Linux Production Shell Scripts

1. File Backup Script:

```
#!/bin/bash

backup_dir="/path/to/backup"
source_dir="/path/to/source"

# Create a timestamped backup of the source directory
tar -czf "$backup_dir/backup_$(date +%Y%m%d_%H%M%S).tar.gz"
"$source_dir"
```

2. System Monitoring Script:

```
#!/bin/bash

threshold=90

# Monitor CPU usage and trigger alert if threshold exceeded
cpu_usage=$(top -bn1 | grep "Cpu(s)" | awk '{print $2}' | cut -d. -f1)
if [ "$cpu_usage" -gt "$threshold" ]; then
    echo "High CPU usage detected: $cpu_usage%"
    # Add alert/notification logic here
fi
```

3. User Account Management Script:

```
#!/bin/bash

username="newuser"

# Check if user exists; if not, create new user
if id "$username" &>/dev/null; then
    echo "User $username already exists."
else
    useradd -m "$username"
    echo "User $username created."
```

```
Fi
```

4. Log Analyzer Script:

```
#!/bin/bash

logfile="/path/to/logfile.log"

# Extract lines with "ERROR" from the log file
grep "ERROR" "$logfile" > error_log.txt
echo "Error log created."
```

5. Password Generator Script:

```
#!/bin/bash

length=12

# Generate a random password
password=$(openssl rand -base64 $length)
echo "Generated password: $password"
```

6. File Encryption/Decryption Script:

```
#!/bin/bash

file="/path/to/file.txt"

# Encrypt file using AES-256-CBC
openssl enc -aes-256-cbc -salt -in "$file" -out "$file.enc"
echo "File encrypted: $file.enc"
```

7. Automated Software Installation Script:

```
#!/bin/bash

packages=("package1" "package2" "package3")
```

```
# Install listed packages using apt-get
for package in "${packages[@]}"; do
    sudo apt-get install "$package" -y
done

echo "Packages installed successfully."
```

8. Network Connectivity Checker Script:

```
#!/bin/bash

host="example.com"

# Check network connectivity by pinging a host
if ping -c 1 "$host" &>/dev/null; then
    echo "Network is up."
else
    echo "Network is down."
fi
```

9. Website Uptime Checker Script:

```
#!/bin/bash

website="https://example.com"

# Check if website is accessible
if curl --output /dev/null --silent --head --fail "$website"; then
    echo "Website is up."
else
    echo "Website is down."
fi
```

10. Data Cleanup Script:

```
#!/bin/bash

directory="/path/to/cleanup"
```

```
# Remove files older than 7 days in specified directory
find "$directory" -type f -mtime +7 -exec rm {} \;
echo "Old files removed."
```

11. CPU Usage Tracker Script:

```
#!/bin/bash

output_file="cpu_usage_log.txt"

# Log current CPU usage to a file with timestamp
echo "$(date) $(top -bn1 | grep 'Cpu(s)' | awk '{print $2}' | cut -d.
-f1)%" >> "$output_file"
echo "CPU usage logged."
```

12. System Information Script:

```
#!/bin/bash

output_file="system_info.txt"

# Gather system information and save to a file
echo "System Information:" > "$output_file"
echo "-------------------" >> "$output_file"
echo "Hostname: $(hostname)" >> "$output_file"
echo "OS: $(uname -a)" >> "$output_file"
echo "Memory: $(free -h)" >> "$output_file"
echo "Disk Space: $(df -h)" >> "$output_file"
echo "System info saved to $output_file."
```

13. Task Scheduler Script:

```
#!/bin/bash

scheduled_task="/path/to/your_script.sh"
schedule_time="0 2 * * *"
```

```
# Schedule a task using cron
echo "$schedule_time $scheduled_task" | crontab -
echo "Task scheduled successfully."
```

14. Disk Space Monitoring Script:

```
#!/bin/bash

threshold=90

# Monitor disk usage and trigger alert if threshold exceeded
disk_usage=$(df -h | grep "/dev/sda1" | awk '{print $5}' | cut -d%
-f1)
if [ "$disk_usage" -gt "$threshold" ]; then
    echo "High disk usage detected: $disk_usage%"
    # Add alert/notification logic here
fi
```

15. Remote Server Backup Script:

```
#!/bin/bash

source_dir="/path/to/source"
remote_server="user@remoteserver:/path/to/backup"

# Backup files/directories to a remote server using rsync
rsync -avz "$source_dir" "$remote_server"
echo "Files backed up to remote server."
```

16. Environment Setup Script:

```
#!/bin/bash

# Customize for your specific environment setup
echo "Setting up development environment..."
# Install necessary packages, configure settings, etc.
echo "Development environment set up successfully."
```

