Bash : les symboles dans les calculs

- Objet : suite de la série de wiki visant à maîtriser bash via les différents caractère spéciaux.
- Niveau requis :
  débutant, avisé
- Commentaires : scripts
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !.
- Suivi :
  - Création par Hypathie le 08/04/2014
  - Testé par Hypathie Avril 2014
- Commentaires sur le forum : ici [1)]

- Vision d'ensemble
- Détail et caractères
- Les opérateurs lexicographiques
- Les opérateurs de comparaison numérique
-
- Les tableaux
- Les caractères de transformation de parametres
- Bash : Variables, globs étendus, ERb, ERe

Opérateurs arithmétiques

opérateurs	opérations
+	addition
-	soustraction
/	division (récup. le quotient)
%	modulo (récup. le reste)
*	multiplication
**	puissance (`bash 2.02 et sup.`)

> Pour les calculs complexes bash n'est pas adapté, il faut utiliser le langage `bc`.

Opérateurs d'affectation arithmétique

opérateurs	opérations
=	affectation arithmétique
+=	incrémentation
-=	décrémentation
/=	affectation par division
+=	affectation par multiplication
%=	afectation du reste

Voir :

- **typologie de variables**
- **variables numériques et calculs**

Opérateurs binaires

Ces opérateurs s'utilisent sur des binaires, (sur des 1 et des zéro).

Opérateurs	significations
«	décalage d'un bit à gauche (=mutiplication par deux)
»	décalage d'un bit à droite (=division par deux)
&<	“et” logique (ex : on a une variable=1; si on fait &1 cela fait 1 (en binaire 1 et 1 = 1)
|<	ou (inclusif) binaire
~	non binaire
^	XOR (ou exclusif) binaire

Le &< (“et” binaire), le |< (“ou” binaire), et le ~ (“non” binaire)
peuvent aussi être remplacés (de façon équivalente) par les opérateurs logiques que l'on a vus au sujet de la composition de commandes sur erreur ou sur réussite.

> && : exécution de la commande suivante si, et seulement si la précédente renvoie 0
>
> || : exécution de la commande suivante si, et seulement si la précédente renvoie autre chose que 0
>
> ! : inverse du retour d'une commande, c'est à dire un “non” logique

Écriture utile pour les boucles

Post-incrémentation et pré-incrémentation :

script

```
#!/bin/bash
declare -i x=20 y  # ici les signes = permettent une affection
(( y = x++ ))      # d'abord la valeur de x est conservée dans la
valeur de y (donc $y= 20) puis la valeur
                   #  de x est incrémentée ($x est donc égal à 21)
                   # les espaces autour du signe = ne sont pas
obligatoires
echo "y=$y x=$x"   # réponse : y=20 x=21
```

script

```
#!/bin/bash
declare -i x=20 y
```

```
(( y = ++x ))      # d'abord la valeur de x est incrémentée puis la
valeur de y reçoit
                   # la valeur du x incrémenté
                   # les espaces autour du signe = ne sont pas
obligatoires
echo "y=$y x=$x"   # réponse : y=21 x=21
```

Post-décrémentation et pré-décrémentation :

script

```
#!/bin/bash
declare -i x=20 y
(( y = x-- ))      # d'abord la valeur de x est conservée dans la
valeur de y (donc $y= 20)
                   # puis la valeur de x est décrémentée ($x est donc
égal à 19)
                   # les espaces autour du signe = ne sont pas
obligatoires
echo "y=$y x=$x"   # réponse : y=20 x=19
```

script

```
#!/bin/bash
declare -i x=20 y
(( y = --x ))          # d'abord la valeur de x est décrémentée
($x=19), puis la valeur de y
                       # reçoit la valeur du x incrémenté (donc $y=19)
                       # les espaces autour du signe = ne sont pas
obligatoires
echo "y=$y x=$x"       # réponse : y=19 x=19
```

En bref,
Post-incrémentation/décrémentation : Les signes d'incrémentation (**++**) ou de décrémentation (**--**) sont placés **après** une valeur à incrémentée (+1) ou à décrémenter (-1) ; cette valeur est conservée dans “y” puis elle est **incrémentée (+1)** ou **décrémentée (-1)**.

Pré-incrémentation/décrémentation : Les signes d'incrémentation (**++**) ou de décrémentation (**--**) sont placés **avant** une valeur à incrémentée ou à décrémenter ; cette valeur est **incrémentée (+1)** ou **décrémentée (-1)** puis elle est conservée dans “y”.

Tuto précédent

Bash : les opérateurs de comparaison numérique

La suite, c'est ici

Bash : les tableaux

[1] N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !