LISV Laboratoire d'ingénierie des systèmes de Versailles
UVSQ université PARIS-SACLAY

Linux : TP 5

Halim Djerroud

révision 1.0

Exercice 1 : Script avec arguments

Écrire un script qui prend deux nombres en arguments et affiche :
— leur somme,
— leur produit,
— et le plus grand des deux.

Exercice 2 : Boucles avec seq

Écrire différents scripts utilisant les boucles et la commande `seq` :

1. Afficher tous les nombres pairs de 1 à 50.
2. Afficher uniquement les multiples de 3 de 1 à 50.
3. Afficher la somme des nombres de 1 à 100 (utiliser une variable pour accumuler).
4. Afficher la table de multiplication de 7 (de 7x1 à 7x10).
5. Demander à l'utilisateur un nombre n et afficher la table de multiplication de n.
6. Afficher la suite des carrés parfaits de 1 à 10 (1, 4, 9, 16, ..., 100).
7. Bonus : écrire un script qui affiche un triangle de nombres :

```
1
1 2
1 2 3
1 2 3 4
...
```

Exercice 3 : Tests de fichiers

Écrire un script qui demande à l'utilisateur de saisir un nom de fichier ou de répertoire et qui affiche :

1. Si l'élément existe ou non.
2. Si c'est un fichier régulier, un répertoire, ou un lien symbolique.
3. Si l'élément est lisible, modifiable, et exécutable par l'utilisateur.
4. Si c'est un fichier régulier : afficher sa taille en octets.
5. Si c'est un répertoire : afficher le nombre de fichiers qu'il contient.
6. Bonus : si c'est un lien symbolique, afficher la cible du lien.

Exercice 4 : Analyse d'un répertoire

Écrire un script qui prend un nom de répertoire en argument et qui affiche :
— le nombre de fichiers contenus,
— le nombre de sous-répertoires,
— le fichier le plus gros (nom + taille).

Exercice 5 : Gestion de processus

Écrire différents scripts autour de la gestion de processus :

1. Lancer la commande `sleep 100` en arrière-plan et afficher son PID.
2. Attendre 5 secondes puis tuer ce processus.
3. Modifier le script pour vérifier que le processus existe encore avant de l'arrêter.
4. Lancer deux commandes `sleep 50` et `sleep 60` en arrière-plan, afficher leurs PID et les tuer tous les deux.
5. Bonus : écrire un script qui lance une commande passée en argument en arrière-plan et qui affiche :
   — le PID,
   — son état (`ps -p PID -o stat=`),
   — sa consommation CPU et mémoire (`ps -p PID -o %cpu,%mem=`).

Exercice 6 : Variables d'environnement

Écrire un script qui manipule les variables d'environnement :

1. Crée un répertoire `binPerso/` dans le dossier personnel.
2. Ajoute ce répertoire au `PATH` (temporairement).
3. Vérifie que ce nouveau chemin est bien pris en compte.
4. Crée dans `binPerso/` un petit script `bonjour.sh` qui affiche « Bonjour $USER ».
5. Rends ce script exécutable et teste qu'il peut être lancé depuis n'importe quel répertoire.
6. Affiche la valeur de variables d'environnement classiques : `HOME`, `USER`, `PWD`, `SHELL`.
7. Bonus : proposer une solution pour que la modification du `PATH` soit permanente (par exemple en modifiant le fichier `.bashrc`).

Exercice 7 : Redirections et logs

Écrire un script qui manipule les redirections :

1. Exécuter une commande passée en argument et :
   — rediriger sa sortie standard vers `stdout.log`,
   — rediriger sa sortie d'erreur vers `stderr.log`.
2. Ajouter à la fin de `stdout.log` le message : « Commande terminée avec succès » si la commande retourne 0.
3. Ajouter à la fin de `stderr.log` le message : « Erreur lors de l'exécution » si la commande échoue.
4. Créer également un fichier `global.log` qui contient à la fois la sortie standard et la sortie d'erreur (fusion des flux).
5. Bonus : ajouter un en-tête dans chaque fichier de log indiquant la date et la commande exécutée.

Mini-projet : Script de sauvegarde (backup.sh)

Écrire un script `backup.sh` qui automatise une sauvegarde simple :

1. Vérifie que deux arguments (source et destination) sont donnés, sinon affiche un message d'utilisation.
2. Vérifie que le dossier source existe, sinon affiche une erreur.
3. Vérifie que le dossier destination existe, sinon le crée.
4. Copie tous les fichiers `.txt` du dossier source vers le dossier destination.
5. Crée un fichier `backup.log` dans le dossier destination contenant :
   — la date et l'heure de l'opération,
   — la liste des fichiers copiés avec leur taille,
   — le nombre de fichiers copiés,
   — la taille totale des fichiers sauvegardés.
6. Ajoute au log une ligne de séparation entre chaque sauvegarde.
7. Affiche un message final de confirmation.
8. Bonus : proposer une option `-a` pour sauvegarder tous les fichiers (pas seulement les `.txt`).