17. File Compression/Decompression Script:

```
#!/bin/bash

file_to_compress="/path/to/file.txt"

# Compress a file using gzip
gzip "$file_to_compress"
echo "File compressed: $file_to_compress.gz"
```

18. Database Backup Script:

```
#!/bin/bash

database_name="your_database"
output_file="database_backup_$(date +%Y%m%d).sql"

# Perform database backup using mysqldump
mysqldump -u username -ppassword "$database_name" > "$output_file"
echo "Database backup created: $output_file"
```

19. Git Repository Updater Script:

```
#!/bin/bash

git_repo="/path/to/your/repo"

# Update a Git repository
cd "$git_repo"
git pull origin master
echo "Git repository updated."
```

20. Directory Synchronization Script:

```
#!/bin/bash

source_dir="/path/to/source"
destination_dir="/path/to/destination"
```

```bash
# Synchronize directories using rsync
rsync -avz "$source_dir" "$destination_dir"
echo "Directories synchronized successfully."
```

21. Web Server Log Analyzer Script:

```bash
#!/bin/bash

log_file="/var/log/apache2/access.log"

# Analyze web server log to count unique IP addresses
awk '{print $1}' "$log_file" | sort | uniq -c | sort -nr
echo "Web server log analyzed."
```

22. System Health Check Script:

```bash
#!/bin/bash

output_file="system_health_check.txt"

# Perform system health check and save results to a file
echo "System Health Check:" > "$output_file"
echo "---------------------" >> "$output_file"
echo "Uptime: $(uptime)" >> "$output_file"
echo "Load Average: $(cat /proc/loadavg)" >> "$output_file"
echo "Memory Usage: $(free -m)" >> "$output_file"
echo "System health check results saved to $output_file."
```

23. Automated Database Cleanup Script:

```bash
#!/bin/bash

database_name="your_database"
days_to_keep=7

# Clean up old database backups older than specified days
find /path/to/database/backups -name "$database_name*.sql" -mtime
+"$days_to_keep" -exec rm {} \;
echo "Old database backups cleaned up."
```

24. User Password Expiry Checker Script:

```
#!/bin/bash

# Check password expiry for users with bash shell
IFS=$'\n'
for user in $(cat /etc/passwd | grep "/bin/bash" | cut -d: -f1); do
    password_expires=$(chage -l "$user" | grep "Password expires" |
awk '{print $4}')
    echo "User: $user, Password Expires: $password_expires"
done
unset IFS
```

25. Service Restart Script:

```
#!/bin/bash

service_name="your_service"

# Restart a specified service
sudo systemctl restart "$service_name"
echo "Service $service_name restarted."
```

26. Folder Size Checker Script:

```
#!/bin/bash

folder_path="/path/to/folder"

# Check and display the size of a specified folder
du -sh "$folder_path"
echo "Folder size checked."
```

27. Backup Rotation Script:

```
#!/bin/bash

backup_dir="/path/to/backups"
```

```
max_backups=5

# Rotate backups by deleting the oldest if more than max_backups
while [ $(ls -1 "$backup_dir" | wc -l) -gt "$max_backups" ]; do
    oldest_backup=$(ls -1t "$backup_dir" | tail -n 1)
    rm -r "$backup_dir/$oldest_backup"
done
echo "Backup rotation completed."
```

28. Remote Script Execution Script:

```
#!/bin/bash

remote_server="user@remote-server"
remote_script="/path/to/remote/script.sh"

# Execute a script on a remote server via SSH
ssh "$remote_server" "bash -s" < "$remote_script"
echo "Remote script executed."
```

29. Network Interface Information Script:

```
#!/bin/bash

network_interface="eth0"

# Display network interface information
ifconfig "$network_interface"
echo "Network interface information displayed."
```

30. Random Quotes Generator Script:

```
#!/bin/bash

quotes=("Quote 1" "Quote 2" "Quote 3" "Quote 4")

# Generate and display a random quote from the array
random_index=$((RANDOM % ${#quotes[@]}))
echo "Random Quote: ${quotes[$random_index]}"
```

Write a shell scripting for below Questions

1)To list down which services are running in my system

list_services.sh

```
#!/bin/bash

echo "Listing all running services on Linux system using systemctl:"

echo "--------------------------------------------------------"

# Check if systemctl is available

if command -v systemctl &> /dev/null; then

   # List running services

   systemctl list-units --type=service --state=running

else

   echo "systemctl is not available on this system."

fi
```

Make it executable

```
chmod +x list_services.sh
```

Run the script with the process name as an argument

```
./kill_process.sh list_services.sh
```

