## 50 Linux and Bash scripting interview questions, along with their answers. These cover a range of difficulty levels, from beginner to advanced:

## Follow – Krishan Bhatt

## General Linux and Bash Basics

1. **What is Bash?**

- Bash (Bourne Again Shell) is a command-line interpreter and scripting language for Linux/Unix systems, commonly used to automate tasks.

2. **How do you make a script executable?**

- Use the chmod command:

```
chmod +x script.sh
```

3. **How do you run a Bash script?**

- Use ./script.sh or bash script.sh.

4. **How can you store the output of a command in a variable?**

```
output=$(command)
```

5. **What is the difference between $* and $@ in Bash?**

- $* treats all arguments as a single string.
- $@ treats each argument as a separate string.

6. **How do you check if a file exists in Bash?**

```
if [ -f filename ]; then
    echo "File exists"
fi
```

7. **How do you check if a directory exists in Bash?**

```
if [ -d dirname ]; then
    echo "Directory exists"
fi
```

8. **What does** #!/bin/bash **mean at the start of a script?**

- It specifies the interpreter to use for the script (Bash in this case).

9. **How do you read input from a user in a Bash script?**

```
read -p "Enter your name: " name
echo "Hello, $name"
```

10. **How do you pass arguments to a Bash script?**

- Use $1, $2, etc. Example:

```
echo "First argument: $1"
```

## Intermediate Bash Scripting

11. **Write a script to find all** .txt **files in a directory.**

```
find /path/to/dir -type f -name "*.txt"
```

12. **How do you schedule a script to run daily?**

- Use cron:

```
crontab -e
0 0 * * * /path/to/script.sh
```

13. **Write a script to check if a number is even or odd.**

```
read -p "Enter a number: " num
if (( num % 2 == 0 )); then
    echo "Even"
else
    echo "Odd"
fi
```

14. **What is the purpose of the** trap **command in Bash?**

- To catch signals and execute specific commands. Example:

```
trap "echo 'Signal caught!'" SIGINT
```

15. **How do you append to a file in Bash?**

```
echo "Text to append" >> filename
```

16. **Write a script to reverse a string in Bash.**

```
read -p "Enter a string: " str
echo "$str" | rev
```

17. **What does** set -e **do in a script?**

- It exits the script immediately if a command fails.

18. **How do you declare and use an array in Bash?**

```
arr=("one" "two" "three")
echo "${arr[0]}"  # Access first element
```

19. **Write a script to count the number of lines in a file.**

```
wc -l filename
```

20. **How do you loop over files in a directory?**

```
for file in /path/to/dir/*; do
    echo "$file"
done
```

## Advanced Bash Scripting

21. **How do you debug a Bash script?**

- Use bash -x script.sh or add set -x for debugging.

22. **Write a script to calculate factorial using recursion.**

```
factorial() {
    if (( $1 <= 1 )); then
        echo 1
    else
        echo $(( $1 * $(factorial $(( $1 - 1 ))) ))
    fi
}
read -p "Enter a number: " num
echo "Factorial: $(factorial $num)"
```

23. **Explain the difference between** exec **and** source**.**

- source executes a script in the current shell, while exec replaces the current shell with the script.

24. **Write a script to check if a process is running.**

```
if pgrep "process_name" > /dev/null; then
    echo "Running"
else
    echo "Not running"
fi
```

25. **How do you create a function in Bash?**

```
my_function() {
    echo "This is a function"
}
my_function
```

26. **Write a script to check disk usage.**

```
df -h
```

27. **How do you compare two strings in Bash?**

```
if [ "$str1" == "$str2" ]; then
    echo "Equal"
fi
```

28. **Write a script to delete files older than 7 days.**

```
find /path/to/dir -type f -mtime +7 -exec rm {} \;
```

29. **What is the difference between > and >> in Bash?**

- > overwrites a file, while >> appends to it.

30. **How do you send an email from a Bash script?**

```
echo "Message" | mail -s "Subject" user@example.com
```

## System Administration and Scripting Challenges

31. **How do you monitor system load in a script?**

```
uptime
```

32. **Write a script to create a backup of a directory.**

```
tar -czf backup.tar.gz /path/to/dir
```

### 33. How do you check memory usage in a script?

```
free -h
```

### 34. Write a script to find duplicate files.

```
find . -type f -exec md5sum {} + | sort | uniq -w32 -d
```

### 35. How do you handle errors in a script?

```
command || echo "Error occurred"
```

### 36. Write a script to compress all .log files in a directory.

```
gzip /path/to/dir/*.log
```

### 37. How do you parse a CSV file in Bash?

```
while IFS=, read -r col1 col2; do
    echo "Col1: $col1, Col2: $col2"
done < file.csv
```

### 38. Write a script to display the largest file in a directory.

```
find /path/to/dir -type f -exec ls -s {} + | sort -n | tail -1
```

### 39. How do you split a string in Bash?

```
IFS="," read -ra arr <<< "a,b,c"
echo "${arr[0]}"
```

### 40. How do you handle script arguments in a professional way?

- Use getopts:

```
while getopts "a:b:" opt; do
    case $opt in
        a) echo "Option A: $OPTARG" ;;
        b) echo "Option B: $OPTARG" ;;
    esac
done
```

## Advanced Challenges

41. **Write a script to calculate the uptime of a system.**

```
uptime -p
```

42. **How do you implement a progress bar in Bash?**

```
for i in {1..10}; do
    echo -n "#"
    sleep 1
done
```

43. **Write a script to extract a specific column from a file.**

```
awk '{print $2}' file.txt
```

44. **How do you sort a file in reverse order?**

```
sort -r file.txt
```

45. **How do you check open ports in a script?**

```
netstat -tuln
```

46. **Write a script to ping a list of servers.**

```
for server in server1 server2; do
    ping -c 1 "$server" && echo "$server is reachable"
done
```

47. **How do you check if a variable is empty?**

```
if [ -z "$var" ]; then
    echo "Empty"
fi
```

48. **How do you count the number of files in a directory?**

```
ls -1 | wc -l
```

49. **How do you print the 10th line of a file?**

```
sed -n '10p' filename
```

50. **Write a script to find the IP address of the system.**

```
ip addr show | grep "inet " | awk '{print $2}' | cut -d/ -f1
```