Commandes Shell Bash

1. Commandes de Base

echo

- **Fonction**: Afficher du texte à l'écran
- **Syntaxe**: `echo [options] [texte]`
- **Options**:
  - `-n`: N'ajoute pas de saut de ligne
- **Exemples**:

```
echo Bonjour  # Affiche simplement "Bonjour"
echo -n Système  # Affiche sans saut de ligne
```

clear

- **Fonction**: Nettoyer l'écran du terminal
- **Syntaxe**: `clear`
- **Exemple**:

```
clear  # Efface le contenu de l'écran
```

uname

- **Fonction**: Afficher des informations système
- **Syntaxes**:
  - `uname`: Nom du système d'exploitation
  - `uname -r`: Version du noyau
- **Exemples**:

```
uname     # Affiche par exemple "Linux"
uname -r  # Affiche la version du noyau
```

2. Gestion des Variables

Déclaration de Variables

- **Syntaxe**: `nom_variable=valeur`
- **Exemples**:

```
a=100
nom="khalid"
```

Utilisation des Variables

- **Syntaxe**: `$nom_variable`
- **Exemple**:

```
echo $nom  # Affiche la valeur de la variable
```

Variables Spéciales Prédéfinies

- **Fonction**: Contiennent des informations spécifiques
- **Variables**:
  - `$0`: Nom du script
  - `$#`: Nombre de paramètres
  - `$*`: Liste complète des paramètres
  - `$?`: Code de retour de la dernière commande
  - `$$`: Numéro de processus du shell
  - `$1`, `$2`, etc.: Paramètres individuels
- **Exemple**:

```
# Dans un script script.sh ./script.sh param1 param2
echo $0    # Affiche ./script.sh
echo $#    # Affiche 2
echo $*    # Affiche param1 param2
echo $1    # Affiche param1
```

Opérations Arithmétiques

Méthode expr

- **Syntaxe**: `resultat=$(expr arg1 operateur arg2)`
- **Exemple**:

```
som=$(expr 3 + 5)
pro=$(expr $a \* $b)
```

Méthode Bash (Double Parenthèses)

- **Syntaxe**: `resultat=$((expression))`
- **Exemples**:

```
som=$((3+5))
pro=$((a*b))
n3=$((17%3))  # Reste de la division
```

3. Commandes Interactives

read

- **Fonction**: Lire une entrée utilisateur
- **Syntaxe**: `read variable1 variable2 ...`
- **Exemple**:

```
read nom age  # L'utilisateur entre deux valeurs
```

4. Commande shift

- **Fonction**: Décaler les paramètres
- **Syntaxe**: `shift`
- **Exemple**:

```
# Permet de traiter plus de 9 paramètres
```

5. Commandes de Contrôle

if

- **Syntaxes**:

```
# Une alternative
if [ condition ]; then
    commandes
fi

# Deux alternatives
if [ condition ]; then
    commandes1
else
    commandes2
fi

# Plusieurs alternatives
if [ condition1 ]; then
    commandes1
elif [ condition2 ]; then
    commandes2
else
    commandes0
fi
```

case

- **Syntaxe**:

```
case $variable in
    motif1)
        commandes1
        ;;
    motif2)
        commandes2
        ;;
    *)
        commandesdéfaut
        ;;
Esac
```

select

- **Syntaxe**:

```
select variable in choix1 choix2 choix3
do
    commandes
done
```

6. Boucles

while

- **Syntaxe**:

```
while [ condition ]; do
    commandes
    incrémentation
done
```

until

- **Syntaxe**:

```
until [ condition ]; do
    commandes
done
```

for

- **Syntaxes**:

```
# Parcourir une liste
for variable in liste; do
    commandes
done

# Parcourir les arguments
for variable; do
    commandes
done

# Parcourir les fichiers
for variable in *; do
    commandes
done
```

seq

- **Fonction**: Générer une séquence de nombres
- **Syntaxe**: `seq debut fin`
- **Exemple**:

```
for i in $(seq 1 10); do
    echo $i
done
```

7. Commande chmod

- **Fonction**: Modifier les permissions de fichiers
- **Syntaxe**: `chmod [options] permissions fichier`
- **Exemple**:

```
chmod u+x script.sh  # Ajoute droit d'exécution pour l'utilisateur
```

8. Commandes de Lancement de Script

- **Syntaxes**:

```
./script.sh   # Exécution directe
bash script.sh  # Exécution via bash
```

Opérateurs Logiques

- `-o`: OU logique
- `-a`: ET logique
- `!`: Négation