Linux and Shell Scripting

8. Linux Shell Scripting: Shell Computing

Shell scripting is a powerful way to automate tasks and enhance productivity in a Linux environment. By writing shell scripts, you can automate repetitive tasks, manage system administration, and process data efficiently. This chapter covers the basics of shell scripting, control structures, loops, functions, and practical examples to get you started with shell computing.

Introduction to Shell Scripting

A shell script is a text file containing a series of commands that the shell executes in sequence. Shell scripts are written in various shell languages, with Bash (Bourne Again Shell) being the most popular. Scripts can automate a wide range of tasks, from simple file operations to complex system administration procedures.

Writing Your First Shell Script

1. Creating a Script File:
    - Use a text editor to create a new file with a .sh extension:

bash

Copy code

nano myscript.sh

2. Adding the Shebang:
    - The first line of a shell script should specify the interpreter to use. This is done with the shebang (#!):

bash

Copy code

#!/bin/bash

3. Adding Commands:
    - Write the commands you want to execute in the script:

bash

Copy code

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

4. Making the Script Executable:
    - Change the file permissions to make the script executable:

bash

Copy code

chmod +x myscript.sh

5. Running the Script:
    - Execute the script from the terminal:

bash

Copy code

./myscript.sh

Variables and Parameters

Variables store data that can be used and manipulated within a script.

- Defining Variables:

bash

Copy code

#!/bin/bash name="John" echo "Hello, $name!"

- Reading User Input:

bash

Copy code

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

- Positional Parameters:

bash

Copy code

#!/bin/bash echo "Script name: $0" echo "First parameter: $1" echo "Second parameter: $2"

Control Structures

Control structures allow you to control the flow of execution in your scripts.

1. Conditional Statements:
    - if-else:

bash

Copy code

#!/bin/bash if ["$1" -gt 10]; then echo "The number is greater than 10." else echo "The number is 10 or less." fi

    - case:

bash

Copy code

#!/bin/bash case "$1" in start) echo "Starting the service..." ;; stop) echo "Stopping the service..." ;; *) echo "Usage: $0 {start|stop}" ;; esac

2. Loops:
    - for Loop:

bash

Copy code

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

    - while Loop:

bash

Copy code

#!/bin/bash counter=1 while [$counter -le 5]; do echo "Counter: $counter" ((counter++)) done

    - until Loop:

bash

Copy code

#!/bin/bash counter=1 until [$counter -gt 5]; do echo "Counter: $counter" ((counter++)) done

Functions

Functions are reusable blocks of code that can be called with arguments.

- Defining and Calling Functions:

bash

Copy code

#!/bin/bash greet() { echo "Hello, $1!" } greet "Alice" greet "Bob"

- Returning Values from Functions:

bash

Copy code

#!/bin/bash add() { local sum=$(($1 + $2)) echo $sum } result=$(add 3 5) echo "Sum: $result"

Practical Examples

1. Backup Script:

bash

Copy code

#!/bin/bash src="/path/to/source" dest="/path/to/destination" backup_file="backup-$(date +%Y%m%d).tar.gz" tar -czf $dest/$backup_file $src echo "Backup completed: $backup_file"

2. User Management Script:

bash

Copy code

#!/bin/bash if ["$EUID" -ne 0]; then echo "Please run as root" exit fi case "$1" in add) useradd $2 echo "User $2 added." ;; delete) userdel $2 echo "User $2 deleted." ;; *) echo "Usage: $0 {add|delete} username" ;; esac

3. System Monitoring Script:

bash

Copy code

#!/bin/bash echo "CPU Load: $(uptime | awk -F'load average: ' '{ print $2 }')" echo "Memory Usage: $(free -h | grep Mem | awk '{ print $3 "/" $2 }')" echo "Disk Usage: $(df -h / | grep / | awk '{ print $5 }')"

Conclusion

Shell scripting is an essential skill for Linux administrators and power users. By automating tasks and creating powerful scripts, you can streamline workflows, improve efficiency, and enhance system management. This chapter has provided a foundation in shell scripting, covering basic syntax, control structures, loops, functions, and practical examples. With practice, you can expand your scripting skills and tackle more complex automation challenges.