Control Flow in Shell Scripts

---

1. Conditional Statements

Introduction

In programming, **conditional statements** let the script make decisions. Instead of executing commands in a straight line, the script can choose different paths depending on conditions.

A. if Statement

- Basic syntax:

```
if [ condition ]
then
  commands
fi
```

👉 **Example**:

```
#!/bin/bash
num=10
if [ $num -gt 5 ]
then
  echo "Number is greater than 5"
fi
```

B. if-else Statement

- Syntax:

```
if [ condition ]
then
  commands_if_true
else
  commands_if_false
fi
```

👉 **Example**:

```
#!/bin/bash
age=18
if [ $age -ge 18 ]
then
  echo "You are eligible to vote"
else
  echo "You are not eligible"
fi
```

C. if-elif-else Statement

- Syntax:

```
if [ condition1 ]
then
  commands1
elif [ condition2 ]
then
  commands2
else
  commands3
fi
```

👉 **Example**:

```
#!/bin/bash
marks=75
if [ $marks -ge 90 ]
then
  echo "Grade: A"
elif [ $marks -ge 60 ]
then
```

```
  echo "Grade: B"
else
  echo "Grade: C"
fi
```

D. Test Conditions

The test command (or [ ]) checks conditions.

- **Numeric tests**
  - -eq → equal
  - -ne → not equal
  - -lt → less than
  - -le → less than or equal
  - -gt → greater than
  - -ge → greater than or equal
- **File tests**
  - -f file → file exists and is a regular file
  - -d dir → directory exists
  - -r file → file is readable
  - -w file → file is writable
  - -x file → file is executable

👉 **Example**:

```
#!/bin/bash
if [ -f /etc/passwd ]
then
  echo "passwd file exists"
else
  echo "passwd file missing!"
fi
```

Task (Conditionals)

1. Write a script that checks if a number entered by the user is **even or odd**.
2. Write a script that checks if a file data.txt exists. If yes, print its size. If no, print "File not found".

DevOps Relevance

- Scripts check whether log files exist before processing.
- Conditions help decide if a service should restart or not.
- Example: In a deployment script, **"If config file exists → backup it, else skip."**

---

2. Looping

Introduction

Loops help repeat tasks without rewriting code.

A. for Loop

- Syntax:

```
for var in list
do
  commands
done
```

👉 **Example (list of names)**:

```
#!/bin/bash
for name in Alice Bob Charlie
do
  echo "Hello $name"
done
```

👉 **Example (loop through files)**:

```
for file in *.txt
do
```

```
  echo "File: $file"
done
```

B. while Loop

- Syntax:

```
while [ condition ]
do
  commands
done
```

👉 **Example**:

```
#!/bin/bash
count=1
while [ $count -le 5 ]
do
  echo "Count is $count"
  count=$((count+1))
done
```

C. case Statement (Switch-Case Equivalent)

- Syntax:

```
case $variable in
  pattern1)
    commands ;;
  pattern2)
    commands ;;
  *)
    default ;;
esac
```

👉 **Example**:

```
#!/bin/bash
echo "Enter a choice (start/stop/restart): "
read choice

case $choice in
  start)
    echo "Service starting..." ;;
  stop)
    echo "Service stopping..." ;;
  restart)
    echo "Service restarting..." ;;
  *)
    echo "Invalid option" ;;
esac
```

Task (Loops)

1. Write a script to print numbers from **1 to 10** using a while loop.
2. Write a script using a for loop to list all .sh files in the current directory.
3. Create a case script where a user can choose **add, subtract, multiply, divide** and the script performs that operation on two numbers.

DevOps Relevance

- **for loops** process multiple log files or configs.
- **while loops** keep checking service health until success.
- **case statements** are used in deployment scripts (e.g., start | stop | restart service).

---

3. User Interaction

Introduction

Shell scripts often need to take input from users or redirect input/output from files.

A. Accepting Input with read

```
#!/bin/bash
echo "Enter your name: "
read name
echo "Hello, $name!"
```

- Use -p option for inline prompt:

  ```
  read -p "Enter age: " age
  ```

B. Input & Output Redirection

- > → Redirect output (overwrite file)

  ```
  echo "Hello" > file.txt
  ```

- >> → Append output to file

  ```
  echo "World" >> file.txt
  ```

- < → Take input from file

  ```
  wc -l < file.txt
  ```

- 2> → Redirect errors

  ```
  ls /wrongpath 2> errors.log
  ```

Example

```
#!/bin/bash
read -p "Enter filename: " fname
if [ -f "$fname" ]
then
  echo "File exists. Content is:"
  cat "$fname"
else
  echo "File not found!" > errors.log
fi
```

Task (User Interaction)

1. Write a script that asks for your **name and city** and prints:
   "Hello <name>, you are from <city>!"
2. Create a script that redirects command output to a file named output.log.
3. Write a script that stores all error messages into errors.txt.

DevOps Relevance

- User input is useful in interactive scripts (e.g., choosing environment: dev/test/prod).
- Redirection is heavily used:
  - Logs redirected to /var/log/.
  - Error outputs saved for debugging.
  - Scripts write results into files for monitoring systems.

EXERCISE

☞ Write a script menu.sh that:

1. Displays a menu:

    1. Show Date
    2. Show Current Users
    3. Show Disk Usage
    4. Exit

2. Accepts user input using read.
3. Uses a case statement to execute the correct option.
4. Redirects output to menu.log and errors to menu_errors.log.