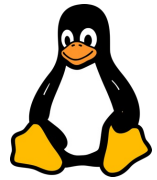
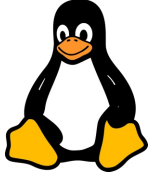


LES SCRIPTS LINUX

PARTIE 6

LES FONCTIONS



Déclaration de Fonctions :

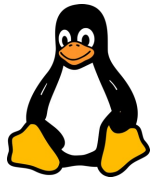
- Syntaxe de base:

```
nom_de_fonction() {
    # Code à exécuter
}
```

- Exemple:

```
dire_bonjour() {
    echo "Bonjour tout le monde!"
}
```

LES FONCTIONS



Appel de Fonctions :

- Exemple:

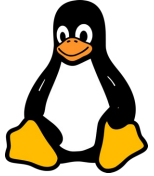
```
dire_bonjour  # Affiche "Bonjour tout le monde!"
```

Paramètres de Fonction :

- Passage de paramètres.
  Accès aux paramètres via $1, $2, etc.
- Exemple:

```
saluer() {
    echo "Salut $1"
}
saluer "Alice"  # Affiche "Salut Alice"
```

LES FONCTIONS



Valeur de Retour des Fonctions en Bash :

Cette section traite de la manière dont les fonctions Bash retournent des valeurs à l'appelant. Il est important de distinguer entre la sortie standard (ce qui est affiché à l'écran) et la valeur de retour (le statut de sortie de la fonction).

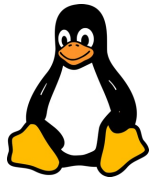
Utilisation de return dans les Fonctions

- But de return: La commande return est utilisée pour quitter une fonction et retourner un statut de sortie (un nombre entre 0 et 255) à l'environnement appelant.

- Syntaxe: return [valeur]
  - Si aucune valeur n'est fournie, le statut de sortie de la dernière commande exécutée dans la fonction est retourné.
- Exemple:

```
verifier_nombre() {
if [ $1 -gt 10 ]; then
return 1
else
return 0
fi
}
```

LES FONCTIONS

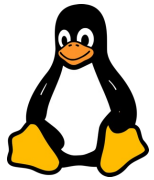


Capturer la Valeur de Retour :

- Utilisation de $?: Après l'appel d'une fonction, la variable spéciale $? contient le code de retour de la dernière commande exécutée, qui, dans ce cas, est la fonction appelée.
- Exemple:

```
verifier_nombre 15
echo $?  # Affichera 1 car 15 est supérieur à 10
```

LES FONCTIONS



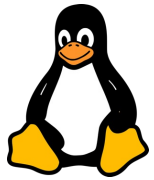
Retourner des Valeurs Complexes :

- Limitation de return: return ne peut retourner que des nombres entiers.
- Utilisation de echo pour des valeurs complexes:
  - Pour retourner des chaînes ou des valeurs plus complexes, utilisez echo ou d'autres commandes d'impression.
  - Capturez ces valeurs en redirigeant la sortie de la fonction.
- Exemple avec echo:

```
obtenir_nom() {
    echo "Alice"
}

nom=$(obtenir_nom)
echo "Le nom est $nom"  # Affichera "Le nom est Alice"
```

LES FONCTIONS



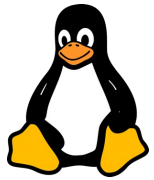
Portée Globale :

- Définition: Une variable ayant une portée globale est accessible partout dans le script, après sa déclaration.
- Comportement par défaut: En Bash, si vous déclarez une variable en dehors de toute fonction, elle est globale par défaut.
- Exemple:

```
variable_globale="Je suis globale"
ma_fonction() {
    echo $variable_globale  # Affiche "Je suis globale"
}
```

Modification: Une variable globale peut être modifiée n'importe où dans le script, y compris à l'intérieur des fonctions.

LES FONCTIONS



Portée Locale :

- Définition: Les variables locales sont celles qui sont déclarées au sein d'une fonction et ne sont accessibles que dans cette fonction.
- Déclaration: Utilisez le mot-clé local pour déclarer des variables locales.
- Exemple:

```
ma_fonction() {
    local variable_locale="Je suis locale"
    echo $variable_locale  # Affiche "Je suis locale"
}
echo $variable_locale  # Erreur ou rien, car $variable_locale n'est pas définie globalement
```

Portée et Durée de Vie: Une variable locale n'existe que pendant l'exécution de la fonction. Après cela, elle n'est plus accessible.