M1 BIO-INFO
X1BI030

Université de Nantes
2021/2022

Feuille d'exercices n° 1
Programmation BASH

BASH 1

Exercice 1.1

- En s'aidant de la commande `man`, utiliser la commande `ls` pour afficher les fichiers du répertoire courant en ordre décroissant de la taille des fichiers ;
- Afficher à l'écran le nom du fichier de plus grande taille dans le répertoire courant ;
- Créer dix répertoires `D1`, `D2`, …, `D10` si le répertoire `D0` peut être créé localement ;
- Créer un sous-répertoire `tmp`. Que fait la commande

  ```
  bash -c "cd tmp; pwd"
  ```

  Que se passe t'il après l'exécution de cette commande ? Expliquer.
- Lancer firefox si aucune instance n'est actuellement exécutée. Afficher le chemin d'accès au fichier `.parentlock` se trouvant dans votre arborescence.

Exercice 1.2

1. Écrire un script *Bash* créant un fichier `poids-age.dat2` à partir du fichier `poids-age.dat1` (disponible sur *madoc*) ;

   `poids-age.dat1` *

   ```
   poids:34,age:9
   poids:18,age:4
   poids:46,age:13
   poids:8,age:1
   poids:47,age:12
   ```

   `poids-age.dat2` *

   ```
   poids age
   34 9
   18 4
   46 13
   8 1
   47 12
   ```

2. Écrire un script *Bash* créant un fichier `poids-age.dat3` (tri par âge croissant) à partir du même fichier de départ `poids-age.dat1` ;

   `poids-age.dat1`

   ```
   poids:34,age:9
   poids:18,age:4
   poids:46,age:13
   poids:8,age:1
   poids:47,age:12
   ```

   `poids-age.dat3`

   ```
   poids age
   8 1
   18 4
   34 9
   47 12
   46 13
   ```

3. Utiliser le programme R ci-dessous pour créer le graphique Postscript de la figure 1 à partir du fichier `poids-age.dat3` ;

   ```
   # Fichier R
   postscript("poids-age.eps",onefile=FALSE);
   data=read.table("poids-age.dat3",header=TRUE);
   plot(data$age,data$poids,xlab="Age",ylab="Poids",type="b");
   ```

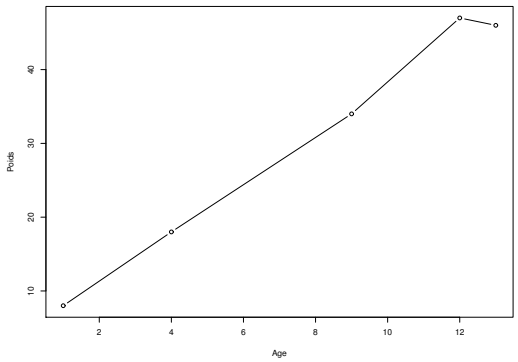

FIGURE 1 – Courbe Poids/âge

Exercice 1.3

Écrire les commandes bash effectuant les manipulations suivantes :

- Déplacement de tous les fichiers dont le nom commence par p et se termine par une voyelle du répertoire `/home/dupont/temp` vers le répertoire `/tmp` ;
- Copie des fichiers ayant l'extension `.txt` du répertoire courant vers le répertoire immédiatement au-dessus ;
- Effacement de tous les fichiers dont le nom se termine par `~` ou par `.bak` dans le répertoire racine de durand.

BASH 2

Exercice 1.4

- Écrire le script demandant son nom à un utilisateur et affichant « `Bonjour nom` » ;

Exercice 1.5

- Écrire un script qui permet d'imprimer les valeurs données comme arguments.

  Exemple d'appel : `imprimearg toto 45 janvier`
- Écrire un script qui affiche ligne par ligne le contenu d'un fichier dont le nom est fourni en paramètre ;
- Écrire un script qui compte le nombre de fichiers ordinaires présents dans le répertoire courant ;
- Trouver le fichier `.parentlock` se trouvant dans un sous-répertoire de `$HOME/.mozilla` et le supprimer ;
- Écrire un script qui liste tous les noms de fichiers présents dans le répertoire courant qui ont une extension `.txt` et qui commencent par `scri`.

Exercice 1.6

Itérer une lecture clavier jusqu'à l'obtention de la chaîne `oui` ou de la chaîne `non`.

Exercice 1.7

- Écrire un script qui ajoute le droit en lecture à tous les fichiers ayant l'extension `.html` dans le répertoire courant ;
- Écrire un script qui affiche les fichiers du répertoire courant par ordre de taille ;
- Supprimer du répertoire courant tous les fichiers dont le nom possède l'extension `.txt` sauf le fichier `jules.txt`.

Exercice 1.8

Écrire un programme bash demandant en boucle un nombre à l'utilisateur et lui indiquant s'il est plus grand ou plus petit qu'un nombre mystère. On affichera le nombre de coups nécessaires pour trouver le nombre.

Exercice 1.9

Écrire un programme bash prenant trois paramètres et effaçant dans le répertoire donné par le premier paramètre tous les fichiers se terminant par l'extension donnée par le deuxième paramètre, sauf le fichier dont le nom est donné par le troisième paramètre (sans le chemin d'accès)
Le programme devra afficher de l'aide s'il est appelé avec moins de trois paramètres
Exemple d'utilisation : `effacer /home/durant txt toto.txt`

Exercice 1.10

Écrire un programme bash affichant la liste de tous les fichiers Python (extension `.py`) se trouvant dans l'arbre des répertoires dont la racine est passée en paramètre.

