

# LES TESTS

---

# Les tests

- Lorsque vous tapez une commande, vous pouvez prendre le temps d'analyser la réponse du système et prendre une décision en fonction de cette réponse.
- Il est tout à fait possible d'effectuer les mêmes étapes avec le scripting Shell grâce aux tests dont la syntaxe est la suivante :

```
[ voici-la-condition-du-test-a-verifier ]
```
- Il est important de respecter les espaces après le [ mais également avant le ].

# Exemple de tests

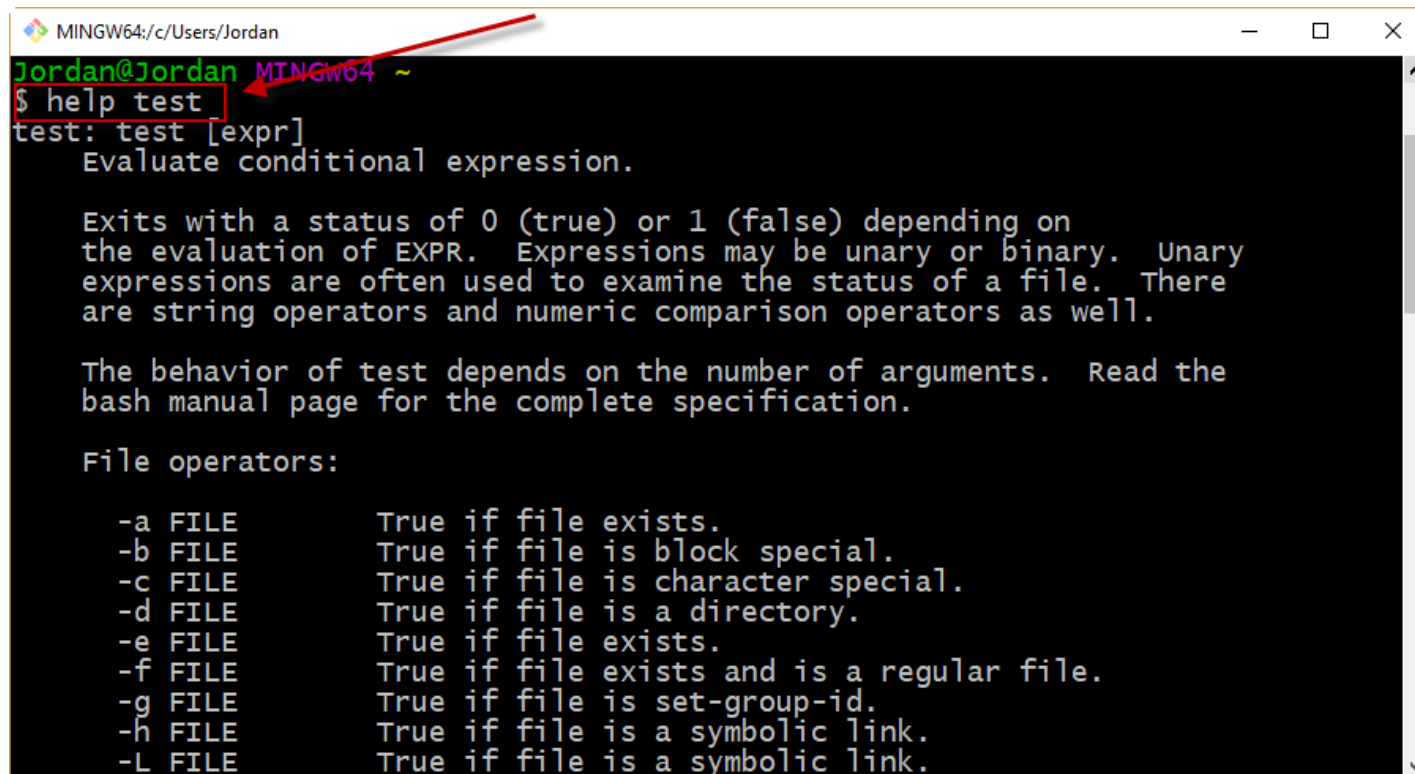
- Vérifie si le fichier /home/jordan/bonjour existe :

```
[ -e /home/jordan/bonjour ]
```