2)Need to kill one process which is running in my system

kill_process.sh

```
#!/bin/bash

# Check if the process name was provided as an argument

if [ $# -eq 0 ]; then

   echo "Usage: $0 <process_name>"

   exit 1

fi

# Get the process name from the argument

process_name=$1

# Find the process ID (PID) of the process

pid=$(pgrep -f "$process_name")

# Check if the process is running

if [ -z "$pid" ]; then
```

```
    echo "Process '$process_name' not found."
    exit 1
fi
# Kill the process
kill $pid
# Check if the kill command was successful
if [ $? -eq 0 ]; then
    echo "Process '$process_name' with PID $pid has been killed."
else
    echo "Failed to kill process '$process_name'."
    exit 1
fi
```

Make it executable

```
chmod +x kill_process.sh
```

Run the script with the process name as an argument

```
./kill_process.sh process_name
```

3)Need to get the disk space and memory space of the system

system_info.sh

```
#!/bin/bash
# Get disk space
echo "Disk Space:"
df -h
# Get memory space
echo "Memory Space:"
free -h
```

To run this script:

1. **Make the script executable**: chmod +x system_info.sh.
2. **Run the script**: ./system_info.sh.

4)List down software's which are installed in my system

Ubuntu:

```
list_installed_software.sh

#!/bin/bash

dpkg --get-selections
```

Make it executable

chmod +x list_installed_software.sh

Run the script

./list_installed_software.sh

CentOS:

```
list_installed_software.sh

#!/bin/bash

rpm -qa
```

Make it executable

chmod +x list_installed_software.sh

Run the script

./list_installed_software.sh

5)To get the service name and stop and start the service like (HTTPD & Nginx & Apache & Docker)

manage_service.sh

```
#!/bin/bash

# Function to check the status of a service

check_status() {

   sudo systemctl is-active --quiet $1 && echo "$1 is running" || echo "$1 is not running"

}

# Function to start a service

start_service() {

   sudo systemctl start $1

   echo "$1 started"

}

# Function to stop a service
```

```
stop_service() {
   sudo systemctl stop $1
   echo "$1 stopped"
}
# Check if the user provided enough arguments
if [ $# -lt 2 ]; then
   echo "Usage: $0 {start|stop|status} {httpd|nginx|apache2|docker}"
   exit 1
fi
# Assign arguments to variables
ACTION=$1
SERVICE=$2
# Perform the action based on the user input
case $ACTION in
   start)
      start_service $SERVICE
      ;;
   stop)
      stop_service $SERVICE
      ;;
   status)
      check_status $SERVICE
      ;;
   *)
      echo "Invalid action. Usage: $0 {start|stop|status} {httpd|nginx|apache2|docker}"
      exit 1
      ;;
esac
```

Make the script executable

```
chmod +x manage_service.sh
```

Run the script

```
./manage_service.sh start nginx

./manage_service.sh stop apache2

./manage_service.sh status docker
```

6)To list down the agent and if agent is stopped state, then start the service and check for every time if its stop script must start the service

manage_agents.sh

```
#!/bin/bash

# Function to start the agent service

start_service() {

  local service_name=$1

  echo "Starting $service_name..."

  sudo systemctl start $service_name

  if [ $? -eq 0 ]; then

    echo "$service_name started successfully."

  else

    echo "Failed to start $service_name."

  fi

}

# Function to check the status of the agent service

check_and_start_service() {

  local service_name=$1

  status=$(sudo systemctl is-active $service_name)

  if [ "$status" == "inactive" ] || [ "$status" == "failed" ]; then

    echo "$service_name is in $status state."

    start_service $service_name

  else

    echo "$service_name is running."

  fi

}

# List of agent services
```

```
agent_services=("agent1" "agent2" "agent3")  # Replace with actual agent service names
# Iterate through each agent service and check its status
for service in "${agent_services[@]}"; do
   check_and_start_service $service
done
```

Make the script executable

```
chmod +x manage_agents.sh
```

Run the script

```
./manage_agents.sh
```

Running the Script Periodically

```
crontab -e
```

To run the script every 5 minutes

```
*/5 * * * * /path/to/manage_agents.sh
```

7)To check the password expiry of the list of created users and if the password is expiring in 3 days, then update the password age for next 15 days

check_password_expiry.sh

```
#!/bin/bash
# List of users to check
users=("user1" "user2" "user3")
# Function to update password age
update_password_age() {
  local user=$1
  echo "Updating password age for user: $user"
  # Set password to expire in 15 days from now
  chage -d $(date +%Y-%m-%d) -M 15 $user
}
# Get the current date in seconds
current_date=$(date +%s)
# Loop through each user
for user in "${users[@]}"; do
  # Get the password expiry date
  expiry_date=$(chage -l $user | grep "Password expires" | cut -d: -f2 | xargs -I{} date -d {}
+%s)

  # Calculate the number of days until expiry

  days_until_expiry=$(( (expiry_date - current_date) / 86400 ))

  # Check if the password expires in 3 days or less

  if [ $days_until_expiry -le 3 ]; then

    echo "Password for user $user is expiring in $days_until_expiry days."

    update_password_age $user

  else

    echo "Password for user $user is not expiring soon."

  fi

done
```

Make the script executable: chmod +x check_password_expiry.sh.