Exercice 1.11

Écrire la commande bash renommant tous les fichiers Postscript du répertoire courant en remplaçant l'extension `.ps` par l'extension `.postscript`.
Note : On pourra utiliser la commande `basename` prenant un nom de fichier et un suffixe et retournant sur `stdout` le nom sans le suffixe s'il y apparaissait (sinon, le nom non modifié est retourné). Exemples : `basename toto.txt .txt` retourne `toto` ; `basename toto.pas .txt` retourne `toto.pas`.

Exercice 1.12

Écrire le programme bash affichant le i^e terme de la suite de Fibonacci, i étant donné en paramètre. Le programme affichera un message d'erreur si i n'est pas un nombre strictement positif.
Rappel : La suite de Fibonacci est définie par :

$$\begin{cases} F_1 = F_2 = 1 \\ F_n = F_{n-2} + F_{n-1}, \quad \text{pour } n \geq 3 \end{cases}$$

Exercice 1.13

Écrire le programme bash qui, pour chaque fichier `nom.eps` du répertoire `/home/dupont/eps` copie le fichier `nom.fig` se trouvant dans le répertoire `/home/dupont/fig` dans le répertoire `/home/durant/fig`.

Exercice 1.14 (Regexp et Bash)

1. Afficher toutes les lignes d'un fichier qui contiennent un nom de couleur (bleu, rouge, vert, orange) et un nom d'oiseau (perroquet, cacatoès, pluvier) ;
2. Afficher toutes les lignes d'un programme C contenant une affectation ;
3. Afficher les variables d'un programme C faisant l'objet d'une affectation. On triera les noms de variables par ordre alphabétique.

Exercice 1.15 (Regexp et Bash)

1. Écrire une commande `sed` extrayant le nom d'un fichier de son chemin d'accès ;
2. Écrire un programme bash utilisant la commande `sed` pour émuler la commande `basename`. On définira une fonction `error()` chargée d'afficher les messages d'erreur éventuels et de quitter le programme.

Exercice 1.16 (Regexp et Bash)

Un fichier contient du texte et des dates écrites au format `AAAA-MM-JJ` (ex. : 2009-01-23). Écrire un script modifiant le fichier de façon à utiliser le format `JJ-MM-AAAA`.

Exercice 1.17 (Bash)

1. Écrire un programme utilisant la commande `convert` pour créer des vignettes de taille 100×100 à partir des fichiers JPEG passés en paramètres. Les vignettes devront être sauvées dans le sous-répertoire `thumbnails/` du répertoire des images (à créer s'il n'existe pas) ; elles seront au format PNG et leur nom sera composé du nom du fichier original suivi de la taille finale (voir exemple ci-dessous) ;

```
# Etat après exécution de la commande
.
|-- chat.jpg
|-- cheval.jpg
|-- lapin.jpg
|-- thumbnails
|   |-- chat_100x100.png
|   |-- cheval_100x100.png
|   `-- lapin_100x100.png
`-- tortue.gif
```

2. Ajouter un contrôle vérifiant que les fichiers passés sont bien des images JPEG (utilisation de `file`).

Exercice 1.18 (Bash)

Écrire le programme bash permettant de retrouver l'index d'une commande se trouvant dans l'*history* à partir d'une expression régulière passée en paramètre. Si plusieurs commandes sont possibles, on retournera la plus récente.

Exercice 1.19 (Bash)

Écrire un programme pour générer un index à partir d'un document texte contenant les marqueurs `@index{txt}`.

Exercice 1.20 (Bash)

On souhaite écrire une commande `del` permettant de sauvegarder dans une poubelle les fichiers effacés pour pouvoir les restaurer si le besoin s'en fait sentir.

1. Écrire la commande `del` déplaçant un ou plusieurs fichier(s) à effacer dans le répertoire `$HOME/.dustbin`. La commande devra afficher un message d'erreur pour chaque fichier pour lequel il existe déjà un homonyme dans `$HOME/.dustbin`;
2. Amélioration 1 : la commande `del` doit désormais déplacer les fichiers sous un nom unique dans le répertoire `$HOME/.dustbin`.

   Exemple : sauver la première occurrence du fichier `toto.txt` en `$HOME/.dustbin/toto.txt_0000`, la deuxième occurrence en `$HOME/.dustbin/toto.txt_0001`, etc. On affichera à l'utilisateur les noms sous lesquels sont sauvés les fichiers;
3. Amélioration 2 : modifier la commande `del` et écrire une commande `restaure` pour autoriser la restauration d'un fichier effacé dans le répertoire où il se trouvait à l'origine. L'utilisateur peut utiliser `restaure` de trois manières :
   (a) S'il tape seulement `restaure`, on affiche la liste des fichiers restaurables dans le répertoire courant;
   (b) S'il tape `restaure ''NOMFIC''`, on restaure le fichier `NOMFIC` qui était auparavant dans le répertoire courant;
   (c) S'il tape `restaure ''CHEMIN/NOMFIC''`, on restaure le fichier `NOMFIC` qui était auparavant dans le répertoire `CHEMIN`.