Scripts shell

F. Lassabe

Contents

1 Introduction

Ce TP se base sur les scripts shell qui sont similaires à des programmes écrits en bash (ou tout autre version de shell). Dans notre cas, nous travaillerons en bash. Les scripts shell sont très utilisés pour les opérations d'administration systèmes afin de réaliser des opérations trop complexes pour être réalisées par une commande seule. Les possibilités de ces scripts sont très larges. Les scripts étant assimilables à des programmes en bash, pour cette session de TP, vous ne répondrez pas directement dans le PDF, mais vous déposerez les fichiers contenant vos scripts dans l'espace de dépôt de Moodle. Vous nommerez les scripts Exercice$N$.sh où N est le numéro de l'exercice dans le document. Lorsqu'un exercice avance par étapes, vous ne déposez sur Moodle que la dernière version du script.

2 Exercice 1

Écrire un script qui affiche le contenu du répertoire courant. Le tester, puis ajouter un argument qui sera le chemin de l'élément à lister. Vous accédez au premier argument d'un script par la variable $1.

3 Exercice 2

Écrire un script qui affiche le nombre de ses arguments. Rajouter l'affichage de la liste des arguments, d'abord dans l'ordre, puis dans l'ordre inverse. Le nombre de paramètres d'un script est défini par la variable $#

4 Exercice 3

Écrire un script avec deux fonctions, nommées *exists* et *usage*. La fonction *exists* teste si le chemin passé en argument existe. La fonction *usage* affiche l'usage du script (c'est-à-dire la façon de l'utiliser, notamment le fait qu'un paramètre soit nécessaire). Tester les fonctions en les appelant l'une après l'autre. Le script génère une erreur si vous ne lui passez pas d'argument. Modifier le script pour qu'il teste que le nombre d'arguments passés est égal à 1. Dans ce cas, appeler la fonction *exists* sur l'argument, dans le cas contraire, appeler la fonction *usage*.

5 Exercice 4

Écrire un script qui génère un mot de passe aléatoire dont on spécifie la longueur en argument. Si l'argument de longueur n'est pas donné, la longueur sera de 16. Vous pouvez utiliser le contenu de **/dev/urandom** en filtrant les caractères que vous souhaitez utiliser, ou utiliser la fonction **md5sum** sur une entrée aléatoire (par exemple le résultat de **ps aux**).

6 Exercice 5

Écrire un script qui permet de trouver des fichiers en double dans une arborescence donnée. Pour le tester, utiliser le dossier fourni avec ce sujet (attention, il est basé sur le même que la fois précédente, mais il est différent, il faut donc télécharger cette version). Pour trouver les doublons, il faut suivre le principe suivant :

- Parcourir les dossiers dans le dossier *Photos*, et les stocker dans un tableau
- Pour chaque dossier, chercher dans les dossiers suivants (inutile de revenir en arrière puisque la comparaison de dossier n avec dossier $n+i, i > 0$ a déjà été faite quand dossier n était traité) si des fichiers existent avec le même nom dans le dossier courant et celui de comparaison.
  - Si c'est le cas, comparer leurs sommes md5. Si elles sont identiques, remplacer le fichier du second répertoire par un lien physique vers celui du premier répertoire.
  - Sinon, ne faites rien.