Understanding variables, loops, conditionals in centos

Topics : Centos Server
Written on March 05, 2024

Understanding variables, loops, conditionals, and other fundamental concepts in shell scripting is essential for writing effective scripts to automate tasks on CentOS. Here's an overview of each concept:

1. Variables:

- Variables are placeholders used to store data or values.
- Declare variables using the syntax `variable_name=value`.
- Access variables using the `$` prefix (`$variable_name`).
- Examples:

```
name="John"
age=30
echo "Hello, $name! You are $age years old."
```

2. Loops:

- Loops allow you to repeat a set of commands multiple times.
- Common types of loops in shell scripting are `for`, `while`, and `until`.
- Examples:
  - `for` loop:

```
for i in {1..5}
do
    echo "Iteration: $i"
done
```

  - `while` loop:

```
count=0
while [ $count -lt 5 ]
do
    echo "Count: $count"
    ((count++))
done
```

3. Conditionals:

- Conditionals allow you to execute commands based on the evaluation of a condition.
- Common conditional constructs are `if`, `elif`, and `else`.
- Examples:

```
age=20
if [ $age -ge 18 ]; then
    echo "You are an adult."
else
    echo "You are not yet an adult."
fi
```

4. Functions:

- Functions allow you to encapsulate a block of code that performs a specific task.
- Declare functions using the syntax `function_name() { ... }`.
- Call functions by their name.
- Example:

```
greet() {
    echo "Hello, $1!"
}

greet "Alice"
```

5. Input/Output:

- Read user input using the `read` command.
- Redirect output using redirection operators (>, >>, <, |, etc.).
- Examples:
    - Read user input:

      ```
      echo "Enter your name:"
      read name
      echo "Hello, $name!"
      ```

    - Redirect output to a file:

      ```
      echo "Hello, world!" > output.txt
      ```

6. Command Substitution:

- Command substitution allows you to capture the output of a command and use it as a value.
- Use backticks (`) or `$()` syntax for command substitution.
- Example:

```
files_count=$(ls | wc -l)
echo "Number of files: $files_count"
```

7. Comments:

- Comments provide explanatory notes or annotations in your script.
- Comments start with the # symbol and extend to the end of the line.
- Example:

```
# This is a comment
```