

**Exercice 1.** Écrire un script qui affiche le contenu de tous les fichiers dont les noms sont passés en paramètre en précisant le nom de chaque fichier avant l’affichage de son contenu. (On ne se préoccupe pas du problème de défilement).

**Exercice 2.** Écrire un script qui copie tous les fichiers d’extension `.txt` du répertoire courant en des fichiers de même nom et d’extension `.txt.archive` (toujours dans le répertoire courant).

**Exercice 3.** Écrire un script qui prend un nombre indéterminé de paramètres et calcule leur somme.

**Exercice 4.** Écrire un script qui affiche pour chaque fichier du répertoire courant une ligne de la forme `<nom_fichier> taille`.

**Exercice 5.** Écrire un script qui boucle en demandant à l’utilisateur de saisir deux variables, affiche leur somme, leur différence, leur produit et leur quotient. Le programme s’arrête lorsque l’utilisateur saisit un point.

**Exercice 6.** Écrire un script qui prend un argument et en affiche la factorielle. Vérifier que le nombre de paramètres est correct.

**Exercice 7.** Écrire un script qui prend deux arguments et affiche leur pgcd. Vérifier que le nombre de paramètres est correct. Afficher un message spécifique quand l’un des deux arguments est nul.

**Exercice 8.** Écrire un script qui prend en argument des noms de fichiers. Une option `-v` peut également être placée parmi les arguments. Si l’option est présente, la commande affiche les caractéristiques de chaque fichier et répertoire (format de `ls -l` sans afficher le contenu des répertoires). Si l’option n’est pas présente, la commande renomme chacun des fichiers (ou répertoires) en ajoutant l’extension `.old`.

**Exercice 9.** Écrire un script qui sort les valeurs successives de la suite définie par :

$$u_{n+1} = \begin{cases} u_n/2 & \text{si } u_n \text{ est pair,} \\ 3 * u_n + 1 & \text{si } u_n \text{ est impair.} \end{cases}$$

La valeur de  $u_0$  est donnée en argument, le programme s’arrête lorsque la valeur à afficher est 1.

**Exercice 10.** Écrire une commande `jeter` qui permet de manipuler une poubelle de fichiers – un répertoire – nommée `poubelle` et située à votre racine. La commande accepte a trois options :

- ♦ `jeter -l` pour lister le contenu de la poubelle ;
- ♦ `jeter -r` pour vider la poubelle ;
- ♦ `jeter fichier1 fichier2 ...` pour déplacer les fichiers considérés vers la poubelle.

Si la poubelle n’existe pas, elle est créée à l’appel de la commande.

**Exercice 11.** Écrire une commande qui prend en argument un nombre quelconque de noms de fichier et affiche pour chacun :

- son nom si c’est un fichier régulier exécutable ;
- la liste des fichiers exécutables qu’il contient si c’est un répertoire.

**Attention.** Le script ne doit pas être récursif, afin de d’obtenir un ordre d’affichage « raisonnable ». Il ne s’agit en aucun cas d’explorer les sous-répertoires.

**Exercice 12.** Écrire une commande qui prend deux arguments, les bornes d’un intervalle, et qui affiche les valeurs du polynôme  $2x + 1$  pour toutes les valeurs entières situées dans cet intervalle (borne inférieure comprise, borne supérieure excluse).

Afficher des messages spécifiques si :

- le nombre de paramètres est incorrect ;
- les arguments ne sont pas des entiers ;
- l’intervalle est incorrect.

**Exercice 13.** Écrire une commande qui prend en argument un nombre quelconque de noms de fichier et affiche pour chacun :

- son nom si c’est un fichier régulier non exécutable suivi de la mention « est un fichier non executable » ;
- son nom si c’est un fichier régulier exécutable suivi de la mention « est un fichier executable »,
- la liste de tous les fichiers réguliers exécutables d’extension « .truc » qu’il contient si c’est un répertoire.