grep " init " | grep -v grep
    1 init [2]
```

Le `grep -v grep` sert à ignorer la ligne du ps contenant `grep " init "`. Les espaces entre les quotes servent à discriminer entre init et kdeinit par exemple.

Bonus : pour n'obtenir que le PID en sortie, et pas toute une ligne de ps, on peut utiliser la commande suivante :

```
$ ps axo pid,command | grep " init " | grep -v grep | sed -re 's/ +/ /g' | cut -f 2 -d '
1
```

Le sed permet de remplacer plusieurs espaces par un seul, et le cut extrait uniquement le deuxième champ de la ligne, les champs étant délimités par des espaces.

Question 6

Parcourez la liste des fichiers présents dans `/dev`, sans scroller dans le terminal, en utilisant `less`.

Réponse :

```
$ ls /dev | less
```

Question 7

Dans `$HOME`, créez les fichiers `fa.c`, `fb.c` et `fc.c`. En une seule ligne de commande, créez une copie de chacun de ces fichiers en rajoutant l'extension `.backup`. A la fin, vous devez avoir 6 fichiers : `fa.c`, `fa.c.backup`, `fb.c`, ...

Réponse :

```
$ for toto in *.c; do mv $toto $toto.backup; done
```

Question 8

En une seule ligne de commande, changez l'extension des fichiers `$HOME/f*.c` en `.txt`.

Réponse :

```
$ for toto in *.c; do mv $toto `echo $toto | sed -re 's/\.c$/.txt/'`; done
```

La commande entre backquotes sera remplacée par la sortie de son exécution par le shell avant d'appeler le mv.

Question 9

Cherchez la chaîne "networking" sans tenir compte de la casse dans tous les fichiers présents dans `/etc` et ses sous-répertoires.

Réponse :

```
$ grep -ri networking /etc
```

Question 10

Donner une commande qui affiche les 3 premiers répertoires de `/dev`, en excluant `/dev` lui même de la sortie. Exemple :

```
/dev/rd
/dev/ida
/dev/ataraid
```

Réponse :

```
$ find /dev -type d | grep -vE '/dev$' | head -n 3
```

Le find cherche tous les répertoires et sous répertoires de `/dev`, le grep enlève l'entrée correspondant au `/dev`, le head ne garde que les 3 premières lignes.

Question 11

Copiez le fichier `/etc/passwd` en `$HOME/mypasswd`. Donnez une commande qui permet de ne garder que le nom de l'utilisateur sur chaque ligne, préfixé par "user : ". Exemple :

```
$ cat mypasswd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
$ commande magique
$ cat mypasswd
user: root
user: daemon
user: bin
```

Réponse :

```
$ sed -re 's/:x:.*//' mypasswd | sed -re 's/^/user: /' > mypasswd.tmp
$ rm mypasswd
$ mv mypasswd.tmp mypasswd
```

Le premier sed remplace tout ce qui suit le nom de l'utilisateur par rien, le deuxième remplace le caractère de début par `user :` et place le résultat dans un fichier temporaire. Etant donnée que le premier sed lit les caractères dans le fichier mypasswd, il serait dangereux d'écrire la sortie directement dedans, sans être certain que le premier sed soit terminé.

Question 12 (bonus)

Donnez une commande permettant de compter le nombre de lignes de vrai code dans un fichier source C++. Vous devez donc ignorer les lignes vides et les lignes commençant par '//'. Exemple :

```
$ cat > foo.cpp
#include <iostream>

// I love useless comments
// to make meaningless lines of code

int main()
{
  // With two space before, it is more difficult.
  std::cout << "Hello World" << std::endl;
}
$ commande-magique foo.cpp
5
```

Réponse :

```
$ grep -vE " *//" foo.cpp | grep -v "^$"
```

Le premier grep enlève les commentaires, éventuellement précédés d'espaces, et le deuxième enlève les lignes vides (lignes constituées du caractère de début et directement du caractère de fin de ligne à coté.