Introduction aux systèmes d'exploitation (IS1)

TP 6 – Variables et expansions du shell

Variables

En plus des usages que vous avez vus jusqu'ici, le shell propose d'autres fonctionnalités (c'est en fait un véritable langage de programmation, comme nous le verrons par la suite). En particulier, comme en Java, il est possible d'utiliser des *variables*.

Une variable est repérée par un nom appelé *identificateur*, qui peut être n'importe quelle suite de caractères commençant par une lettre ou le caractère « souligné » ('_') et ne contenant que des lettres, des chiffres ou le caractère souligné[1].

On peut assigner une valeur à la variable `var` avec l'instruction `var=`*valeur* (sans espace avant ni après le signe '='). Cette valeur peut être une chaîne de caractères ou un entier par exemple.

On accède à la valeur associée à une variable en faisant précéder son identificateur du symbole `$`, comme par exemple dans la commande `echo $var` qui affiche la valeur de la variable `var`.

Exercice 1 – *Variables du shell*

1. Vérifiez que la variable `SHELL` a déjà une valeur lorsque vous lancez votre terminal. Quelle est cette valeur ?
2. Déclarez une variable `NOM` contenant votre prénom **et** votre nom. Comment résoudre le problème de l'espace entre le prénom et le nom ? Affichez la valeur de `NOM` pour vérifier que l'affectation est correcte.
3. La commande `set` utilisée sans argument affiche la liste de toutes les variables dont la valeur est instanciée dans le shell courant. Utilisez la commande `less` pour faire défiler le résultat de la commande `set` page par page, à l'aide d'un tube. Vérifiez que la variable précédemment définie apparaît bien dans cette liste.
4. Écrivez une ligne de commande qui affiche « `Bonjour, Machin Chose !` » (remplacez Machin par votre prénom et Chose par votre nom) en utilisant la variable `NOM`.
5. Depuis le terminal courant, ouvrez un nouveau shell en tapant la commande `bash`, puis affichez la valeur de la variable `NOM` (quand vous avez terminé, tapez `exit` pour revenir au shell précédent). Faites de même dans un nouveau terminal lancé depuis la barre d'icônes du bureau. Qu'en déduisez-vous sur la portée des variables du shell ?

Dans certains cas, on souhaite que la valeur d'une variable soit accessible également aux processus fils du shell courant. De telles variables sont appelées variables *d'environnement*.

Exercice 2 – *Variables d'environnement*

1. Affichez la liste des variables d'environnement grâce à la commande `env`. Que remarquez-vous par rapport à la sortie de la commande `set` ?

[1] À l'exception de certaines variables spéciales du shell.

2. On transforme une variable `var` en variable d'environnement grâce à la commande `export var`. Transformez `NOM` en variable d'environnement et répétez la question 5. Qu'en déduisez-vous 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.

Exercice 3 – *La variable d'environnement* `PATH`

1. Quelle est la valeur de la variable d'environnement `PATH` ?

   Le caractère ' `:` ' sert de séparateur entre les différents chemins contenus dans la valeur de `PATH`.
2. Créez dans votre répertoire maison un répertoire que vous nommerez `bin_perso`, et créez dans ce répertoire un fichier, nommé `fic`, contenant la ligne suivante :

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

   Détaillez les commandes utilisées (essayez de faire le plus court possible).
3. Vérifiez que vous avez les droits d'exécution sur le fichier `fic`. Modifiez-les si nécessaire. Félicitations, vous venez de créer votre premier exécutable shell. Exécutez-le depuis le répertoire `bin_perso` en tapant `./fic`, et décrivez le résultat. Essayez maintenant de taper `./fic` depuis un autre répertoire. Cela vous permet-il d'exécuter le fichier `fic` ? Que faut-il faire pour pouvoir exécuter `fic` depuis un autre répertoire que `bin_perso` ?
4. Ajoutez maintenant à votre variable `PATH` le chemin `/home/login/bin_perso` (où `login` est votre login). Vous devrez pour celà affecter à la variable `PATH` son ancienne valeur à laquelle vous ajouterez la chaîne de caractères

   ```
   :/home/login/bin_perso
   ```
5. Placez-vous maintenant dans votre répertoire maison, et tapez `fic`. Que se passe-t-il ? Qu'en déduisez-vous sur l'utilité de la variable `PATH` ?
6. 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 `/home/login/bin_perso/` soit inclus dans la valeur de la variable `PATH`, dès qu'on lance un nouveau terminal ?