Run the script: ./check_password_expiry.sh.

8)To check the particular mount point if it's reached the 70% utilization then move zip the file which is older than 7 days and move those files into /tmp/ directory.

script.sh

```
#!/bin/bash

# Variables

MOUNT_POINT="/your/mount/point"  # Replace with your actual mount point

TARGET_DIR="/your/target/directory"  # Directory to check for old files

TMP_DIR="/tmp"

# Check disk usage

usage=$(df -h | grep "$MOUNT_POINT" | awk '{print $5}' | sed 's/%//g')

# Check if usage is greater than or equal to 70%

if [ "$usage" -ge 70 ]; then

    echo "Disk usage at $MOUNT_POINT is $usage%, which is above the threshold."

    # Find files older than 7 days and zip them

    find "$TARGET_DIR" -type f -mtime +7 -print0 | while IFS= read -r -d '' file; do

        zip_file="${file}.zip"

        zip "$zip_file" "$file"
        mv "$zip_file" "$TMP_DIR/"

    done

else

    echo "Disk usage at $MOUNT_POINT is $usage%, which is below the threshold."

fi
```

Make the script executable: chmod +x script.sh.

Run the script: ./ script.sh.

9)If the particular mount point is reached the 70 % then delete the older files starting 7 days of files

cleanup.sh

```
#!/bin/bash

# Mount point to check

MOUNT_POINT="/path/to/mount"

# Threshold percentage (70%)

THRESHOLD=70

# Directory to clean up

DIR_TO_CLEAN="/path/to/directory"

# Check the disk usage

USAGE=$(df -h "$MOUNT_POINT" | grep -vE '^Filesystem|tmpfs|cdrom' | awk '{ print $5 }' |
sed 's/%//g')

# If usage is greater than or equal to the threshold

if [ "$USAGE" -ge "$THRESHOLD" ]; then

  echo "Disk usage is $USAGE%, which is greater than or equal to the threshold of
$THRESHOLD%."

  echo "Deleting files older than 7 days in $DIR_TO_CLEAN..."

  # Find and delete files older than 7 days

  find "$DIR_TO_CLEAN" -type f -mtime +7 -exec rm -f {} \;

  echo "Old files deleted."

else

  echo "Disk usage is $USAGE%, which is below the threshold of $THRESHOLD%."

fi
```

Make the script executable: chmod +x cleanup.sh

Run the script: ./ cleanup.sh

10)Every day in the morning at 9am IST and Evening 7 PM IST I need to check the disk space and free memory and need to run the script and make a cron job to it and store the output in /tmp directory as diskspace.txt and process.txt

check_system.sh

```
#!/bin/bash

# Define the output files

DISKSPACE_FILE="/tmp/diskspace.txt"

PROCESS_FILE="/tmp/process.txt"

# Get disk space usage and free memory

df -h > "$DISKSPACE_FILE"

free -h > "$PROCESS_FILE"
```

Make the script executable: chmod +x check_system.sh

Run the script: ./ check_system.sh

Set Up the Cron Job:

crontab -e

To run the script at 9 AM and 7 PM IST

0 3 * * * /path/to/check_system.sh

0 13 * * * /path/to/check_system.sh

SHELL Scripting Interview Questions and Answers

1. What is a shell script?

Question: What is a shell script?

Answer: A shell script is a text file that contains a sequence of commands for a UNIX-based operating system's shell to execute. Shell scripts are used to automate repetitive tasks, manage system operations, and simplify complex commands. They can contain standard UNIX commands, conditional statements, loops, and functions.

2. How do you create and execute a shell script?

Question: How do you create and execute a shell script?

Answer:

1. **Create a Shell Script:**
   - Open a text editor and write your script.
   - Save the file with a `.sh` extension (e.g., `myscript.sh`).

   Example script:

   ```
   #!/bin/bash
   echo "Hello, World!"
   ```

2. **Make the Script Executable:**

   ```
   chmod +x myscript.sh
   ```

3. **Execute the Script:**

   ```
   ./myscript.sh
   ```

3. What is the significance of `#!/bin/bash` at the beginning of a script?

Question: What is the significance of `#!/bin/bash` at the beginning of a script?

Answer: The `#!/bin/bash` line at the beginning of a shell script is called a shebang or hashbang. It specifies the path to the interpreter that should be used to execute the script. In this case, it indicates that the script should be run using the Bash shell. It ensures that the script runs with the correct interpreter, regardless of the user's default shell.

