CPE Lyon - 4ETI, 3IRC & 3ICS - Année 2019/2020

Administration Système

TP 2 - Bash

Ce deuxième TP a pour but d'approfondir vos connaissances sur Bash, les variables d'environnement et l'automatisation de tâches via la programmation de scripts.

Exercice 1. Variables d'environnement

1. Dans quels dossiers bash trouve-t-il les commandes tapées par l'utilisateur ?
2. Quelle variable d'environnement permet à la commande **`cd`** tapée sans argument de vous ramener dans votre répertoire personnel ?
3. Explicitez le rôle des variables **`LANG`**, **`PWD`**, **`OLDPWD`**, **`SHELL`** et **`_`**.
4. Créez une variable locale **`MY_VAR`** (le contenu n'a pas d'importance). Vérifiez que la variable existe.
5. Tapez ensuite la commande **`bash`**. Que fait-elle ? La variable **`MY_VAR`** existe-t-elle ? Expliquez. A la fin de cette question, tapez la commande **`exit`** pour revenir dans votre session initiale.
6. Transformez **`MY_VAR`** en une variable d'environnement et recommencez la question précédente. Expliquez.
7. Créer la variable d'environnement **`NOMS`** ayant pour contenu vos noms de binômes séparés par un espace. Afficher la valeur de **`NOMS`** pour vérifier que l'affectation est correcte.
8. Ecrivez une commande qui affiche "Bonjour à vous deux, binôme1 binôme2 !" (où binôme1 et binôme2 sont vos deux noms) en utilisant la variable **`NOMS`**.
9. Quelle différence y a-t-il entre donner une valeur vide à une variable et l'utilisation de la commande **`unset`** ?
10. Utilisez la commande **`echo`** pour écrire **exactement** la phrase : **`$HOME =`** *chemin* (où chemin est votre dossier personnel **d'après bash**)

Programmation Bash

Vous enregistrerez vos scripts dans un dossier **`script`** que vous créerez dans votre répertoire personnel. Tous les scripts sont bien entendu à **tester**.

Ajoutez le chemin vers **`script`** à votre **`PATH`** de manière permanente.

Exercice 2. Contrôle de mot de passe

Écrivez un script **`testpwd.sh`** qui demande de saisir un mot de passe et vérifie s'il correspond ou non au contenu d'une variable `PASSWORD` dont le contenu est codé en dur dans le script. Le mot de passe saisi par l'utilisateur ne doit pas s'afficher.

Exercice 3. Expressions rationnelles

Ecrivez un script qui prend un paramètre et utilise la fonction suivante pour vérifier que ce paramètre est un nombre réel :

```
function is_number()
{
  re='^[+-]?[0-9]+([.][0-9]+)?$'

  if ! [[ $1 =~ $re ]] ; then
    return 1
  else
    return 0
  fi
}
```

Il affichera un message d'erreur dans le cas contraire.

Exercice 4. Contrôle d'utilisateur

Écrivez un script qui vérifie l'existence d'un utilisateur dont le nom est donné en paramètre du script. Si le script est appelé sans nom d'utilisateur, il affiche le message : "Utilisation : *`nom_du_script`* `nom_utilisateur`", où `nom_du_script` est le nom de votre script récupéré automatiquement (si vous changez le nom de votre script, le message doit changer automatiquement)

Exercice 5. Factorielle

Écrivez un programme qui calcule la factorielle d'un entier naturel passé en paramètre (on supposera que l'utilisateur saisit toujours un entier naturel).

Exercice 6. Le juste prix

Écrivez un script qui génère un nombre aléatoire entre 1 et 1000 et demande à l'utilisateur de le deviner. Le programme écrira "C'est plus !", "C'est moins !" ou "Gagné !" selon les cas (vous utiliserez `$RANDOM`).

Exercice 7. Statistiques

1. Écrivez un script qui prend en paramètres trois entiers (entre -100 et +100) et affiche le min, le max et la moyenne. Vous pouvez réutiliser la fonction de l'exercice 3 pour vous assurer que les paramètres sont bien des entiers.
2. Généralisez le programme à un nombre quelconque de paramètres (pensez à `SHIFT`)
3. Modifiez votre programme pour que les notes ne soient plus données en paramètres, mais saisies et stockées au fur et à mesure dans un tableau.

Exercice 8. Pour les plus rapides

Écrivez un script qui affiche les combinaisons possibles de couleurs (cf. TP 1) :


FG \ BG 40 41 42 43 44 45 46 47
Bash