**Variables et Scripts shell**

**Objectif**

L’objectif de ce TP est d’apprendre à définir et à manipuler des variables simples et des variables d’environnement. Dans une deuxième partie, ce TP constitue une initiation à la programmation shell.

Des exemples montrant l’utilisation des conditions et des boucles seront étudiés.

# Les variables du shell et les variables d'environnement

1. Les variables Unix, en particulier celles dites d’environnement, sont des variables que l'on peut utiliser dans n'importe quel programme.

A l’aide de la commande env, afficher le contenu de toutes les variables d’environnement. Quel signifie le contenu des variables SHELL et PATH ?

1. Pour examiner la valeur d'une variable d’environnement telle que PATH, exécuter la commande :

## $ printenv PATH

1. La valeur d’une variable peut également être récupérée en utilisant le caractère $ devant son nom. Exécuter la commande suivante pour afficher la valeur de la variable PATH:

### $ echo $PATH

1. L’affectation d’une valeur à une variable se fait comme suit: variable="valeur" (Il est important de noter qu’avec le shell bash, il ne doit pas y avoir d’espace avant et après le caractère =).

Définir une variable simple nommée var, et lui affecter la chaine "ceci est un exemple", avec la commande :

## $ var="ceci est un exemple"

1. A l’aide de la commande export, transformer la variable simple var en une variable d’environnement
2. Lister de nouveau les variables d’environnement et vérifier que celle-ci existe
3. Le caractère ' définit une chaîne sans évolution. Le caractère " définit une chaîne toute substitution faite. Le caractère ` définit une chaîne interprétée comme une commande. Exécuter les commandes suivantes et interpréter leurs résultats :

$ echo $var

$ var=$var"comment"

$ echo "ceci $var cela"

$ echo ‘ceci $var cela’

$ echo `pwd`= répertoire courant

1. Exécuter la commande unset pour effacer la variable var crée.
2. Il existe d’autres variables spéciales dans le shell bash, dont voici le nom et la description :

|  |  |
| --- | --- |
| Variable | Fonction |
| $\* | Contient la liste des arguments |
| $# | contient le nombre d'arguments |
| $? | contient le code de retour de la dernière opération |
| $0 | contient le nom du script |
| $n | contient l'argument n (n étant un nombre) |
| $! | contient le PID de la dernière commande lancée |

A l’aide de la commande vi créer un nouveau fichier exemple.sh dans votre répertoire personnel et y ajouter le contenu qui suit :

#!/bin/bash

echo "Nombre d’arguments est : "$# echo "Les arguments sont : "$\*

echo "Le deuxième argument est : "$2

1. Ajouter le droit d’exécution au fichier exemple.sh créé
2. Lancez ce script exemple.sh avec un ou plusieurs arguments, et interpréter le résultat

## $ ./exemple.sh a b c

1. Pour pouvoir utiliser le script exemple.sh créé en tapant directement son nom (sans le précéder par "./") depuis n'importe quel répertoire, il suffit d’ajouter le répertoire contenant ce script à la variable d’environnement PATH, en exécutant la commande :

## $ PATH=$PATH:$HOME

Ré-exécuter de nouveau le script en l’appelant directement par la commande

## $ exemple.sh a b c

1. Utiliser la commande read afin de lire une chaine de caractères depuis la ligne de commande et l’affecter à une variable, notée v :

## $ read v

« Chaine de caractères quelconque »

## $ echo $v

### Développement de scripts shell

1. Développer des scripts pour tester les exemples suivants
2. La commande expr permet de réaliser entre autres un calcul arithmétique sur des entiers.

Vérifier cela sur un exemple, puis exploiter les redirections 1> et >2 pour rediriger son résultat

1. Poser un filtre entre les commandes date et cut pour n’afficher que l’heure système sous la forme **hh:mm**
2. Quel est résultat de la commande suivante :

**$ ./monshell abc def -d -4 + 900 xyz** sachant que le script monshell contient :

**echo $# $3 $7 $\* $0**

b) Quel est le résultat de la commande précédente, si monshell contenait la commande shift 2 en première ligne.

1. Les fichiers de commande (Shell script):
2. Indiquer ce que réalise le script suivant :

if test $# -ne 2

then echo "erreur sur l\'appel de $0" else

if ! test -f $2

then echo "erreur, le deuxième parametre doit etre un fichier" else

if grep "$1" $2 >/dev/null 2>/dev/null then echo "$1 trouvé dans $2"

else echo "$1 non trouvé dans $2" fi

fi fi

1. Ecrire un shell script nommé answer qui affiche le texte d’une question à laquelle correspond une réponse de type Oui ou Non. Si la réponse est Oui, le script doit retourner la valeur 0 ; par contre si elle correspond à Non la valeur de retour sera 1. Si la réponse n’est ni Oui, ni Non, la question sera posée une nouvelle fois.
2. Créer quatre fichiers **file1, file2, file3, et file4**, sous votre répertoire personnel, en les remplissant par un contenu aléatoire. Créer par la suite un répertoire **dir**, et déplacer les deux fichiers **file3** et

**file4** dedans.

Ecrire trois scripts où vous comparez les deux fichiers file1 et file2 (vous pouvez faire appel à la commande diff)

1. Les noms des deux fichiers sont indiqués dans le script
2. Les noms des deux fichiers sont lus depuis le clavier
3. Les noms des deux fichiers sont donnés au script comme argument
4. Ecrire un script pour comparer tous les fichiers du répertoire dir au fichier file1. Le nom du répertoire et celui du fichier sont donnés comme argument au script.
5. Vous allez écrire trois scripts dont chacun prend en argument le répertoire **dir**, et qui fait un traitement sur tous les fichiers de ce répertoire pour
6. afficher pour chaque élément son type : fichier ou répertoire (script type.sh)
7. afficher l'information sur son propriétaire (script prop.sh)
8. afficher l'information sur sa date (script date.sh)
9. Ecrire un script qui compte le nombre de fichiers dans un répertoire donné.