4. How do you define and use variables in a shell script?

Question: How do you define and use variables in a shell script?

Answer:

- **Define a Variable:**

  ```
  my_variable="Hello"
  ```

- **Use a Variable:**

  ```
  echo $my_variable
  ```

- Example script:

  ```
  #!/bin/bash
  greeting="Hello, World!"
  echo $greeting
  ```

5. How do you write comments in a shell script?

Question: How do you write comments in a shell script?

Answer: Comments in a shell script are written using the `#` symbol. Anything following `#` on a line is treated as a comment and ignored by the shell.

Example:

```
#!/bin/bash
# This is a comment
echo "Hello, World!" # This is also a comment
```

6. What are positional parameters in shell scripting?

Question: What are positional parameters in shell scripting?

Answer: Positional parameters are variables that hold the arguments passed to a script or function. They are referenced using `$1`, `$2`, `$3`, etc., where `$1` is the first argument, `$2` is the second, and so on. `$0` refers to the script's name.

Example script:

```
#!/bin/bash
echo "First argument: $1"
echo "Second argument: $2"
```

If the script is executed with `./myscript.sh arg1 arg2`, it will output:

```
First argument: arg1
Second argument: arg2
```

7. How do you use conditional statements in a shell script?

Question: How do you use conditional statements in a shell script?

Answer: Conditional statements in shell scripts are used to perform different actions based on conditions. The `if` statement is commonly used for this purpose.

Example:

```
#!/bin/bash
number=5
if [ $number -gt 3 ]; then
  echo "The number is greater than 3"
else
  echo "The number is not greater than 3"
fi
```

8. How do you loop through a list of items in a shell script?

Question: How do you loop through a list of items in a shell script?

Answer: The `for` loop is used to iterate through a list of items in a shell script.

Example:

```
#!/bin/bash
for item in apple banana cherry; then
  echo "Fruit: $item"
done
```

9. How do you read input from the user in a shell script?

Question: How do you read input from the user in a shell script?

Answer: You can use the `read` command to read input from the user.

Example:

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

10. How do you debug a shell script?

Question: How do you debug a shell script?

Answer: You can debug a shell script using the `-x` option with `bash` to execute the script in debug mode.

Example:

```
bash -x myscript.sh
```

This will print each command and its arguments as they are executed, helping you to identify where the script is failing or producing unexpected results.

1. What is the difference between `[` and `[[` in shell scripting?

Question: What is the difference between `[` and `[[` in shell scripting?

Answer: `[` is a synonym for the `test` command and is POSIX compliant. `[[` is a keyword and provides more features, such as:

- Enhanced pattern matching.
- Logical operators (`&&`, `||`) without the need for `-a` or `-o`.
- Safer handling of complex expressions (e.g., regex).

Example:

```
#!/bin/bash
# Using [
if [ "$a" == "$b" ]; then
  echo "Equal"
fi

# Using [[
if [[ "$a" == "$b" ]]; then
  echo "Equal"
fi
```

2. How do you handle errors in a shell script?

Question: How do you handle errors in a shell script?

Answer: Use exit codes, `trap` command, and error checking after each command.

Example:

```
#!/bin/bash

# Exit on any error
set -e

# Function to handle errors
error_handler() {
  echo "Error occurred in script at line: $1"
  exit 1
}

# Trap errors
trap 'error_handler $LINENO' ERR

# Commands
```

```
command1
command2
```

3. How can you pass an array to a function in shell scripting?

Question: How can you pass an array to a function in shell scripting?

Answer: Use `local` and `eval` to handle array parameters.

Example:

```
#!/bin/bash

# Function to print an array
print_array() {
  eval "local arr=(\"\${$1[@]}\")"
  for element in "${arr[@]}"; do
    echo $element
  done
}

# Main script
my_array=("one" "two" "three")
print_array my_array[@]
```

4. How do you check if a process is running in shell scripting?

Question: How do you check if a process is running in shell scripting?

Answer: Use `pgrep` or `ps` to check for a running process.

Example:

```
#!/bin/bash

process_name="my_process"

if pgrep "$process_name" > /dev/null; then
  echo "$process_name is running"
else
  echo "$process_name is not running"
fi
```

5. How do you handle command substitution in shell scripting?

Question: How do you handle command substitution in shell scripting?

Answer: Use backticks (``) or `$()` for command substitution.

Example:

```
#!/bin/bash

# Using backticks
current_date=`date`

# Using $()
current_date=$(date)

echo "Current date and time: $current_date"
```

6. How do you create and use a Here Document in a shell script?

Question: How do you create and use a Here Document in a shell script?

Answer: A Here Document allows you to pass a block of text to a command.

