

Department Of Cybersecurity

Linux Programming

Assignment 9

Name: Mohd Sahil Ansari

USN: ENG24CY0134


Roll No:22

Class: 3A

1. Write a shell script using **if...else** to check if a number is even or odd. (CO4)

Ans:

```
6  #!/bin/bash
7
8  echo "prompt a number:"
9  read num
10
11 if [ $((num % 2)) -eq 0 ]; then
12     echo "$num is even"
13 else
14     echo "$num is odd"
15 fi
16 whoami
17
```



2. Explain the difference between **if** and **case** statements in bash. (CO4)

Ans:

If- Statement	Case Statement
---------------	----------------

Used when dealing with multiple different conditions.	Apt when dealing with pattern matching and also in string comparisons.
Supports comparison operators like -eq, -lt, -gt, -ne	In use when dealing with multiple string values
Can handle ranges and complex Boolean expressions	Goes with regular and wildcard patterns as well
Best choice for numerical comparisons tests	Faster execution for multiple string comparisons


3. Write a script to find the **largest of three numbers** entered by the user. (CO4)

Ans:

```

5  #!/bin/bash
6
7  echo "Enter three numbers:"
8  read -p "First number: " n1
9  read -p "Second number: " n2
10 read -p "Third number: " n3
11
12 if [ $n1 -ge $n2 ] && [ $n1 -ge $n3 ]; then
13     echo "Largest number is: $n1"
14 elif [ $n2 -ge $n1 ] && [ $n2 -ge $n3 ]; then
15     echo "Largest number is: $n2"
16 else
17     echo "Largest number is: $n3"
18 fi
19
20 whoami

```



```

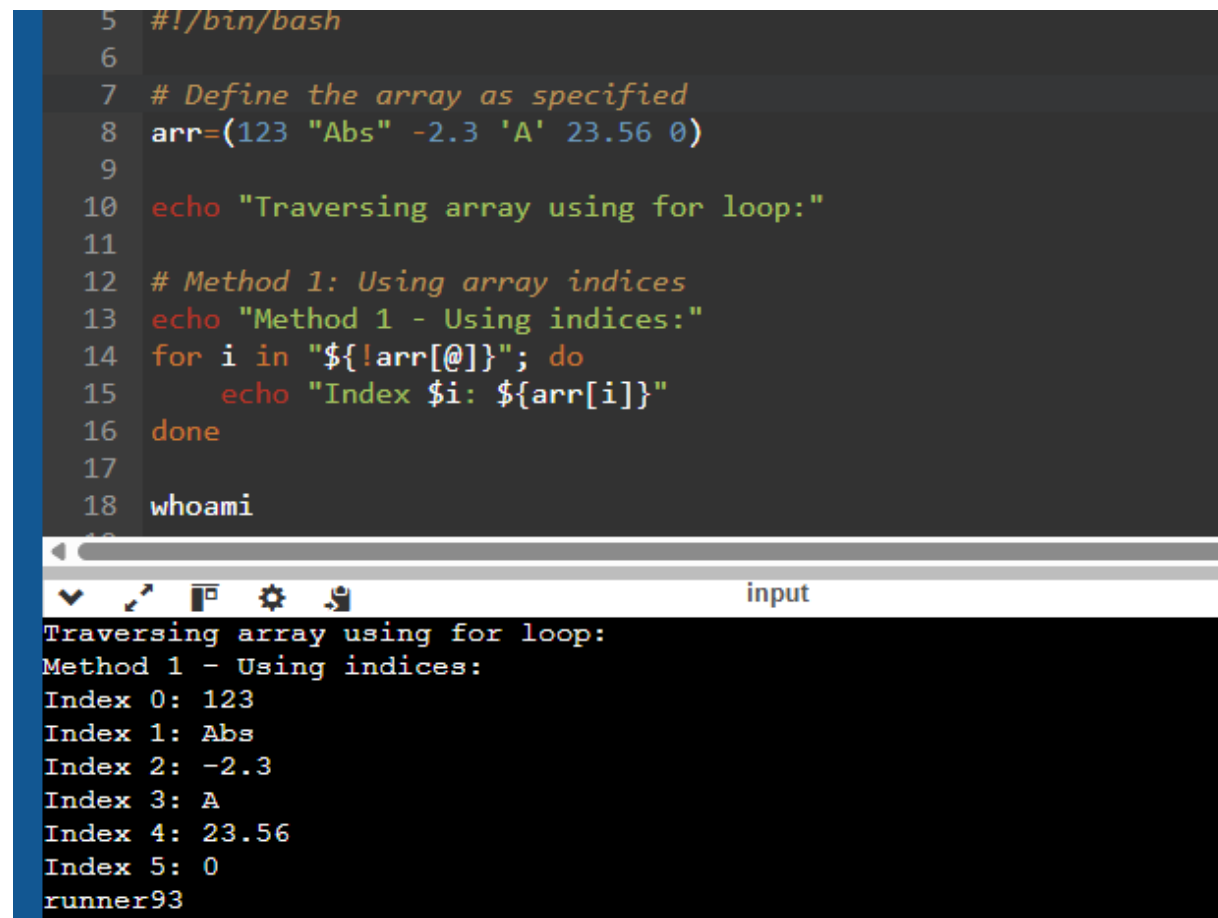
Enter three numbers:
First number: 54
Second number: 67
Third number: 78
Largest number is: 78
runner56

