Por: Alice Dantas

Funções no Shell Script

Funções são blocos de código que você pode definir uma vez e chamar quantas vezes quiser, em qualquer parte do seu script. Elas ajudam a deixar o código mais organizado, reutilizável e fácil de entender.

- Declarando uma função:

```
nome_da_funcao() {
    # comandos
}
```

- Chamando uma função:

Basta escrever o nome da função no script:

```
nome_da_funcao
```

- Parâmetros na função:

Dentro da função você pode acessar os parâmetros usando **$1**, **$2**, **...**

```
#!/bin/bash

saudacao() {
    echo "Olá, $1! Seja bem-vindo(a)!"
}

# Chamando a função passando um argumento
saudacao "Alice"
```

Saída:

```
Olá, Alice! Seja bem-vindo(a)!
```

- Retorno de valores em funções:

Em outras linguagens **return** serve para devolver um valor (string, número, lista, etc.) da função para quem chamou. No Bash o **return** não retorna valores comuns como texto ou números arbitrários. Ele serve exclusivamente para retornar um código de status, que é sempre um número inteiro de 0 a 255.

O que é um código de status?

Um código que indica se o comando anterior ou a função foi bem-sucedida ou falhou. Se:
0 → sucesso
Diferente de 0 → erro (1, 2, 3... podem ter significados específicos, se você quiser)

Por: Alice Dantas

Exemplo usando return:

```
verificar() {
    if [[ -f "$1" ]]; then
        return 0    # sucesso
    else
        return 1    # erro
    fi
}

verificar "arquivo.txt"
if [[ $? -eq 0 ]]; then
    echo "Arquivo existe."
else
    echo "Arquivo NÃO existe."
fi
```

Saída se o arquivo existir:

```
Arquivo existe.
```

Se não existir:

```
Arquivo NÃO existe.
```

- Retornando valores "de verdade" (texto, números) no Bash

Usamos **echo** dentro da função e capturamos com **$(...)**

Exemplo:

```
soma() {
    resultado=$(( $1 + $2 ))
    echo $resultado
}

# Capturando o "retorno"
valor=$(soma 5 3)
echo "Resultado: $valor"
```

Saída:

```
Resultado: 8
```