**🖥️ Shell Scripting in Linux – Detailed Notes**

**1. What is Shell Scripting?**

* A **shell script** is a text file containing a sequence of **commands** that the Linux shell can execute.
* It automates repetitive tasks, system monitoring, backups, deployments, etc.
* Common shells:
  + **Bash (Bourne Again Shell)** – most widely used.
  + **sh, ksh, zsh, csh** etc.

👉 File usually ends with .sh.

**2. Steps to Create & Run a Shell Script**

**Step 1: Create a file**

vi myscript.sh

**Step 2: Add a shebang line (tells system which shell to use)**

#!/bin/bash

**Step 3: Add commands**

#!/bin/bash

echo "Hello, $USER"

date

**Step 4: Save & give execute permission**

chmod +x myscript.sh

**Step 5: Run the script**

./myscript.sh

**3. Shell Script Basics**

**(a) Variables**

#!/bin/bash

name="Sagar"

echo "Hello $name"

**(b) User Input**

#!/bin/bash

echo "Enter your name:"

read username

echo "Welcome $username"

**(c) Command Substitution**

date=$(date)

echo "Today is $date"

**4. Conditional Statements**

**(a) If statement**

#!/bin/bash

num=10

if [ $num -gt 5 ]

then

echo "Number is greater than 5"

fi

**(b) If-else**

#!/bin/bash

echo "Enter a number:"

read n

if [ $n -gt 100 ]

then

echo "Greater than 100"

else

echo "Less than or equal to 100"

fi

**(c) If-elif-else**

#!/bin/bash

echo "Enter marks:"

read marks

if [ $marks -ge 75 ]; then

echo "Distinction"

elif [ $marks -ge 35 ]; then

echo "Pass"

else

echo "Fail"

fi

**5. Loops**

**(a) For loop**

for i in 1 2 3 4 5

do

echo "Number $i"

done

**(b) While loop**

count=1

while [ $count -le 5 ]

do

echo "Count = $count"

((count++))

done

**(c) Until loop**

n=1

until [ $n -gt 5 ]

do

echo "Value: $n"

((n++))

done

**6. Functions**

#!/bin/bash

greet() {

echo "Hello $1"

}

greet "Sagar"

**7. Case Statement**

#!/bin/bash

echo "Enter choice: start/stop/restart"

read choice

case $choice in

start) echo "Starting service..." ;;

stop) echo "Stopping service..." ;;

restart) echo "Restarting service..." ;;

\*) echo "Invalid option" ;;

esac

**8. Exit Status & Error Handling**

* Every command returns an **exit code** (0 = success, non-zero = failure).

ls /home/sgharge

echo $? # Prints exit code

* Use in script:

if [ $? -eq 0 ]; then

echo "Command successful"

else

echo "Command failed"

fi

**9. Practical Examples**

**(a) Script to Check Disk Usage**

#!/bin/bash

disk=$(df -h | grep '/dev/sda1' | awk '{print $5}' | sed 's/%//')

if [ $disk -ge 80 ]; then

echo "Disk usage is high: $disk%" | mail -s "Disk Alert" admin@example.com

fi

**(b) Script for Daily Backup**

#!/bin/bash

src="/home/sgharge/data"

dest="/backup/data\_$(date +%F).tar.gz"

tar -czf $dest $src

echo "Backup stored at $dest"

**(c) Script to Monitor Service**

#!/bin/bash

service="sshd"

if systemctl is-active --quiet $service

then

echo "$service is running"

else

echo "$service is stopped, starting now..."

systemctl start $service

fi

**10. Best Practices**

* Always use #!/bin/bash at the top.
* Use meaningful variable names.
* Comment your script (#).
* Test scripts before scheduling in cron.
* Use **set -e** to exit on error.

**11. Interview Questions (with Answers)**

**Q1. What is a shell script?**  
👉 A file containing Linux commands executed sequentially by a shell.

**Q2. What is the use of #!/bin/bash?**  
👉 Shebang line tells which interpreter to use.

**Q3. Difference between sh and bash?**  
👉 sh is Bourne Shell (basic), bash is Bourne Again Shell (advanced with more features).