```

4. How do you use a **for loop** to traverse an array in bash? Give an example. The array is defined as **arr=(123, "Abs", -2.3, 'A', 23.56, 0)**. (CO4)

Ans: For loops provide a reliable and optimised mechanism to iterate through the array elements. The main purpose of loops is to reduce time and fuel the work efficiency in any block of specified code.

```
5  #!/bin/bash
6
7  # Define the array as specified
8  arr=(123 "Abs" -2.3 'A' 23.56 0)
9
10 echo "Traversing array using for loop:"
11
12 # Method 1: Using array indices
13 echo "Method 1 - Using indices:"
14 for i in "${!arr[@]}"; do
15     echo "Index $i: ${arr[i]}"
16 done
17
18 whoami
```



Traversing array using for loop:
Method 1 - Using indices:
Index 0: 123
Index 1: Abs
Index 2: -2.3
Index 3: A
Index 4: 23.56
Index 5: 0
runner93

5. Write a shell script to **loop through all files in the current directory** and display their names. (C04)

```
4 #!/bin/bash
5
6 echo "Detailed file information:"
7 for item in *; do
8     if [ -f "$item" ]; then
9         echo "📄 File: $item"
10    elif [ -d "$item" ]; then
11        echo "📁 Directory: $item"
12    elif [ -L "$item" ]; then
13        echo "🔗 Link: $item"
14    fi
15 done
16 whoami
17
```

runner80

Ans:

6. What is the difference between **while and until loops** in bash? (CO4)

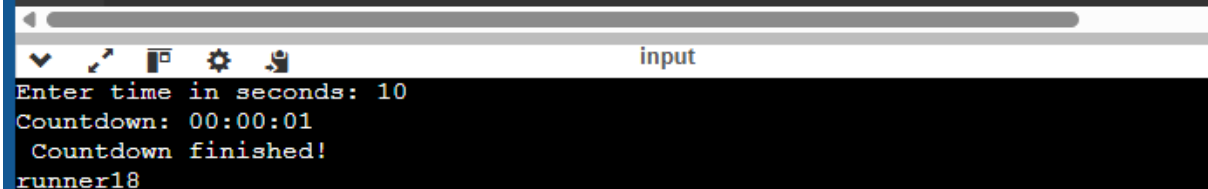
Ans:

While Loop	Until loop
In process until conditions are met.	Executes as long as set of commands are FALSE.
Continues when condition results in 0(success).	Continues when condition results to non-zero(failure)
Used when unsure about number of iterations to be performed.	Comes to work when waiting for condition to be true.

7. Write a **countdown timer script** using a while loop. (CO4)

Ans:

```
6  #!/bin/bash
7  countdown_timer() {
8      local sec=$1
9
10     while [ $sec -gt 0 ]; do
11         local hours=$((sec / 3600))
12         local minutes=$(( (sec % 3600) / 60 ))
13         local secs=$((sec % 60))
14
15         printf "\rCountdown: %02d:%02d:%02d" $hours $minutes $secs
16         sleep 1
17         sec=$((sec - 1))
18     done
19
20     echo -e "\n Countdown finished!"
21 }
22
23 read -p "Enter time in seconds: " input_time
24 countdown_timer $input_time
25
26 whoami
27
```



```
Enter time in seconds: 10
Countdown: 00:00:10
Countdown: 00:00:09
Countdown: 00:00:08
Countdown: 00:00:07
Countdown: 00:00:06
Countdown: 00:00:05
Countdown: 00:00:04
Countdown: 00:00:03
Countdown: 00:00:02
Countdown: 00:00:01
Countdown finished!
runner18
```

8. How do you use **break and continue statements** in loops? Give examples. (CO4)

Ans:

Break Statement	Continue statement
At the very instance it terminates the loop ,when its encountered	When encountered, skips the remaining statements in current presentation.
Statement followed by the loop gets the control.	Moves to next iteration of loop
Can specify nesting level with break n.	Can specify nesting level with continue n.

Example:

