Linux Programming - Assignment 9

Name – Suraj P Das
Batch - 3C
Roll no. - 66
USN - ENG25CY1006

1. Q21. Write a shell script using if...else to check if a number is even or odd.

Use modulus operator and arithmetic evaluation.

```
#!/bin/bash
read -p "Enter a number: " n
if (( n % 2 == 0 )); then
  echo "$n is even"
else
  echo "$n is odd"
fi
```

2. Q22. Explain the difference between if and case statements in bash.

if evaluates boolean conditions (test expressions, command exit codes) and is good for range checks and complex logic. case matches a value against patterns and is preferable for fixed-value dispatch (like a switch), pattern matching, and less verbose multi-way branching.

3. Q23. Write a script to find the largest of three numbers entered by the user.

Compare numbers sequentially to determine the maximum.

```
#!/bin/bash
read -p "Enter a: " a
read -p "Enter b: " b
read -p "Enter c: " c
max=$a
if (( b > max )); then max=$b; fi
if (( c > max )); then max=$c; fi
echo "Largest = $max"
```

4. Q24. 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).

Define the array without commas and iterate over indices or elements. Example code below.

```
#!/bin/bash
arr=(123 "Abs" -2.3 A 23.56 0)
```

```
for item in "${arr[@]}"; do
  echo "$item"
done
# or with indices
for i in "${!arr[@]}"; do
  echo "arr[$i]=${arr[$i]}"
done
```

5. Q25. Write a shell script to loop through all files in the current directory and display their names.

Use a for loop or find; handle spaces safely using a while read loop from find null-separated.

```
#!/bin/bash
for f in *; do
  [ -e "$f" ] || continue
  echo "$f"
done
# safer alternative for deep lists:
# find . -maxdepth 1 -type f -print0 | while IFS= read -r -d '' file;
do echo "$file"; done
```

6. Q26. What is the difference between while and until loops in bash?

while executes as long as the condition is true. until executes until the condition becomes true (i.e., it runs while the condition is false). until is logically the inverse of while.

7. Q27. Write a countdown timer script using a while loop.

Use sleep to wait one second between iterations.

```
#!/bin/bash
read -p "Enter seconds for countdown: " t
while (( t > 0 )); do
  echo "$t"
  sleep 1
  ((t--))
done
echo "Time's up!"
```

8. Q28. How do you use break and continue statements in loops? Give examples.

break exits the current loop immediately; continue skips the rest of the loop body and proceeds to the next iteration. Examples:

```
#!/bin/bash
for i in {1..10}; do
  if (( i == 5 )); then
    echo "skip 5"
    continue
  fi
  if (( i == 8 )); then
```

```
    echo "stop at 8"
    break
  fi
  echo "i=$i"
done
```

9. Q29. Write a script to check if a file exists or not using the if and else loop.

Test file existence with -e or -f.

```
#!/bin/bash
read -p "Enter filename: " file
if [ -e "$file" ]; then
  echo "File exists: $file"
else
  echo "File does not exist: $file"
fi
```

10. Q30. Write a script to calculate factorial of a number using for loop.

Compute factorial iteratively; handle 0! = 1.

```
#!/bin/bash
read -p "Enter a non-negative integer: " n
fact=1
for (( i=2; i<=n; i++ )); do
  fact=$((fact * i))
done
echo "$n! = $fact"
```