Lab-04

Bipasha Majumder

Lecturer, CSE, DIU

1) How do you use the for loop to iterate through a list of values?

```
#!/bin/bash
fruits=("apple" "Orange" "cherry")
for fruit in "${fruits[@]}"; do
echo "Current fruit: $fruit"
done
```

Explanation:

`fruits=` line creates an array named fruits with three elements: "apple", “Orange", and "cherry".

•**for fruit in "${fruits[@]}"; do:** This line starts a for loop. Here's what each part means:

- •**for fruit:** This declares a loop variable called fruit. In each iteration of the loop, fruit will hold the value of the current element from the fruits array.
- •**"${fruits[@]}":** This is an array expansion that takes all the elements from the fruits array. The "${...}" syntax ensures that each element is treated as a separate item.
- •**do:** This keyword marks the beginning of the loop body.

•**echo "Current fruit: $fruit":** Inside the loop, this line uses the echo command to display the current value of the loop variable fruit. It prints a message like "Current fruit: apple" for each fruit in the array.

•**done:** This keyword marks the end of the loop body. It tells the script that the loop has finished.

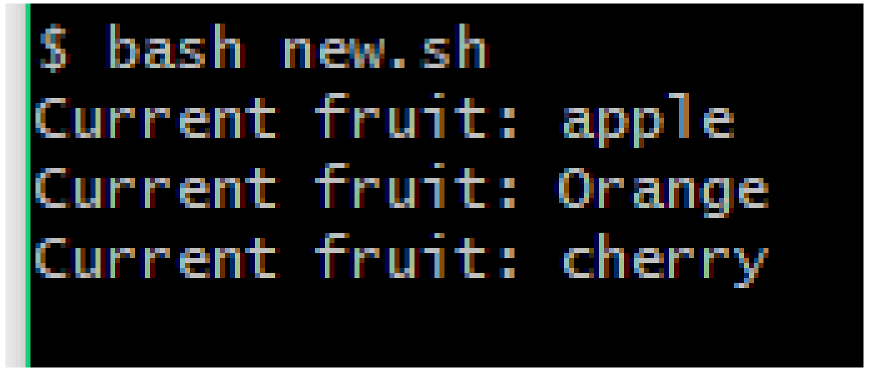
In Bash, the **semicolon (;)** is used to **end a command** *when multiple commands appear on the same line*.
Why is it needed in your script?

for fruit in "${fruits[@]}"; do
Here:
•for fruit in "${fruits[@]}" is **one command**.
•do must start **after that command ends**.

Bash allows two ways to separate them:
Option 1: Use semicolon on the same line
for fruit in "${fruits[@]}"; do
Option 2: Put do on the next line (no semicolon needed)
for fruit in "${fruits[@]}“
do

Both are correct because:
• ; means “end of command”
• **newline** also means “end of command”

Output:



```
$ bash new.sh
Current fruit: apple
Current fruit: Orange
Current fruit: cherry
```

2) Write a shell script that calculates the sum of integers from 1 to N using a loop.

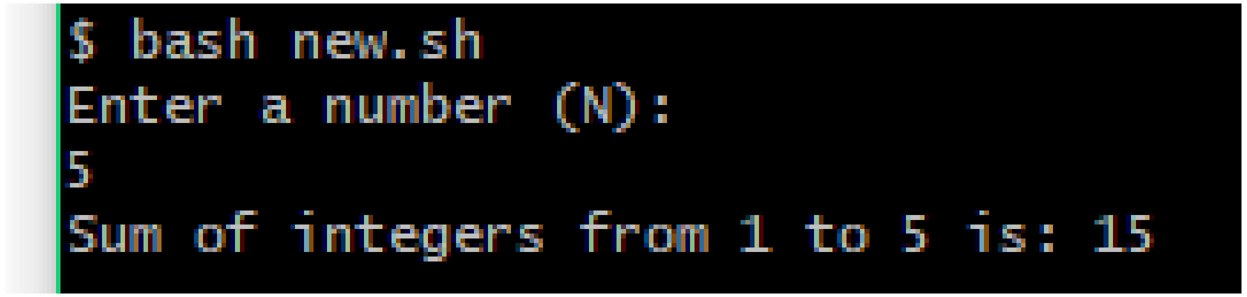
```
#!/bin/bash
echo "Enter a number (N):"
read N
sum=0
for (( i=1; i<=$N; i++ )); do
sum=$((sum + i))
done
echo "Sum of integers from 1 to $N is: $sum"
```

