





# Script bash : modification de variable et de paramètre

- Objet : Script bash : modification de variable et de paramètre
- Niveau requis :  
[débutant](#), [avisé](#)
- Commentaires : 
- Débutant, à savoir : [Utiliser GNU/Linux en ligne de commande, tout commence là !](#) 😊
- Suivi :
  - Création par  Hypathie le 18/03/2014
  - Testé par  Hypathie Juin 2014
- Commentaires sur le forum : [Lien vers le forum concernant ce tuto](#) <sup>1)</sup>

Contributeurs, les  sont là pour vous aider, supprimez-les une fois le problème corrigé ou le champ rempli !

## Nota : Les autres wiki :

- [debuter-avec-les-scripts-shell-bash](#)
- [script-bash-variables-arguments-parametres](#)
- 😊
- [script-bash-enchainement-de-commandes-et-etat-de-sortie](#)
- [script-bash-etat-de-sortie-et-les-tests](#)
- [script-bash-les-tableaux](#)
- [script-bash-les-fonctions](#)

## Modifications de variables

Pré-requis : utiliser les structures de contrôle

- Voir :
  - [boucle while](#)
  - [boucle for](#)

## Modifier la variable d'une structure de contrôle par les paramètres de position

[script](#)

```
#!/bin/bash
for i in "$@"
do
    echo "Vous avez donné à la variable 'i' la valeur : $i."

    if [ "$1" != "coucou" ] ; then
        echo "Le premier argument doit être 'coucou'."
```

```
else
    echo "OK"
    if [ "$2" != "toi" ] ; then
        echo "Le deuxième paramètre doit être 'toi'."
    else
        echo "MERCI"
    fi
fi
done
echo " " #pour sauter une ligne

echo $@
```

Ci-dessus, la variable `i` de la boucle prend tour à tour la valeur des paramètres passé au script depuis le terminal et la structure `if` teste si la chaîne du paramètre passé au script (valeur de la variable) correspond au motif voulu.

## Modifier une variable par les paramètres de position déclarés dans le script

### script

```
#!/bin/bash
set a b c
    echo "Avec 'shift', on se décale d'un paramètre à chaque boucle."

for i in "$@"
do
    i=$1
    shift 1
    echo "les paramètres sont : $1 :$2 :$3 ."
done
```

Avec 'shift', on se décale d'un paramètre à chaque boucle.  
les paramètres sont : b :c : .  
les paramètres sont : c : : .  
les paramètres sont : : : .



Contrairement aux boucles `while` et `for`, la variable du sélecteur `case` n'est pas modifiée par les paramètres, mais permet de tester chacune des valeurs déclarées dans ses cases, soit avec un paramètre de position, soit avec une variable du script.  
Voir : [Tests sur paramètres passés au scripts](#)

# Manipuler la valeur des paramètres de position : set

## Syntaxe de de la commande set

```
set argument1 [argument2] ...
```

À savoir :

1. variables de substitution prédéfinies [bash-les-differents-caracteres-speciaux](#)
2. [quand les valeurs sont des paramètres](#)

La *commande set* permet d'affecter une valeur provisoire à un ou plusieurs paramètres de position. Les arguments de la commande set seront les valeurs des paramètres.

## Exemple

Soit le script "mon-script"

[mon-script](#)

```
#!/bin/bash
set un deux trois
echo "$1" "$2" "$3"
echo "$#"
echo "il y a deux paramètres de position : "$1" ; "$2"."
echo " et les enregistre dans une variable prédéfinie pour le shell
courant."
```

```
./mon-script
```

[retour de la commande](#)

```
un deux trois
3
il y a deux paramètres de position : un ; deux.
 et les enregistre dans une variable prédéfinie pour le shell courant.
```

## Suppression des paramètres de position

Soit le script "mon-script"

[mon-script](#)

```
#!/bin/bash
```

```
set a b c
echo $@
set --
echo "Plus de paramètre $@ !"
```

./mon-script

[retour de la commande](#)

```
a b c
Plus de paramètre !
```

## Attention à la commande set !

Comparez les trois exemples ci-dessous :

### Exemple 1

Soit le script "mon-script1".

