



I106B

18. Alternatives et conditions

[HTTPS://WEB.ARCHIVE.ORG/WEB/20161119025624/HTTP://RYANSTUTORIALS.NET:80/BASH-SCRIPTING-TUTORIAL/BASH-IF-STATEMENTS.PHP](https://web.archive.org/web/20161119025624/http://ryanstutorials.net:80/bash-scripting-tutorial/bash-if-statements.php)

if cmd1	→	Si cmd1 réussit (code retour=0):
then		
cmd2	→	exécute cmd2
elif cmd3	→	Sinon Si cmd3 réussit (code retour=0):
then		
cmd4	→	exécute cmd4
else	→	Sinon:
cmd5	→	exécute cmd5
fi		

Les parties `elif` et `else` sont facultatives.

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

Exemple

3

```
if mkdir temp
then
    cd temp
else
    echo "Erreur création temp"
    exit 1
fi
```

Exemple

- ▶ Autre syntaxe :
sur une seule ligne, plusieurs commandes terminées par des ;

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

```
if mkdir temp; then cd temp
```

```
else echo "Erreur"; exit 1; fi
```

test

5

- ▶ `test` teste une expression booléenne



- code retour = 0 si l'expression est vraie
- code retour != 0 si fausse

- ▶ Peut donc être utilisé en condition d'un `if`

```
test $# -gt 5
```

Teste si le nombre de paramètres est strictement supérieur à (*greater than*) 5

- ▶ Opérateurs booléens et de comparaison:

```
man test
```

ou

<https://wiki.bash-hackers.org/commands/classictest>

Principaux opérateurs de test

6

<u>OPÉRATEUR</u>	<u>DESCRIPTION DU TEST</u>
! EXPRESSION	L'EXPRESSION est fausse
-n STRING	STRING est non vide
-z STRING	STRING est vide
STRING1 = STRING2	STRING1 est égale à STRING2
STRING1 != STRING2	STRING1 est différente de STRING2
INTEGER1 -eq INTEGER2	INTEGER1 est numériquement égal à INTEGER2
INTEGER1 -gt INTEGER2	INTEGER1 est strict. plus grand que INTEGER2
INTEGER1 -lt INTEGER2	INTEGER1 est strict. plus petit que INTEGER2
-d FILE	FILE existe et est un répertoire
-e FILE	FILE existe
-s FILE	FILE existe et n'est pas vide
-r FILE	FILE existe et peut être lu
-w FILE	FILE existe et peut être modifié
-x FILE	FILE existe et peut être exécuté

Attention : les **espaces** entre opérateur et opérandes sont importants !

test

7

► **test** *expression*

```
if test $# -gt 5
then
    echo Trop de paramètres
    exit 1
fi
```

Syntaxe []

8

► **test** *expression* ↔ **[** *expression* **]**

```
if [ $# -gt 5 ]  
then  
    echo Trop de paramètres  
    exit 1  
fi
```


Syntaxe []

9

► **test** *expression* ↔ **[***expression***]**

```
if [ $# -gt 5 ]  
then  
    echo Trop de paramètres  
    exit 1  
fi
```

Les espaces sont importants!

Syntaxe (())

10

- L'opérateur (()) peut également être utilisé pour évaluer une expression booléenne

```
if (($# > 5))  
then  
    echo Trop de paramètres  
    exit 1  
fi
```

Tester une regex

11

► `[["$var" =~ regex]]`

► Négation : `[[! "$var" =~ regex]]`

```
if [[ "$line" =~ ^[a-z]+$ ]]  
then  
    echo Mot composé de minuscules  
fi
```

Tester une regex

- ▶ `[["$var" =~ regex]]`
- ▶ Négation : `[[! "$var" =~ regex]]`

```
LOWCASE_REGEX="^[a-z]+$"  
if [[ "$line" =~ $LOWCASE_REGEX ]]  
then  
    echo Mot composé de minuscules  
fi
```

Succession de commandes (rappel)

13

▶ `cmd1 ; cmd2`

exécute `cmd1` et ensuite `cmd2`

▶ `cmd1 && cmd2`

exécute `cmd1` puis `cmd2` si `cmd1` a réussi

▶ `cmd1 || cmd2`

exécute `cmd1` puis `cmd2` si `cmd1` a échoué

OU logique

14

```
if [ $USER = 'bob' ] || [ $USER = 'andy' ]
then
    ls -alh
else
    ls
fi
```

Alternative :

```
if [ $USER = 'bob' -o $USER = 'andy' ]
then
    ls -alh
else
    ls
fi
```

ET logique

15

```
if [ -r "$1" ] && [ -s "$1" ] ; then  
    echo This file is useful.  
fi
```

Alternative :

```
if [ -r "$1" -a -s "$1" ] ; then  
    echo This file is useful.  
fi
```

ET logique

16

```
if [ -r "$1" ] && [ -s "$1" ] ; then  
    echo This file is useful.  
fi
```

Alternative :

```
if [ -r "$1" -a -s "$1" ] ; then  
    echo This file is useful.  
fi
```

Rappel: TOUJOURS encadrer les paramètres \$i par des **guillemets**!

case

17

```
#!/bin/bash
case $1 in
    start)  # equivalent au test [ "$1" = "start" ]
        echo starting
        ;;   # comme le 'break' de java
    stop)
        echo stopping
        ;;
    restart)
        echo restarting
        ;;
    *)      # comme le 'default:' de java
        echo don\'t know
        ;;
esac
```