Explanation:

The script starts by asking you to enter a number (N) using read. This number will determine how many times the loop runs.

1. The variable sum is initialized to 0. This variable will keep track of the sum of integers.
2. The for loop begins with for ((i=1; i<=$N; i++)). This loop structure is used to repeat a set of actions a certain number of times, in this case, from 1 to the value of N.
3. Inside the loop, these things happen:
   - i=1 sets the loop variable i to 1 at the beginning of each iteration.
   - The loop condition i<=$N checks if i is still less than or equal to the given number N.
   - If the condition is true, the loop body executes.
   - sum=$((sum + i)) calculates the new value of sum by adding the current value of i to it. This adds up the integers from 1 to the current i value.
4. After each iteration, i++ increments the value of i by 1.
5. The loop continues running until the condition i<=$N becomes false (when i becomes greater than N).
6. Once the loop finishes, the script displays the sum of the integers from 1 to the entered number N.

Output:



```
$ bash new.sh
Enter a number (N):
5
Sum of integers from 1 to 5 is: 15
```

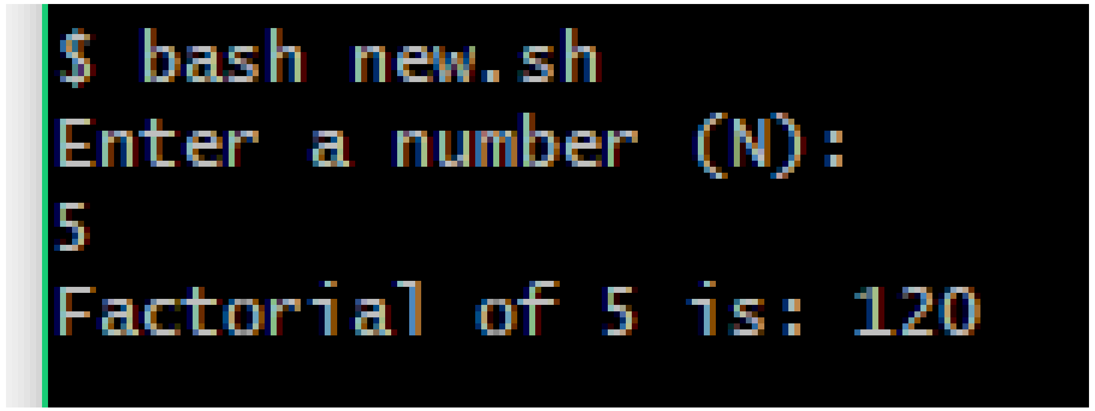
3) Write a Bash script that takes a number *N* from the user and calculates the factorial of *N* using a loop.

Example:

If input = 5 → factorial = 120

(5! = 5 × 4 × 3 × 2 × 1)

```
#!/bin/bash
echo "Enter a number (N):"
read N
fact=1
for (( i=1; i<=N; i++ )); do
   fact=$((fact * i))
done
echo "Factorial of $N is: $fact"
```



```
$ bash new.sh
Enter a number (N):
5
Factorial of 5 is: 120
```

Practice Problem:

1. Sum of Even Numbers
Write a script that calculates the sum of all even integers from 1 to N.

2. Reverse Loop
Write a script to print numbers from N down to 1 using a reverse for loop.

3. Count the Number of Elements in a Fruit Array
Create an array containing the names of fruits. Then, write a Bash script that prints the total number of elements present in the array.