Expansions

Quand un processus shell reçoit une ligne de commande, il pré-traite cette chaîne de caractères avant de l 'exécuter. Tout d'abord, il découpe cette chaîne de caractères, selon le séparateur défini dans la variable `IFS` (par défaut, c'est le caractère« espace »). Ainsi la ligne

```
cat fic1 fic2 fic3
```

est découpée en quatre chaînes de caractères

`cat` `fic1` `fic2` `fic3`

Ce qui permet au processus shell de différencier la commande des arguments dans la chaîne de caractères qu'on lui fournit. Mais ce n'est pas tout, il existe d'autres caractères spéciaux qui permettent de demander au shell de faire d'autres pré-traitements sur la chaîne de caractères, avant de l'exécuter (on dit que le shell procède à l'*expansion* de cette chaîne de caractères). Nous avons déjà vu, dans la partie précédente que lorsqu'on fournit la chaîne de caractères `$id` au shell, le shell va la remplacer par la valeur de la variable dont l'identifiant est `id`. Voici un tableau récapitulant les différents types d'expansions en `bash`.

Syntaxe	Expression	Expansion	Explication
`{ }`	`ba{ba,bu}`	`baba` `babu`	énumération
`\`	`\"\'\$\{\ \A B`	`"'${ A` `B`	échappement
`$`	`$HOME`	`/home/roger`	valeur de variable
`${...}`	`${HOME}`	`/home/roger`	valeur de variable
`$(...)`	`$(which true)`	`/bin/true`	sortie d'une commande
`‘...‘`	`‘which true‘`	`/bin/true`	sortie d'une commande
`$((...))`	`$((12 + 4 * 2))`	`20`	valeur d'une expression
`"..."`	`" a $HOME $((1+1)) ' "`	`a /home/roger 2 '`	$ expansés
`'...'`	`' a $HOME $((1+1)) " '`	`a $HOME $((1+1)) "`	$ non-expansés

D'autres expressions sont expansées selon le contenu du répertoire courant. Par exemple, dans un répertoire contenant des fichiers nommés `abx`, `abd`, `abcde`, `adx` et `bdx`, on aura :

Caractères	Expression	Expansion	Explication
`*`	`ab*`	`abx` `abd` `abcde`	joker chaîne de caractères
`?`	`ab?`	`abx` `abd`	joker un caractère
`[...]`	`a[abcd]x`	`abx` `adx`	liste de possibilités

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 précédents.

Exercice 4 – *Parlez-vous* `bash` *? Leçon 1*

Utilisez la commande `echo` pour écrire **exactement** les lignes suivantes dans un fichier que vous nommerez `leçon1`.

1. `A B`
2. `L'ensemble {1, 2, 3} est inclus dans N`
3. `Elle s'écria : "Ciel mon mari !"`
4. `$HOME = `*`chemin`* où *chemin* est le chemin de votre répertoire de login **d'après le bash**
5. `19 * 216 = `*`produit`* où *produit* est la valeur du produit **calculée par le bash**

Exercice 5 – *Parlez-vous* `bash` *? Leçon 2*

Supposons que vous vous trouviez dans un répertoire contenant les fichiers suivants :

`marin marine marino Marin Marine Martin Martine Martinique Maurine`

Quelle expression (**une seule chaîne de caractères**) faut-il utiliser pour désigner, après expansion **exactement** :

1. `marin` et `Marin` ?
2. `marin` et `marine` et `marino` ?
3. `marin` et `Marin` et `martin` et `Martin` ?
4. `Martine` et `Martinique` ?
5. `Marine` et `Martine` et `Maurine` ?

Exercice 6 – *Outils pour les chaînes de caractères*

Instanciez une variable que vous nommerez `mot`, à la valeur `unmottrestreslong` Affichez les expansions des expressions suivantes. Deéduisez le sens de chacune d'entre elles.

1. `${#mot}`
2. `${mot:4}`
3. `${mot:4:3}`
4. `${mot#unmot}`
5. `${mot%long}`

Exercice 7 – *Mon premier script*

Créez un fichier exécutable, que vous nommerez `env_perso`, et que vous rendrez exécutable depuis n'importe quel répertoire de votre arborescence. Le résultat de cette nouvelle commande sera de vous afficher dans le terminal et de la façon la plus lisible possible :

1. votre login
2. la valeur de votre variable `HOME`
3. la date du jour
4. le chemin d'accès à votre exécutable `xemacs`
5. la somme des numéros de jour, mois et année du jour présent
6. une ligne vous indiquant si votre exécutable `xemacs` et votre exécutable `bash` sont dans le même répertoire
7. le type de shell que vous utilisez par défaut (sans son chemin d'accès complet)