Example:

```
#!/bin/bash

cat <<EOF
This is a Here Document
It allows multi-line strings
EOF
```

7. Explain the usage of `trap` command with an example.

Question: Explain the usage of `trap` command with an example.

Answer: The `trap` command is used to specify commands to be executed when the shell receives a signal.

Example:

```
#!/bin/bash

# Function to execute on exit
cleanup() {
  echo "Cleaning up..."
  rm -f /tmp/mytempfile
}

# Trap EXIT signal
trap cleanup EXIT

# Commands
echo "Script is running"
touch /tmp/mytempfile
```

8. How do you implement logging in a shell script?

Question: How do you implement logging in a shell script?

Answer: Redirect output to a log file and use `exec` to direct stdout and stderr.

Example:

```
#!/bin/bash

log_file="script.log"

# Redirect stdout and stderr
exec > >(tee -a $log_file) 2>&1

# Commands
echo "This is a log entry"
```

9. How do you perform arithmetic operations in a shell script?

Question: How do you perform arithmetic operations in a shell script?

Answer: Use `$(( ... ))` for arithmetic operations.

Example:

```
#!/bin/bash

a=10
b=5

sum=$((a + b))
difference=$((a - b))
product=$((a * b))
quotient=$((a / b))

echo "Sum: $sum"
echo "Difference: $difference"
echo "Product: $product"
echo "Quotient: $quotient"
```

10. How do you schedule a shell script to run at a specific time?

Question: How do you schedule a shell script to run at a specific time?

Answer: Use `cron` for scheduling scripts.

1. **Edit the crontab:**

   ```
   crontab -e
   ```

2. **Add a cron job:**

```
      0 2 * * * /path/to/script.sh
```

This schedules `script.sh` to run every day at 2:00 AM.

11. What is the use of `set -e`, `set -u`, `set -o pipefail`?

Question: What is the use of `set -e`, `set -u`, `set -o pipefail`?

Answer:

- `set -e`: Exit immediately if a command exits with a non-zero status.
- `set -u`: Treat unset variables as an error and exit immediately.
- `set -o pipefail`: Return the exit status of the last command in the pipeline that failed.

Example:

```
#!/bin/bash
set -euo pipefail

command1
command2
```

12. How do you manage background processes in shell scripting?

Question: How do you manage background processes in shell scripting?

Answer: Use `&` to run a command in the background and `wait` to wait for background processes to finish.

Example:

```
#!/bin/bash

# Run in background
command1 &
pid1=$!

command2 &
pid2=$!

# Wait for both processes
wait $pid1
wait $pid2
```

13. How do you handle functions in a shell script?

Question: How do you handle functions in a shell script?

Answer: Define and call functions using the `function` keyword or directly with the function name.

Example:

```
#!/bin/bash

# Function definition
my_function() {
  echo "Hello from my_function"
}

# Call the function
my_function
```

14. How do you parse command-line arguments in a shell script?

Question: How do you parse command-line arguments in a shell script?

Answer: Use `getopts` to parse command-line options.

Example:

```
#!/bin/bash

while getopts "a:b:" opt; do
  case $opt in
    a) param_a=$OPTARG ;;
    b) param_b=$OPTARG ;;
    *) echo "Invalid option" ;;
  esac
done

echo "Param A: $param_a"
echo "Param B: $param_b"
```

15. How do you perform string manipulation in a shell script?

Question: How do you perform string manipulation in a shell script?

Answer:

- **Extract substring:**

  ```
  string="Hello, World!"
  substring=${string:7:5}  # Output: World
  ```

- **String length:**

  ```
  length=${#string}  # Output: 13
  ```

- **Replace substring:**

  ```
  new_string=${string/World/Universe}  # Output: Hello,
  Universe!
  ```

1. Scenario: Monitoring Disk Usage

Question: How would you write a shell script to monitor disk usage and send an email alert if usage exceeds 90%?

Answer:

```
#!/bin/bash

THRESHOLD=90
EMAIL="admin@example.com"

df -h | grep -vE '^Filesystem|tmpfs|cdrom' | awk '{ print $5 "
" $1 }' | while read output; do
  usage=$(echo $output | awk '{ print $1}' | sed 's/%//g')
  partition=$(echo $output | awk '{ print $2 }')
  if [ $usage -ge $THRESHOLD ]; then
    echo "Running out of space \"$partition ($usage%)\"" |
mail -s "Disk Space Alert: $partition ($usage%)" $EMAIL
  fi
done
```

Explanation:

- `df -h`: Get disk usage in human-readable format.
- `grep -vE '^Filesystem|tmpfs|cdrom'`: Exclude certain filesystems.
- `awk '{ print $5 " " $1 }'`: Print usage and partition.
- Loop through each line and check if usage exceeds threshold, then send an email.

