TD : scripts bash

Combinaison de commandes

Manipulation de la date

La commande date affiche la date et l'heure avec le format suivant :

```
Mer 23 fév 2011 15:00:42 CET
```

- Utilisez l'instruction `cut` et un pipe pour n'afficher que la date (ici le 23)
- Utilisez l'instruction `cut` et un pipe pour n'afficher que l'heure complète
- Utilisez deux instructions `cut` et des pipes pour n'afficher que l'heure.

Occupation du système de fichier

Vous allez écrire un groupement de commandes (reliées par des `|`) pour rechercher les 5 plus gros répertoires contenus dans votre répertoire.

Utilisez la commande `du` avec les bons arguments pour afficher la taille des sous répertoires présents sur votre compte.

Triez les ensuite par taille décroissantes. Pour trier des données formatées Human Readable (comme 12k, 32M, …) vous devez utiliser l'option `-h`

Affichez ensuite les 5 plus gros répertoires.

Modifiez ensuite votre commande pour que les erreurs de la commande du soit redirigées vers `/dev/null`. Placez vous ensuite à la racine du système (`cd /`) et exécutez de nouveau votre commande pour afficher les 5 plus gros répertoires du systèmes.

Extraction de l'information

La commande `ping` permet de tester une connexion réseau. L'affichage obtenu est le suivant :

```
PING g2inf012s.adaux.iut21.u-bourgogne.fr (192.168.5.12): 56 data bytes
```

```
64 bytes from 192.168.5.12: icmp_seq=0 ttl=63 time=1.203 ms

64 bytes from 192.168.5.12: icmp_seq=1 ttl=63 time=0.613 ms

64 bytes from 192.168.5.12: icmp_seq=2 ttl=63 time=0.928 ms

64 bytes from 192.168.5.12: icmp_seq=3 ttl=63 time=0.563 ms

^C

--- g2inf012s.adaux.iut21.u-bourgogne.fr ping statistics ---

4 packets transmitted, 4 packets received, 0% packet loss

round-trip min/avg/max/stddev = 0.563/0.827/1.203/0.258 ms
```

Par défaut la commande `ping` ne s'arrête jamais, pour l'arrêter appuyer sur la touche `CTRL` et la touche `C` en même temps (d'où le caractère `^C` qui apparaît).

L'option `-cXX` permet d'envoyer `XX` requêtes puis la commande s'arrête normalement (donc sans `^C`). Ecrivez un groupement de commande qui permet d'afficher le temps moyen (sur 10 essais) de connexion vers le serveur `pedago`. L'information est contenue dans la dernière ligne, il s'agit de la valeur `avg`.

Explorateur de fichiers

Ecrire un script qui précise le type de fichier passé en paramètre et les permissions d'accès pour l'utilisateur. La commande `whoami` retourne le nom de l'utilisateur. La commande `ls -L` (qui peut être effectuée sur un fichier) à l'affichage suivant :

```
-rw-r--r-- 1 jb users   85405 janv. 11 07:58 TP1.pdf

-rw-r--r-- 1 jb users   68671 janv. 15 06:12 TP2.pdf

drwxr-xr-x 1 jb users      68 janv. 15 22:13 TP3

-rw-r--r-- 1 jb users 1119106 janv. 16 10:48 TP3.pages

-rw-r--r-- 1 jb users   78647 janv. 16 10:48 TP3.pdf
```

La commande `file` retourne le type du fichier avec le formatage ci-dessous :

```
jb@Linux ~ $ file TP2.pdf

TP2.pdf: PDF document, version 1.3
```

Le script doit par exemple, afficher :

```
Le fichier ./TP3 est un répertoire

./TP3 est accessible par jb en lecture et ecriture



Le fichier ./TP2.pdf est un fichier ordinaire de type PDF document, version 1.3

Le propriétaire de ./TP2.pdf est jb

./TP3 est accessible par jb en lecture et ecriture
```

A partir du script précédent, réalisez un script qui parcours un répertoire donné par l'utilisateur en paramètre d'appel et qui affiche le nombre de fichier pour chaque type (PDF, images, …). Pour cela, remplissez un fichier texte avec le type retourné par `file` puis triez le et comptez les répétitions pour chaque ligne. Vous devez obtenir un affichage comme ci-dessous :

Le répertoire contient :

```
      5  OpenDocument Presentation

      2  Zip archive data, at least v2.0 to extract

      1  UTF-8 Unicode text

      1  Bourne-Again shell script, ASCII text executable

      1  ASCII text
```

Construction aléatoire d'une arborescence

Eléments utiles

Dans un script bash, la variable `RANDOM` (donc lue avec `$RANDOM`) contient une valeur aléatoire comprise entre 0 et 32767 (voir le détail sur http://stackoverflow.com/questions/1194882/generate-random-number).