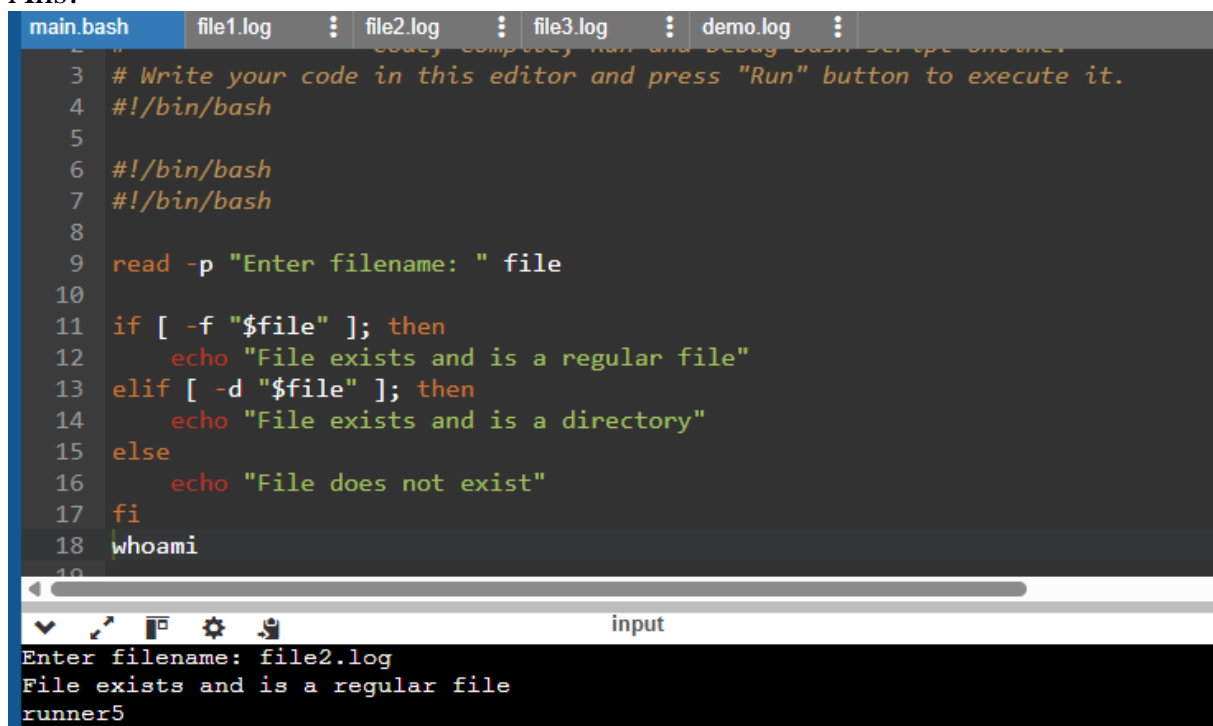
```

7  #!/bin/bash
8
9  echo "Demonstrating break and continue statements:"
10
11 # Example 1: break statement
12 echo "Break statement (1-5, stop at 3):"
13 for i in {1..5}; do
14     if [ $i -eq 3 ]; then
15         echo "Breaking at $i"
16         break
17     fi
18     echo "Number: $i"
19 done
20
21 echo
22
23 # Example 2: continue statement
24 echo "Continue statement (1-5, skip even numbers):"
25 for i in {1..5}; do
26     if [ $((i % 2)) -eq 0 ]; then
27         continue
28     fi
29     echo "Odd number: $i"
30 done
31 whoami

```

9. Write a script to check if a **file exists or not using the if and else loop**.
(CO4)

Ans:



```

main.bash  file1.log  file2.log  file3.log  demo.log
3  # Write your code in this editor and press "Run" button to execute it.
4  #!/bin/bash
5
6  #!/bin/bash
7  #!/bin/bash
8
9  read -p "Enter filename: " file
10
11 if [ -f "$file" ]; then
12     echo "File exists and is a regular file"
13 elif [ -d "$file" ]; then
14     echo "File exists and is a directory"
15 else
16     echo "File does not exist"
17 fi
18 whoami
19
input
Enter filename: file2.log
File exists and is a regular file
runner5

```

10. Write a script to calculate **factorial of a number** using for loop. (CO4)

Ans:

```
4  #!/bin/bash
5  echo "Factorial Calculator"
6  # Function to calculate factorial
7  calc_fact() {
8      local number=$1
9      local factorial=1
10
11      if [ $number -lt 0 ]; then
12          echo "Error: Factorial is not defined for negative numbers"
13          return 1
14      fi
15
16      if [ $number -eq 0 ] || [ $number -eq 1 ]; then
17          echo "Factorial of $number is: 1"
18          return 0
19      fi
20
21      for ((i=1; i<=number; i++)); do
22          factorial=$((factorial * i))
23      done
24
25      echo "Factorial of $number is: $factorial"
26  }
27
28  # Input validation function
29  validate_input() {
30      local input=$1
31      if [[ ! "$input" =~ ^[0-9]+$ ]]; then
32          echo "Error: Please enter a non-negative integer"
33          return 1
34      fi
35      return 0
36  }
```

```

38 # Main program
39 while true; do
40     read -p "Enter a non-negative integer (or 'quit' to exit): " input
41
42     if [ "$input" = "quit" ]; then
43         echo "Goodbye!"
44         break
45     fi
46
47     if validate_input "$input"; then
48         calc_fact "$input"
49     fi
50     echo
51 done
52 whoami
53

```

input

```

Enter a non-negative integer (or 'quit' to exit): -21
Error: Please enter a non-negative integer

Enter a non-negative integer (or 'quit' to exit): 5
Factorial of 5 is: 120

Enter a non-negative integer (or 'quit' to exit): 1
Factorial of 1 is: 1

Enter a non-negative integer (or 'quit' to exit): 0
Factorial of 0 is: 1

Enter a non-negative integer (or 'quit' to exit): quit
Goodbye!
runner35

```