Introduction aux systèmes d'exploitation (IS1)

TP 8 – Variables du shell

En plus des usages que vous avez vus jusqu'ici, le shell propose d'autres fonctionnalités qui en font un véritable langage de programmation. En particulier, comme en Java, il est possible d'utiliser des *variables*.

Les variables Elles possèdent un *identificateur*, qui peut être n'importe quelle suite de caractères commençant par une lettre ou '_' et ne contenant que des lettres, des chiffres ou '_'. Elles n'ont qu'un type possible, chaîne de caractères (mais l'évaluation numérique est possible lorsqu'elle a du sens).

La déclaration et l'affectation se font simultanément par :

`var=`*valeur* **sans espace avant ni après le signe '='**.

On accède à la valeur d'une variable `var` par `$var` ou `${var}`.

Exercice 1 – *variables du shell*

1. Vérifier que la variable `SHELL` a déjà une valeur lorsque vous lancez votre terminal. Laquelle ?
2. Déclarer une variable `NOM` contenant votre prénom **et** votre nom. Comment résoudre le problème de l'espace entre le prénom et le nom ? Afficher la valeur de `NOM` pour vérifier que l'affectation est correcte.

   La commande « `set` » utilisée sans argument affiche la liste de toutes les variables dont la valeur est instanciée dans le shell courant.

3. Utiliser un tube et la commande « `less` » pour faire défiler le résultat de la commande « `set` » page par page. Vérifier que la variable `NOM` précédemment définie apparaît bien dans cette liste.
4. Écrire une ligne de commande qui vous accueille en affichant gaiement "`Bonjour, Seraphin Lampion !`" (en supposant qu'il s'agisse du contenu de la variable `NOM`).
5. Repérer avec « `set` » les variables donnant le chemin absolu de votre répertoire privé, celui de votre répertoire courant, ainsi que votre nom d'utilisateur.
6. Modifier le script `enterre_tresor.sh` du TP n° 7 pour qu'il affiche le texte suivant :
   `Bonjour` *nom_d'utilisateur*,
   `je viens de cacher un trésor dans ton répertoire` *nom_du_répertoire_courant*.

Les paramètres Il est possible de passer des paramètres aux scripts. Pour les manipuler, il existe plusieurs variables spéciales, en particulier :

- `$#` donne le nombre de paramètres passés au script,
- `$0` donne le nom du programme en cours d'exécution,
- pour chaque entier `i` entre 1 et `$#`, `$i` donne le i^e paramètre,
- `$@` contient la liste de tous les paramètres, séparés par des espaces. **Attention, il faut toujours protéger l'évaluation de cette variable avec des guillemets :** `"$@"`.

Exercice 2 – *paramètres d'un script*

1. Écrire un script `analyse.sh` qui affiche :

   `Bonjour, vous avez rentré` *nombre_de_paramètres* `paramètres.`
   `Le nom du script est` *nom_du_script*.
   `Le 3ème paramètre est` *3ème_paramètre*.
   `Voici la liste des paramètres :` *liste_des_paramètres*.
   `Au revoir !`
2. Écrire un script `compare.sh` prenant deux paramètres et affichant :
   - `"usage : compare.sh fic1 fic2"` si le nombre de paramètres n'est pas 2 ;
   - `"fichier fic introuvable"` si l'un des paramètres (par exemple `fic`) n'est pas une référence de fichier valide ;
   - `"fichiers identiques"` ou `"fichiers différents"` selon les cas, si les deux références sont valides.

Exercice 3 – *portée des variables par défaut*

1. Vérifier que la variable `NOM` contient toujours vos nom et prénom.
2. Depuis le terminal courant, ouvrir un nouveau shell à l'aide de la commande « `bash` », puis afficher à nouveau la valeur de la variable `NOM`. Taper `Ctrl+D` ou `exit` pour revenir au shell précédent. Faire ensuite de même dans un autre terminal ouvert depuis le bureau. Qu'en déduisez-vous sur la portée des variables du shell ?
3. Écrire un script `portee.sh` qui affiche la valeur de la variable `NOM`, puis lui donne la valeur `Tartempion` et l'affiche à nouveau. Quel affichage obtient-on ? Et si on affiche à nouveau la valeur de `NOM` depuis le shell initial ?

Variables d'environnement Dans certains cas, on souhaite que la valeur d'une variable soit accessible également aux processus descendant du shell courant ; on parle alors de variables *d'environnement*, dont la liste est accessible grâce à la commande « `env` ».
La commande « `export` » permet de transformer des variables en variables d'environnement.

Exercice 4 – *variables d'environnement*

1. Afficher la liste des variables d'environnement. Que remarquez-vous par rapport à la sortie de la commande `set` ?
2. Transformer `NOM` en variable d'environnement ; est-elle visible par un shell enfant ? Inversement, exporter une variable dans un shell enfant ; est-elle visible par le shell parent. Que peut-on en déduire sur la portée des variables d'environnement ?
3. La commande « `cd` », que vous connaissez bien, permet de changer de répertoire courant. Utilisée sans argument, elle fixe le répertoire courant au répertoire personnel de l'utilisateur. Ce comportement est en fait déterminé par la valeur d'une certaine variable d'environnement.

   Repérez cette variable dans la liste des variables d'environnement, et faites en sorte que la commande `cd` renvoie par défaut vers le répertoire racine.

Trouver le bon chemin La variable d'environnement `PATH` joue un rôle primordial : c'est elle qui contient la liste de tous les chemins vers les répertoires dans lesquels les fichiers exécutables doivent être recherchés. C'est grâce à elle qu'il est possible d'utiliser des noms de commandes simples tels que `bash`, `grep` ou `ls` en lieu et place du chemin complet vers les exécutables concernés.

