LES CONDITIONS

L'utilisation du if

```
if [ condition-est-vraie ]
then
        command
        command2
fi
```

- Lorsque l'on veut exécuter un certain nombre de commande si la condition est vraie, on utilise le if.

- Ainsi si la condition est vraie, alors le script va exécuter les commandes situées après le "**then**".

- Le "**fi**" marque la fin de la condition.

Exemple d'utilisation

```
#!/bin/bash

touch /home/jordan/bonjour.sh

if [ -e /home/jordan/bonjour.sh ]
then
        echo "Le fichier bonjour.sh a bien été créé"
fi
```

```
./script.sh

Le fichier bonjour.sh a bien été créé
```

L'utilisation du if et du else

```
if [ condition-est-vraie ]
then
        command
else
        command
fi
```

- Si l'on veut agir sur la possibilité que la condition soit fausse, on peut utiliser le else.

- En effet, si la condition est vraie, alors on exécutera les commandes situées après le then.
- Si la condition est fausse, ce seront les commandes situées après le else qui seront exécutées.

Exemple d'utilisation

```
#!/bin/bash

if [ -e /home/jordan/bonjour.sh ]
then
	echo "Le fichier bonjour.sh a bien été créé"
else
	echo "Le fichier bonjour.sh n'a pas été créé"
fi
```

```
./script.sh

Le fichier bonjour.sh n'a pas été créé
```

L'utilisation du if, du elif, et du else

```
if [ condition-est-vraie ]
then
       command
elif [ condition-est-vraie ]
then
       command
else
       command
fi
```

- Le mot elif correspond à la contraction de else et if. En effet, il est possible d'indiquer au script d'exécuter des commandes en fonction de la validité d'une ou de l'autre condition.

Exemple d'utilisation

```
#!/bin/bash
CHIFFRE1='16'
CHIFFRE2='17'
if [ $CHIFFRE1 -lt $CHIFFRE2 ]
then
	echo "$CHIFFRE1 est plus petit que $CHIFFRE2"

elif [ $CHIFFRE1 -gt $CHIFFRE2 ]
then
	echo "$CHIFFRE1 est plus grand que $CHIFFRE2"

else
	echo "$CHIFFRE1 est égal à $CHIFFRE2"
fi
```

```
./script.sh

16 est plus petit que 17
```