**Q4. How do you debug a shell script?**  
👉 Run with:

bash -x script.sh

**Q5. How do you schedule a script in Linux?**  
👉 Using **cron**. Example:

crontab -e

0 2 \* \* \* /home/sgharge/backup.sh

**Q6. How do you handle errors in shell scripts?**  
👉 By checking $?, using set -e, or custom error handling.

**🐚 Shell Scripting – Detailed Notes**

**1. What is Shell Scripting?**

* A **Shell Script** is a text file containing a sequence of commands that the Linux shell (bash, sh, ksh, zsh) executes.
* Purpose: Automate repetitive tasks, system administration, monitoring, backups, deployments, etc.
* Shell: Command interpreter between user and kernel.

**2. Why use Shell Scripts?**

✅ Automate routine tasks  
✅ Simplify complex command sequences  
✅ Useful in system monitoring, backups, deployments  
✅ Enhances productivity for sysadmins/devops engineers

**3. Steps to Create a Shell Script**

1. Create a file with .sh extension
2. vi myscript.sh
3. Add a **shebang line** at the top (defines interpreter):
4. #!/bin/bash
5. Write script commands inside. Example:
6. #!/bin/bash
7. echo "Hello, World!"
8. date
9. Save and exit.
10. Give execute permission:
11. chmod +x myscript.sh
12. Run the script:
13. ./myscript.sh

**4. Shell Script Basics**

**Variables**

* Store values:
* NAME="Sagar"
* echo "My name is $NAME"

**Input from User**

echo "Enter your name: "

read USERNAME

echo "Hello, $USERNAME"

**Command Substitution**

DATE=$(date)

echo "Today is $DATE"

**5. Conditional Statements**

**If-Else**

#!/bin/bash

echo "Enter a number: "

read NUM

if [ $NUM -gt 10 ]; then

echo "Number is greater than 10"

else

echo "Number is 10 or less"

fi

**Case Statement**

#!/bin/bash

echo "Enter choice (start/stop/status): "

read choice

case $choice in

start) echo "Service starting..." ;;

stop) echo "Service stopping..." ;;

status) echo "Service running..." ;;

\*) echo "Invalid choice" ;;

esac

**6. Loops**

**For Loop**

for i in 1 2 3 4 5

do

echo "Number $i"

done

**While Loop**

count=1

while [ $count -le 5 ]

do

echo "Count = $count"

((count++))

done

**Until Loop**

n=1

until [ $n -ge 5 ]

do

echo "Value: $n"

n=$((n+1))

done

**7. Functions**

* Reusable block of code

greet() {

echo "Hello $1, welcome!"

}

greet Sagar

**8. Script Debugging**

Run with:

bash -x script.sh

* -x: shows each command before execution (helps debugging).

**9. Practical Examples**

**Example 1: Backup Script**

#!/bin/bash

SRC="/home/sgharge/data"

DEST="/backup/data\_$(date +%F).tar.gz"

tar -czf $DEST $SRC

echo "Backup stored in $DEST"

**Example 2: Service Monitoring**

#!/bin/bash

service="sshd"

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

echo "$service is running"

else

echo "$service is not running, restarting..."

systemctl start $service

fi

**Example 3: Disk Usage Alert**

#!/bin/bash

THRESHOLD=80

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

if [ $USAGE -gt $THRESHOLD ]; then

echo "Disk usage above threshold: $USAGE%"

fi

**10. Best Practices**

* Always start with #!/bin/bash
* Use meaningful variable names
* Add comments for clarity (# comment)
* Test scripts in a safe environment before production
* Use set -e (exit on error) for critical scripts
* Log outputs for troubleshooting

**11. Interview Q&A**

**Q1. What is the difference between sh and bash?**

* sh: Bourne shell, basic features.
* bash: Bourne Again Shell, advanced (functions, arrays, command history).

**Q2. How do you debug a shell script?**

* Use bash -x script.sh
* Add set -x inside the script to trace.

**Q3. How do you schedule a script to run automatically?**

* Using cron job:
* crontab -e
* 0 2 \* \* \* /home/sagar/backup.sh

**Q4. How to pass arguments to a script?**

* Example:
* ./myscript.sh arg1 arg2

Inside script:

echo "First arg = $1"

echo "Second arg = $2"