- La commande nous retourne la valeur 0 (True) si le fichier existe
  - La commande nous retourne la valeur 1 (False) si le fichier n'existe pas
- 
- Parmi les opérateurs principaux nous avons :
    - -e : 0 (True) si le fichier existe
    - -d : 0 (True) s'il s'agit d'un dossier
    - -r : 0 (True) si le fichier est disponible en lecture pour l'utilisateur
    - -s : 0 (True) si le fichier existe et n'est pas vide
    - -w : 0 (True) si le fichier est disponible en écriture pour l'utilisateur
    - -x : 0 (True) si le fichier est disponible en exécution pour l'utilisateur

# Quels sont les tests possibles ?

- Vous pouvez utiliser la command 'help test' dans le shell bash pour obtenir les différents types de test qui existent.



```
MINGW64:/c/Users/Jordan
Jordan@Jordan MINGW64 ~
$ help test
test: test [expr]
    Evaluate conditional expression.

    Exits with a status of 0 (true) or 1 (false) depending on
    the evaluation of EXPR.  Expressions may be unary or binary.  Unary
    expressions are often used to examine the status of a file.  There
    are string operators and numeric comparison operators as well.

    The behavior of test depends on the number of arguments.  Read the
    bash manual page for the complete specification.

    File operators:

    -a FILE      True if file exists.
    -b FILE      True if file is block special.
    -c FILE      True if file is character special.
    -d FILE      True if file is a directory.
    -e FILE      True if file exists.
    -f FILE      True if file exists and is a regular file.
    -g FILE      True if file is set-group-id.
    -h FILE      True if file is a symbolic link.
    -L FILE      True if file is a symbolic link.
```

# Les tests sur les chaînes de caractères

- Il est également possible de faire des tests sur des chaînes de caractères.

```
#!/bin/bash
PRENOM='Jordan'
[ -z $PRENOM ]
echo $?
```

```
./script.sh
```

```
1
```

- Le `-z` nous renvoie 0 si la chaîne de caractère est vide et 1 si elle ne l'est pas.
- Le  `$?`  permet de demander d'afficher le retour de la dernière commande lancée

# Les tests sur les chaînes de caractères (2)

- Un autre test possible :

```
#!/bin/bash
PRENOM='Jordan'
[ -n $PRENOM ]
echo $?
```

```
./script.sh
```

```
1
```

- Le `-n` nous renvoie 1 si la chaîne de caractère est vide et 0 si elle ne l'est pas.

# Les tests sur les chaînes de caractères (2)

- Comparer deux chaînes entre elles :

```
#!/bin/bash
PRENOM='Jordan'
NOM='Thomas'
[ $PRENOM = $NOM ]
echo $?
```

```
./script.sh
```

```
1
```

- En effet, on peut comparer les chaînes en utilisant le signe =.
- Le script nous renvoie 0 si les deux chaînes sont identiques et 1 si elles ne le sont pas.

# Les tests sur les chaînes de caractères (2)

- Comparer deux chaînes entre elles :

```
#!/bin/bash
PRENOM='Jordan'
NOM='Thomas'
[ $PRENOM != $NOM ]
echo $?
```

```
./script.sh
```

```
0
```

- On peut vérifier si deux chaînes sont différentes grâce au " != "
- Le script nous renvoie 1 si les deux chaînes sont identiques et 0 si elles sont différentes.



# Les tests sur les chiffres

- De la même manière que les chaînes de caractères, il est tout à fait possible de comparer deux nombres entre eux.
- `chiffre1 -eq chiffre2` : 0 si chiffre1 est égal à chiffre2
- `chiffre1 -ne chiffre2` : 0 si chiffre1 est différent de chiffre2
- `chiffre1 -lt chiffre2` : 0 si chiffre1 est plus petit que chiffre2
- `chiffre1 -le chiffre2` : 0 si chiffre1 est plus petit ou égal que chiffre2
- `chiffre1 -gt chiffre2` : 0 si chiffre1 est plus grand que chiffre2
- `chiffre1 -ge chiffre2` : 0 si chiffre1 est plus grand ou égal que chiffre2