TP6 - Programmation en shell
ING1-GI-GM – Commandes Unix
Année 2017–2018

EISTI

Traitement des données

1 Écrivez un script qui permet un affichage trié d'un fichier de note, sans trier l'entête (première ligne). □

Fichiers et répertoires

2 Dans un répertoire, on a des fichiers. On désire réaliser un script qui déplace tous les fichiers dans des sous répertoires. Pour ce faire, on extrait les extensions des fichiers, puis on crée des répertoires ayant comme nom la valeur des extensions. Enfin, on déplace l'ensemble des fichiers correspondants dans les sous répertoires adéquats. Réaliser ce script. Voici un exemple :

```
> ls ./
tata.adb toto.adb toto.ads toto.tex

> reparti_fichier
> ls ./
adb ads tex

> ls -R ./
./:
adb ads tex

./adb:
tata.adb toto.adb

./ads:
toto.ads

./tex:
toto.tex
```

□

Gestion des arguments d'entrée

3 En utilisant la fonction `getopts` (aide : `man bash`), étendez le script précédent pour qu'il traite les arguments suivants :

- `-h`, `-help` : affiche de l'aide sur la commande.
- `-d rep` : lance le script sur le répertoire `rep`.
- `-v` : mode *verbose*. Il faudra faire une fonction qui, si on active ce paramètre, montre plus d'information sur la démarche du script.

□

Gestion des données avec *sed*

4 Soit le fichier contenant les lignes :

```
asterix ; 20
troll de troy ; 8
rubrique à brac ; 46
Lanfeust de troy ; 8
autre / diver ; 30
```

Avec la commande `sed`, faites les modifications suivantes (elles seront traitées indépendamment les unes des autres) :

— remplacer les lettres **`;`** par un **`!`**

— remplacer les lettres **`t`** en début de ligne par un **`T`**

— remplacer les lettres **`/`** précédé d'un espace par une virgule

— remplacer les lettres **`/`** par un **`+`** et les lettres **`;`** par un **`@`**

— ne pas afficher les lignes 2 à 4

Rappel : dans **sed**, la forme de la commande de substitution est : `s/remplacé/remplacement/modifiers`
Ensuite, exécuter les commandes suivantes :

```
sed -e 's/\(^.\)/\1\1/g' fichier
sed -e 's/\(^.\)/\1\1\1\1/g' fichier
```

□

Gestion d'utilisateurs

5 Écrivez un script qui teste si un nom proposé par l'utilisateur est un login ou pas tant qu'il le souhaite. Si le login existe, le script doit montrer le(s) groupe(s) auquel il appartient. □

Détection de doublons

Nous allons écrire un ensemble de scripts. Ces scripts s'appelleront entre eux. De plus, ces scripts vont réaliser un parcours d'arborescence. Il est donc indispensable de pouvoir accéder à l'ensemble des scripts depuis un endroit quelconque de l'arborescence.

6 Écrire un script qui prend en paramètre un fichier, et qui affiche son nom, sa taille, son *md5sum*, et son *cksum*. Les deux dernières informations sont obtenues en appliquant les fonctions `md5sum` et `cksum` sur le fichier. □

7 Écrire un script qui prend en paramètre un répertoire, et qui applique le script de la question 6, sur tous les fichiers présents dans ce répertoire. □

8 Écrire un script qui prend en paramètre un nom de script et un répertoire, et qui appelle le script passé en paramètre sur tous les sous répertoires du répertoire passé en paramètre. □

9 Écrire un script qui affiche l'ensemble des informations suivantes : *nom*, *taille*, *md5sum*, *cksum*. Vous devez afficher les informations pour tous les fichiers présents dans l'arborescence à partir de l'endroit où l'on appelle le script. □

10 Écrire un script pour détecter s'il existe des doublons. On considérera qu'un fichier est doublon d'un autre si son nom, sa taille, son *md5sum* et son *cksum* sont identiques. Vous devez vérifier tous les fichiers présents dans l'arborescence à partir de l'endroit où l'on appelle le script. □