INF203
Memo bash

1 Variables

Affectation (et définition au besoin) : `nom=valeur`
Accès : `$nom`

1.1 Variables d'environnement

Valeur conservée lors de l'invocation d'un sous shell. Transformation en variable d'environnement :
`export nom`
`export nom=valeur`

1.2 Variables automatiques

`$?`	code de retour de la dernière commande (0 si ok)
`$0`	nom du script
`$1` à `$9`	les éventuels 9 premiers arguments passés au script
`$#`	nombre d'arguments
`$*`	liste des arguments (à partir de `$1`)

2 Expansion des métacaractères

2.1 En correspondance avec le système de fichiers

* remplacée par un nombre quelconque de caractères quelconques ;
? remplacé par exactement un caractère quelconque ;
[ensemble] remplacé par exactement un caractère faisant partie de l'ensemble donné. L'ensemble est une séquence de caractères et/ou d'intervalles de caractères. Un intervalle de caractères est un caractère, suivi d'un tiret suivi d'un caractère supérieur au premier dans l'ordre des codes ASCII.

2.2 De substitution

$ remplacé avec le `nom` qui suit par la valeur de la variable correspondante ;
$(commande) remplacé par la sortie standard obtenue lors de l'exécution de la `commande`.

2.3 De protection

\ protège le caractère qui suit de l'expansion ;
'chaine' protège la `chaine` de l'expansion ;
"chaine" protège le `chaine` de l'expansion sauf pour les caractères `$`, `$()` et `.`

3 Redirection des entrées/sorties

`< fichier` : lecture de l'entrée standard depuis `fichier` ;
`> fichier` : écriture de la sortie standard vers `fichier` ;
`2> fichier` : écriture de la sortie d'erreur standard vers `fichier` ;
`>> fichier` : concaténation de la sortie standard à la fin de `fichier` ;
`2>> fichier` : concaténation de la sortie d'erreur standard à la fin de `fichier` ;
`commande_1 | commande_2` :
redirection de la sortie standard de `commande_1` vers l'entrée standard de `commande_2`.

4 Les tests

Syntaxe : `[ expression ]` (ou `test expression`).

4.1 Sur les fichiers et répertoires

option	signification
`-e fich`	`fich` existe
`-s fich`	`fich` n'est pas vide
`-f fich`	`fich` est un fichier
`-d fich`	`fich` est un répertoire
`-r fich`	`fich` a le droit `r` (lecture)
`-w fich`	`fich` a le droit `w` (écriture)
`-x fich`	`fich` a le droit `x` (exécution)

4.2 Sur les entiers

option	signification
n_1 -eq n_2	$n_1 = n_2$
n_1 -ne n_2	$n_1 \neq n_2$
n_1 -lt n_2	$n_1 < n_2$
n_1 -gt n_2	$n_1 > n_2$
n_1 -le n_2	$n_1 \leq n_2$
n_1 -ge n_2	$n_1 \geq n_2$

4.3 Sur les chaines

option	signification
`-z chaine`	`chaine` est vide
`-n chaine`	`chaine` n'est pas vide
`chaine`$_1$ `= chaine`$_2$	les 2 chaines sont identiques
`chaine`$_1$ `!= chaine`$_2$	les 2 chaines sont différentes

4.4 Entre expressions

option	signification
`expr`$_1$ `-a expr`$_2$	et
`expr`$_1$ `-o expr`$_2$	ou
`! expr`	non

5 Les structures de contrôle

5.1 Structure conditionnelle

```
if <condition>
then
    # si condition est vraie
    <suite de commandes 1>
else
    # sinon
    <suite de commandes 2>
fi
```

La condition est déterminée par le code de retour d'une commande. La partie `else` est facultative.

5.2 Conditionnelles imbriquées

```
if <condition_1>
then
    # si condition_1 est vraie
    <suite de commandes 1>
elif <condition_2>
then
    # si condition_1 est fausse
    # et condition_2 vraie
    <suite de commandes 2>
else
    # si les deux conditions sont fausses
    <suite de commandes 3>
fi
```

5.3 Boucle `for`

```
for <variable> in <liste>
do
    <suite de commandes>
done
```

5.4 Boucle `while`

```
while <condition>
do
    <suite de commandes>
done
```

5.5 Fonctions

```
<nom_fonction>() {
   <suite de commandes>
}
```

6 Commandes utiles (petit sous ensemble...)

`echo [arguments]` : affiche ses arguments sur une ligne ;
`read nom` : lit une ligne et la stocke dans la variable `nom` ;
`expr expression` : affiche la valeur d'une expression arithmétique ;
`basename nom [extension]` : affiche le `nom` privé de sa partie chemin (jusqu'au dernier slash) et de son éventuelle extension ;
`dirname nom` : affiche la partie chemin (avant le dernier slash) de `nom` ;
`grep motif [fichiers ...]` : affiche les lignes de ses entrées contenant le `motif`.
`diff, find, tr, head, tail, cut, sed, sort, ...` : et bien d'autres...