Le caractère `':'` sert de séparateur entre les différents chemins contenus dans la valeur de `PATH`.

Exercice 5 – *la variable d'environnement* `PATH`

1. Quelle est la valeur de la variable d'environnement `PATH` ?
2. Créer dans votre répertoire maison un répertoire que vous nommerez `bin_perso`, et créer dans ce répertoire un fichier nommé `fic`, contenant la ligne suivante :

   ```
   echo "ceci est le résultat de l'exécution du fichier fic."
   ```

3. Vérifier que vous avez les droits d'exécution sur le fichier `fic`, puis essayer de l'exécuter de différentes manières : depuis le répertoire `bin_perso` en tapant `./fic`, depuis un répertoire quelconque par `~/bin_perso/fic` et décrire le résultat.
4. Essayer ensuite de l'exécuter en tapant simplement `fic`, depuis le répertoire `bin_perso`, puis depuis un autre répertoire. Cela permet-il d'exécuter le fichier `fic` ?
5. Ajouter maintenant à votre variable `PATH` le chemin `~/bin_perso`. Pour cela, il faudra affecter à la variable `PATH` son ancienne valeur concaténée avec la chaîne de caractères `:~/bin_perso`.
6. Essayer d'exécuter `fic` en tapant simplement la commande `fic` depuis n'importe quel répertoire de travail. Que se passe-t-il ?
7. Comme vous avez dû le remarquer dans l'exercice précédent, les valeurs des variables d'environnement sont accessibles à tous les processus fils du shell courant. Comment faire pour que le chemin `~/bin_perso/` soit inclus dans la valeur de la variable `PATH`, dès qu'on lance un nouveau terminal ?

Exercice 6 – *la variable d'environnement* `CDPATH` *(optionnel)*

1. Créer dans votre répertoire personnel une arborescence comprenant les répertoires `~/DossiersImportants/Je/Vais/Tous/Les/Jours/Ici/` et `~/DossiersImportants/Je/Le/Consulte/Sans/Arret/La/`.
2. Affecter la variable d'environnement `CDPATH` de manière à pouvoir aller dans ces répertoires avec les commandes `cd Ici` et `cd La`.
3. Cette méthode nécessite de modifier la variable d'environnement `CDPATH` pour chaque

Exercice 7

Écrire un script qui affiche dans le terminal les lignes suivantes ; utiliser le mécanisme d'expansion afin d'éviter de taper la dernière ligne en entier !

`Le chemin absolu de mon répertoire de login est` *valeur donnée par le terminal*
`Le résultat de la commande whoami est` *résultat*
`Le produit de 33 par 17 est` *résultat*
`Il a agi anticonstitutionnellement anticonventionnellement antisocialement`

D'autres expressions sont expansées selon le contenu du répertoire courant :

Caractères	Interprétation
`*`	chaîne de caractères quelconque (sans `/`, ni `.` en début de mot)
`?`	un caractère exactement (autre que `/`, ou `.` en début de mot)
`[...]`	ensemble de possibilités pour un caractère
`[!...]`	ensemble de possibilités pour un caractère, défini par complémentation
`-`	intervalle dans un ensemble

Par exemple, dans un répertoire contenant des fichiers nommés `ab`, `abx`, `abd`, `abdx`, `adx`, `ax` et `bdx`, on aura :

Exemple	Résultat
`ab*`	`ab` `abx` `abd` `abdx`
`ab?`	`abx` `abd`
`a[abcd]x`	`abx` `adx`

Exercice 8

Dans un répertoire Maritime, créer les fichiers suivants :

```
butin gratin main malin marin marine matin mutin patin patio
```

Écrire un script affichant les lignes suivantes, en utilisant le mécanisme d'expansion :

```
marin marine
matin patin
gratin matin patin
matin patin patio
malin marin
butin matin mutin
main malin marin matin
```

Échappement

Nous venons de voir que certains caractères indiquent au shell qu'il doit opérer une expansion. D'autres caractères tels que ~, &, <, >, | et ; jouent également un rôle spécial lors de l'interprétation des commandes, comme nous l'avons déjà vu au cours des TP prédédents. Pour pouvoir afficher tels quels ces caractères spéciaux, il existe un mécanisme dit *d'échappement*.

Syntaxe	**Expression**	**Résultat**	**Explication**
\	`\"\'\$\{\ \A B`	`"'${ A` `B`	échappement d'un caractère
"..."	`" a $HOME $((1+1)) ' "`	`a /home/roger 2 '`	échappe tout sauf `$`,`\`, `!` et `"`
'...'	`' a $HOME $((1+1)) " '`	`a $HOME $((1+1)) "`	échappe tout sauf `\`

Attention, dans le tableau ci-dessus, ' est une apostrophe et pas un accent grave.

Exercice 9

Écrire un script qui affiche dans le terminal les lignes suivantes :

```
A B
L'ensemble {1, 2, 3} est inclus dans N
Elle demanda d'une voix *forte* : "Qui est-ce ?"
$HOME = le chemin absolu vers votre répertoire de login donné par le shell
${HOME} = le chemin absolu vers votre répertoire de login donné par le shell
19 * 216 = la valeur du produit calculée par le shell
Ajoutez un peu d'huile d'olive, d'humeur joyeuse, d'humus, et remuez bien !
```

Exercice 10

Écrire un script définissant une variable `TOTO` égale à `whoami`, puis affichant le texte ci-dessous. Attention, on demande que chaque ligne à afficher utilise la variable `TOTO` !

```
J'aime bien utiliser la commande "whoami".
Ma variable $TOTO contient 'whoami'.
Mon login est votre login donné par le shell.
whoamitruc, c'est un bidule imaginaire !
La commande d'aujourd'hui est \whoami\.
```