On rappelle qu'il est possible de mettre des variables bout-à-bout (la concaténation) de manière très simple :

```
a="fichier"

b=4

mkdir $a$b
```

Le programme ci-dessus va créer le répertoire `fichier4`.

Travail à réaliser

Ecrivez un script qui construit un nombre aléatoire (compris entre 1 et 10) de répertoire. Dans chacun de ces répertoires il doit créer un nombre aléatoire de fichiers (compris entre 1 et 10). Pour réaliser des boucles utilisant des variables pour les bornes vous pouvez regarder : http://unix.stackexchange.com/questions/55392/in-bash-is-it-possible-to-use-an-integer-variable-in-the-loop-control-of-a-for

Les fichiers font entre 1 et 25 MiB Pour éviter des problèmes, créez vos fichiers dans le répertoire `/tmp`.

Création des fichiers

Nous allons créer des fichiers contenant des valeurs aléatoires avec la commande `dd`. Cette commande permet de copier des données d'un fichier vers un autre. Elle est très puissante mais aussi très « destructrice » en cas d'erreur. Avant de faire quelques « bêtises » jetez un coup d'oeil sur http://doc.ubuntu-fr.org/dd. Avant de lancer la commande, demandez à l'enseignant de valider votre commande si vous avez un doute (sinon, c'est à vos risque et périls !!!).

La ligne suivante permet de créer un fichier de 10 MiBi :

```
dd if=/dev/urandom of=/tmp/fichier.tmp bs=1M count=10 1>/dev/null 2>/dev/null
```

Votre programme aura donc 2 boucles imbriquées : la première construisant les répertoires et la seconde construisant les fichiers.

Lorsque ce programme fonctionne, copiez quelques fichiers d'un répertoire vers un autre (vous avez donc des fichiers en double).

Recherche de doublons

Le but de cet exercice et rechercher les fichiers qui sont identiques (qui contiennent les même données) dans les répertoires que vous venez de créer.

Calcul d'une empreinte

La commande `md5sum` permet de calculer l'empreinte d'un fichier. Elle retourne une valeur hexadécimale unique au fichier (c'est-à-dire que si 2 fichiers ont la même empreinte, ils sont identiques).

La commande `find . -name "*" -type "f"` retourne tous les fichiers présents dans le répertoire courant avec l'arborescence complète (donc les fichiers contenus dans les sous répertoires).

Commencez par écrire un script qui calcule la somme MD5 pour tous les fichiers présents dans le répertoire (avec l'arborescence complète). Modifiez ensuite votre script pour que le résultat soit stocké dans un fichier (par ex. `sommes_md5`)

Recherche des doublons

Dans un premier temps travaillez cette partie dans le terminal, vous l'incorporerez à votre programme lorsqu'elle sera fonctionnelle.

En utilisant `sort`, `cut`, `uniq`, … rechercher toutes les sommes de contrôle qui sont répétées dans votre fichier.

Placez le résultat dans un fichier.

Recherche des noms de fichiers

Dans le fichier obtenu précédemment vous n'avez donc que les signatures des fichiers qui sont en doubles (par ex.) :

```
e343c2f5d0c3965c4719475152740466

080052359c06f3ff4df577684be2c3d9

59df848bef27f8372de87d9bed2680ba

abdfd9718633ddfb5b66b491aa88d7f6
```

Il reste à trouver les fichiers qui correspondent à cette signature. La solution la plus simple consiste à parcourir le tableau élément par élément avec une boucle `for` et rechercher la présence de l'élément dans le fichier de signature. La commande `grep motif fichier` permet de rechercher la chaîne de caractères `motif` dans le `fichier`.

Le résultat de la commande grep doit être mis en forme (avec les `tr`,`cut`, …) pour qu'à la fin du programme l'affichage soit proche du suivant :

```
Les fichiers suivants sont identiques :

        ./rep3/fichier_3

        ./rep5/fichier_3

Les fichiers suivants sont identiques :

        ./rep3/fichier_4

        ./rep5/fichier_4

Les fichiers suivants sont identiques :

        ./rep3/fichier_0

        ./rep5/fichier_0
```