[mon-script1](#)

```
#!/bin/bash
var=lettres
set a b c      #affectation des paramètres de position a b c
echo "$var" "$1" "$2" "$3"
echo "$#"
if [ $# == 3 ] ; then
echo "il y a trois paramètres, leurs valeurs sont : "$1", "$2", "$3" "
echo " et une variable nommée var de valeur \"lettres\"."
fi
```

./mon-script1

[retour de la commande](#)

```
lettres a b c
3
il y a trois paramètres, leurs valeurs sont : a, b, c
et une variable nommée var de valeur "lettres".
```

## Exemple 2

Soit le script "mon-script2" à comparer avec "mon-script1" de l'exemple 1.

mon-script2

```
#!/bin/bash
var=lettres
echo "$var" "$1" "$2" "$3"
echo "$#"
if [ $# == 3 ] ; then
echo "il y a trois paramètres, leurs valeurs sont : "$1", "$2", "$3" "
echo " et une variable nommée var de valeur \"lettres\"."
fi
```

```
./mon-script2 a b c
```

retour de la commande

```
lettres a b c
3
il y a trois paramètres, leurs valeurs sont : a, b, c
et une variable nommée var de valeur "lettres".
```

## Exemple 3

Soit le script "mon-script3" à comparer avec "mon-script2" et "mon-script3".

mon-script3

```
#!/bin/bash
var=lettres
set 1 2 # affectation des paramètres 1 2
echo "$var" "$1" "$2" "$3"
echo "$#"
if [ $# == 2 ] ; then
echo "il y a deux paramètres, leurs valeurs sont : "$1", "$2", "$3",
"$4"."
echo " et une variable nommée var de valeur \"lettres\"."
fi
```

```
./mon-script3 a b c d
```

retour de la commande

```
lettres 1 2
2
il y a deux paramètres, leurs valeurs sont : 1, 2, , .
et une variable nommée var de valeur "lettres".
```

La commande `set` supprime toute possibilité de se référer à des paramètres passés au script (depuis le terminal), et ceci y compris si on supprime avec `set -` les paramètres de position déclarés avec `set`.

- Sans argument la commande `set` affiche tous les noms et toutes les valeurs des variables pré-définies. Mais elle n'enregistre pas une valeur d'une variable non-déclarée.
- Soit "mon-script" :

mon-script

```
#!/bin/bash
var=lettres
set coucou
set
```

```
./mon-script | grep var
```

retour de la commande



```
BASHOPTS=cmdhist:extquote:force_ignore:hostcomplete:interactive_comments:progcomp:promptvars:sourcepath
XAUTHORITY=/var/run/gdm3/auth-for-hypathie-HhozZG/database
var=lettres
```

Nous retrouvons la variable `var` de notre script avec sa valeur "lettres".

```
./mon-script | grep coucou
```

retour de la commande

```
_ =coucou
```

On retrouve un argument unique

```
./mon-script | grep a b c
```

### retour de la commande



```
grep: b: Aucun fichier ou dossier de ce type
grep: c: Aucun fichier ou dossier de ce type
```

Mais on ne retrouve pas les paramètres a b c.

## Usage avancée de la commande set

### Activer les options du shell dans un script

```
set -o nom-de-l'option
```

ou

```
set -abréviation-de-l'option
```

### Exemple

Il peut être utile de vérifier grâce à un message d'erreur, si on appelle la valeur d'un paramètre qui n'a pas été défini.

### script

```
#!/bin/bash
set -o nounset
var=a
var1=

echo $var
echo $var1
echo $var3
```

a

```
/home/hypathie/MesScripts/scriess: ligne8: var3 : variable sans liaison
```

Un tableau des options ici: <http://abs.traduc.org/abs-5.3-fr/ch30.html#optionsref>

### Ordonner en colonne la sortie de commandes avec set et une boucles

Par exemple : soit le script ci-dessous boucle-set

## script boucle-set.sh

```
#!/bin/bash
var=$1
echo $1
set a b c
#set --
echo $@

for i in "$@"
do
i=$1

echo "les paramètres sont : $1 :$2 :$3 ."
shift 1
echo $var
done
```

```
./boucle-set un deux trois
```

### retour de la commande

```
a b c
les paramètres sont : a :b :c .

les paramètres sont : b :c : .

les paramètres sont : c : : .

~$
```

```
./boucle-set.sh argument1
```

### retour de la commande

```
argument1
a b c
les paramètres sont : a :b :c .
argument1
les paramètres sont : b :c : .
argument1
les paramètres sont : c : : .
argument1
```



# La suite c'est ici

[script-bash-enchainement-de-commandes-et-etat-de-sortie](#)

1)

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

From:

<http://debian-facile.org/> - Documentation - Wiki

Permanent link:

<http://debian-facile.org/doc:programmation:shells:script-bash-detail-sur-les-parametres-et-les-boucles>

Last update: 21/10/2015 19:46