2. Scenario: Backup and Clean-up Logs

Question: Write a shell script to backup log files from `/var/log` to `/backup/logs` and delete log files older than 7 days.

Answer:

```
#!/bin/bash

SOURCE_DIR="/var/log"
BACKUP_DIR="/backup/logs"
DAYS=7

# Create backup directory if it doesn't exist
mkdir -p $BACKUP_DIR
```

```
# Backup log files
cp $SOURCE_DIR/*.log $BACKUP_DIR/

# Delete log files older than 7 days
find $SOURCE_DIR/*.log -type f -mtime +$DAYS -exec rm {} \;
```

Explanation:

- `mkdir -p $BACKUP_DIR`: Create backup directory if not exists.
- `cp $SOURCE_DIR/*.log $BACKUP_DIR/`: Copy log files to backup directory.
- `find $SOURCE_DIR/*.log -type f -mtime +$DAYS -exec rm {} \;`: Delete files older than 7 days.

3. Scenario: Checking Service Status

Question: How would you write a script to check if a service (e.g., `apache2`) is running and restart it if it's not?

Answer:

```
#!/bin/bash

SERVICE="apache2"

if systemctl is-active --quiet $SERVICE; then
  echo "$SERVICE is running"
else
  echo "$SERVICE is not running, restarting..."
  systemctl start $SERVICE
  if systemctl is-active --quiet $SERVICE; then
    echo "$SERVICE restarted successfully"
  else
    echo "Failed to restart $SERVICE"
  fi
fi
```

Explanation:

- `systemctl is-active --quiet $SERVICE`: Check if service is running.
- If not running, restart the service and check status again.

4. Scenario: User Account Management

Question: Write a shell script to create a user account, set a password, and ensure the user's home directory is created.

Answer:

```
#!/bin/bash
```

```
USERNAME=$1
PASSWORD=$2

if [ -z "$USERNAME" ] || [ -z "$PASSWORD" ]; then
  echo "Usage: $0 <username> <password>"
  exit 1
fi

# Create user account
useradd -m -s /bin/bash $USERNAME

# Set user password
echo "$USERNAME:$PASSWORD" | chpasswd

# Check if user was created successfully
if id "$USERNAME" &>/dev/null; then
  echo "User $USERNAME created successfully"
else
  echo "Failed to create user $USERNAME"
fi
```

Explanation:

- `$1` and `$2` for username and password.
- `useradd -m -s /bin/bash $USERNAME`: Create user with home directory and bash shell.
- `echo "$USERNAME:$PASSWORD" | chpasswd`: Set user password.

5. Scenario: File Synchronization

Question: Write a script to synchronize files between two directories (`/source` and `/destination`) and log the synchronization process.

Answer:

```
#!/bin/bash

SOURCE_DIR="/source"
DEST_DIR="/destination"
LOG_FILE="/var/log/sync.log"

# Synchronize files
rsync -av --delete $SOURCE_DIR/ $DEST_DIR/ > $LOG_FILE 2>&1

# Check if rsync was successful
if [ $? -eq 0 ]; then
  echo "Synchronization completed successfully" >> $LOG_FILE
else
  echo "Synchronization failed" >> $LOG_FILE
fi
```

Explanation:

- `rsync -av --delete $SOURCE_DIR/ $DEST_DIR/`: Synchronize directories.
- Log the output and check rsync exit status.

6. Scenario: Parsing a Configuration File

Question: Write a script to parse a configuration file (`config.cfg`) and print each key-value pair.

Answer:

```
#!/bin/bash

CONFIG_FILE="config.cfg"

if [ ! -f $CONFIG_FILE ]; then
  echo "Configuration file not found!"
  exit 1
fi

while IFS='=' read -r key value; do
  # Skip comments and empty lines
  if [[ "$key" =~ ^#.* ]] || [ -z "$key" ]; then
    continue
  fi
  echo "Key: $key, Value: $value"
done < $CONFIG_FILE
```

Explanation:

- `IFS='=' read -r key value`: Split each line by =.
- Skip comments and empty lines, then print key-value pairs.

7. Scenario: Archive Old Files

Question: Write a script to archive files in `/data` older than 30 days into `/archive`.

Answer:

```
#!/bin/bash

SOURCE_DIR="/data"
ARCHIVE_DIR="/archive"
DAYS=30

# Create archive directory if it doesn't exist
mkdir -p $ARCHIVE_DIR
```

```
# Find and archive files older than 30 days
find $SOURCE_DIR -type f -mtime +$DAYS -exec mv {}
$ARCHIVE_DIR/ \;

# Verify the files have been moved
if [ $? -eq 0 ]; then
  echo "Files older than $DAYS days have been archived."
else
  echo "Failed to archive files."
fi
```

Explanation:

- `mkdir -p $ARCHIVE_DIR`: Create archive directory if not exists.
- `find $SOURCE_DIR -type f -mtime +$DAYS -exec mv {} $ARCHIVE_DIR/ \;`: Move files older than 30 days to archive directory.

