Shell Scripting Step by Step

Contents

Introduction

Shell scripting allows users to automate tasks in Linux by writing scripts that execute a sequence of commands. This tutorial covers basic to intermediate shell scripting using commonly used Linux commands.

---

1. Creating a Basic Shell Script

#!/bin/bash

echo "Hello, World!"

Save this as script.sh and give it execute permissions:

chmod +x script.sh

./script.sh

---

2. Variables and User Input

#!/bin/bash

echo "Enter your name:"

read name

echo "Hello, $name!"

---

3. Using Conditional Statements

#!/bin/bash

if [-f "file.txt"]; then

```
  echo "file.txt exists"
else
  echo "file.txt does not exist"
fi
```

---

4. Loops in Shell Scripts

```
#!/bin/bash
for i in {1..5}; do
  echo "Iteration $i"
done
```

---

5. Searching Within Files

Using grep to Find Words in a File

```
grep "error" logfile.txt
```

Using grep with Wildcards

```
grep "fail*" logfile.txt  # Matches 'fail', 'failure', etc.
```

Finding Lines That Match a Pattern

```
grep -E "(error|warning)" logfile.txt
```

Searching for a Word in Multiple Files

```
grep "critical" *.log
```

Using find to Locate Files Matching a Pattern

```
find /var/log -name "*.log"
```

Using awk for Advanced Search

```
awk '/error/ {print}' logfile.txt
```

---

6. Working with Functions

```
#!/bin/bash

function greet() {

   echo "Hello, $1!"

}

greet "Alice"
```

---

7. Automating Tasks with Cron Jobs

```
crontab -e

# Add the following line to run script.sh every day at 5 AM

0 5 * * * /path/to/script.sh
```

---

8. Handling Command Line Arguments

```
#!/bin/bash

echo "Script name: $0"

echo "First argument: $1"

echo "Second argument: $2"
```

Run with:

```
./script.sh arg1 arg2
```

---

9. Logging and Debugging

Redirecting Output to a Log File

./script.sh > output.log 2>&1

Debugging with set Command

```
#!/bin/bash

set -x  # Enable debugging

echo "Debugging this script"

set +x  # Disable debugging
```

---

10. File and Directory Operations

Copying Files

```
cp file1.txt file2.txt
```

Moving Files

```
mv file1.txt /home/user/
```

Deleting Files

```
rm file1.txt
```

Changing File Permissions

```
chmod 755 script.sh
```

Changing File Ownership

```
chown user:user file.txt
```

Advanced Shell Scripts

System Monitoring Script

Monitors CPU, memory, and disk usage in real time.

```bash
#!/bin/bash

while true; do
  clear
  echo "===== SYSTEM MONITORING ====="
  echo "CPU Usage:"
  mpstat 1 1 | awk '/all/ {print "CPU Usage: " 100 - $13"%"}'

  echo -e "\nMemory Usage:"
  free -h | awk 'NR==2{printf "Used: %s / Total: %s\n", $3, $2}'

  echo -e "\nDisk Usage:"
  df -h | awk '$NF=="/"{printf "Used: %d/%d GB (%s)\n", $3, $2, $5}'

  echo -e "\nTop 5 Memory Consuming Processes:"
  ps aux --sort=-%mem | awk 'NR<=6{print $2, $4, $11}'

  sleep 5
done
```

Find Processes Hogging System Resources

Finds top CPU and memory-consuming processes.

```bash
#!/bin/bash

echo "Top 5 CPU-consuming processes:"
ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%cpu | head -6

echo -e "\nTop 5 Memory-consuming processes:"
```

ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem | head -6
```

## Display Threads of a Given Process

Displays the threads running under a given process.

```bash
#!/bin/bash

if [ -z "$1" ]; then
  echo "Usage: $0 <pid>"
  exit 1
fi

echo "Threads of process $1:"
ps -T -p $1
```

Monitor Network Usage

Monitors real-time network traffic on all interfaces.

```bash
#!/bin/bash

echo "Monitoring network traffic..."
ifstat -t 1 5
```

Monitor Disk I/O Activity

Displays disk I/O statistics.

```bash
#!/bin/bash

echo "Disk I/O Statistics:"
iostat -dx 1 5
```
```