



I106B

19. Boucles, listes et tableaux

[HTTPS://WEB.ARCHIVE.ORG/WEB/20161119052258/HTTP://RYANSTUTORIALS.NET:80/BASH-SCRIPTING-TUTORIAL/BASH-LOOPS.PHP](https://web.archive.org/web/20161119052258/http://ryanstutorials.net:80/bash-scripting-tutorial/bash-loops.php)

Tableau

2

- ▶ **Toutes les variables sont des tableaux !**
- ▶ Les indices commencent à 0
- ▶ Accès à un indice particulier :
 - `t[i]="valeur"` → assignation à l'indice `i` de `t`
 - `echo ${t[i]}` → accès à la valeur de `t[i]`
 - `$` optionnel pour accéder à la valeur de l'indice `i`
$$\${t[\$i]} \leftrightarrow \${t[\mathbf{i}]}$$
- ▶ Toute variable est un tableau \Rightarrow accès habituel à la variable = accès à l'indice 0 de ce tableau :
 - `t="toto" \Leftrightarrow t[0]="toto"`
 - `echo $t \Leftrightarrow echo ${t[0]}`

- ▶ Pas de contrôle de borne, tous les indices sont disponibles
- ▶ Trous possibles
- ▶ Nombre d'éléments dans le tableau t :

$\$ \{ \#_t [*] \}$

- Il s'agit de la taille logique du tableau
- Attention : s'il y a des trous dans le tableau, ils ne seront pas comptés → le nombre d'éléments ne donnera pas l'indice du dernier élément + 1

Liste

4

- ▶ Une variable peut aussi être considérée comme une liste
 - Les éléments sont séparés par des *whitespaces* (i.e. espace, passage à la ligne, tabulation)

```
liste1="e11 e12 e13 e14"
```

```
liste2=$(ls)
```

- ▶ **\$*** → *liste* des arguments du script
(rappel: **\$#** = nombre d'arguments)

Conversions liste \leftrightarrow tableau

5

En considérant que t est un tableau et l une liste:

- conversion liste \rightarrow tableau

$$t = (\$l)$$

- conversion tableau \rightarrow liste

$$l = \$\{t[*]\}$$

while

6

```
while cmd1  → Tant que cmd1 réussit (code retour=0):  
do  
    cmd2    → exécute cmd2  
done
```

Attention: les espaces et les retours à la ligne sont importants.

Example

7

```
#!/bin/bash
cnt=0
args=($*)
while [ $cnt -lt ${#args[*]} ]
do
    ((cnt++))
    echo argument $cnt: ${args[cnt-1]}
done
```

while (suite)

8

Les boucles infinies permettent de garder le shell à l'écoute. Dans ce cas, il faut utiliser la commande `exit` ou `break` pour sortir de la boucle.

```
while (true)    # ou while true ou while ((1))
do
    read -p "Enter a line (quit to stop): " rep
    if [ "$rep" = "quit" ]; then
        exit 0
    fi
    echo $rep
done
```


until

9

Comme `while`, mais la condition est inversée.

```
#!/bin/bash
cnt=0
args=($*)
until [ $cnt -ge ${#args[*]} ]
do
    echo ${args[$cnt]}
    ((cnt++))
done
```

for

10

for var in \$list	→	Assigne <code>var</code> successivement à tous les éléments de la liste <code>list</code> :
do		
cmd2	→	exécute <code>cmd2</code>
done		

Le `for` du bash est donc une boucle « for each ».

Attention: les espaces et les retours à la ligne sont **importants**.

Examples

11

```
for i in 1 2 3 4 5 ; do  
    echo $i  
done
```

```
liste="a b c d"  
for el in $liste ; do  
    echo $el  
done
```

```
for arg in $* ; do  
    echo argument: $arg  
done
```

for range

12

```
#!/bin/bash
for value in {1..5}    # valeurs: 1 2 3 4 5
do
    echo $value
done
```

```
for value in {10..0..2}    # valeurs: 10 8 6 4 2 0
do
    echo $value
done
```

Notez que les bornes d'un *range* ne peuvent pas être des variables (dans ce cas, mieux vaut utiliser une boucle *while* avec compteur).

break & continue

13

- ▶ **break** : quitte instantanément une boucle
- ▶ **continue** : passe à l'itération suivante

Exemples de scripts

17

- ▶ Quelques scripts utilisant des alternatives et boucles sont disponibles sur *courslinux*, dans le repertoire :
“/home/anthony.legrand/**ex_scripts**”