8. Scenario: Automatic Database Backup

Question: Write a script to backup a MySQL database and delete backups older than 7 days.

Answer:

```
#!/bin/bash

DB_NAME="mydatabase"
DB_USER="dbuser"
DB_PASS="dbpassword"
BACKUP_DIR="/backup/db"
DAYS=7

# Create backup directory if it doesn't exist
mkdir -p $BACKUP_DIR

# Backup database
backup_file="$BACKUP_DIR/$DB_NAME-$(date +%F).sql"
mysqldump -u $DB_USER -p$DB_PASS $DB_NAME > $backup_file

# Check if backup was successful
if [ $? -eq 0 ]; then
  echo "Database backup successful: $backup_file"
else
  echo "Database backup failed!"
  exit 1
fi

# Delete backups older than 7 days
find $BACKUP_DIR -type f -mtime +$DAYS -exec rm {} \;
```

Explanation:

- `mkdir -p $BACKUP_DIR`: Create backup directory if not exists.
- `mysqldump`: Backup the database.
- `find $BACKUP_DIR -type f -mtime +$DAYS -exec rm {} \;`: Delete backups older than 7 days.

Top Shell Scripts Asked in Interviews

Backup Script

Task: Write a script to back up a directory (e.g., /var/log/) to another location, compress it, and add a timestamp to the filename.

```
#!/bin/bash

src_dir="/var/log"

backup_dir="/backup/logs"

filename="log_backup_$(date +%Y%m%d_%H%M%S).tar.gz"


tar -czf $backup_dir/$filename $src_dir

echo "Backup completed: $filename"
```

Disk Usage Alert Script

Task: Monitor disk usage and send an alert if usage exceeds a defined threshold.

```
#!/bin/bash

threshold=80

usage=$(df / | grep / | awk '{ print $5 }' | sed 's/%//g')


if [ $usage -gt $threshold ]; then

  echo "Disk usage at $usage%, sending alert!"

  # Uncomment below to send email (requires mail setup)

  # echo "Disk usage alert" | mail -s "Disk Usage Warning" admin@example.com

fi
```

Service Monitoring Script

Task: Check if a service (e.g., Apache) is running, and restart it if it is down.

```
#!/bin/bash

service="apache2"



if ! systemctl is-active --quiet $service; then

  echo "$service is down, restarting..."

  systemctl start $service

else

  echo "$service is running."

fi
```

User Creation Script

Task: Create users from a file containing usernames.

```
#!/bin/bash

input="users.txt"

while IFS= read -r user

do

  useradd -m $user

  echo "User $user created."

done < "$input"
```

Log Rotation Script

Task: Rotate logs by compressing logs older than 7 days.

```
#!/bin/bash

log_dir="/var/log/myapp"

find $log_dir -type f -mtime +7 -exec tar -czf {}.tar.gz {} \; -exec rm {} \;

echo "Logs older than 7 days have been archived."
```

File Archiving Script

Task: Archive files older than 7 days.

```
#!/bin/bash

src_dir="/data/files"

archive_dir="/archive"



find $src_dir -type f -mtime +7 -exec mv {} $archive_dir \;

echo "Archived files older than 7 days."
```

Database Backup Script

Task: Back up a MySQL database.

```
#!/bin/bash

db_name="mydatabase"

backup_dir="/backup/db"

filename="db_backup_$(date +%Y%m%d_%H%M%S).sql.gz"



mysqldump -u root -p $db_name | gzip > $backup_dir/$filename

echo "Database backup completed: $filename"
```

Log File Parsing Script

Task: Extract specific keywords from a log file.

```
#!/bin/bash

logfile="/var/log/syslog"

keyword="ERROR"



grep $keyword $logfile > error_log.txt

echo "All ERROR entries saved to error_log.txt."
```

File Processing Script (Processing 100 lines at a time)

Task: Process a large file in chunks.

```
#!/bin/bash

filename="largefile.txt"

chunk_size=100

split -l $chunk_size $filename part_



for file in part_*

do

  echo "Processing $file..."

  # Do some processing here

  rm $file

done
```

Automated Remote File Transfer Script

Task: Automate the transfer of files from a local machine to a remote server.

```
#!/bin/bash

src_dir="/local/path"

dest_user="user"

dest_server="server.com"

dest_dir="/remote/path"



scp -r $src_dir $dest_user@$dest_server:$dest_dir

echo "Files transferred to $dest_server."
```