Linux Programming: Assignment-8:

NAME: SANDEEP
USN:ENG24CY0047
ROLL NO: 40
SECTION: B

11. What is a user-defined function in shell scripting? Explain with an example.

A: User-Defined Function in Shell ScriptingA user-defined function is a collection of commands under a name.You can use the function anywhere in the script.

It organizes code and prevents repetition.

Syntax

```
function_name() {
    # commands
}
```

Example

```
#!/bin/bash
# This script exhibits a user-defined function
# Define a function
greet() {
    echo "Hello! Welcome to Bash scripting."
}
# Call the function
greet
```

Output

```
Hello! Welcome to Bash scripting.
```

Explanation

greet() → This is the function name.

The instructions within { } execute when you invoke the function.

greet → This line invokes the function and displays the message.

12. Write a bash script with a function that multiply two integer numbers.

A: Script: multiply.sh

```
#!/bin/bash
# This script multiplies two integers using a function
# Function to multiply two numbers
multiply() {
    result=$(( $1 * $2 ))
    echo "The multiplication of $1 and $2 is: $result"
}
# Read two integers from the user
read -p "Enter the first number: " num1
read -p "Enter the second number: " num2
# Call the function with the two numbers
multiply $num1 $num2
```

Steps to run:

1. Save the file as multiply.sh.
2. Make it executable:

```
chmod +x multiply.sh
```

3. Run the script:

```
./multiply.sh
```

Sample Run:

```
Enter the first number: 5
Enter the second number: 4
The multiplication of 5 and 4 is: 20
```

Explanation:

- multiply() → Defines a function that multiplies two numbers.
- $1 and $2 → Represent the first and second arguments passed to the function.

- $(($1 * $2)) → Performs arithmetic multiplication.
- Function is called with the user input to get the result.

This is a clean and reusable way to multiply numbers in a Bash script.

If you want, I can also make a version that multiplies any number of integers using a loop.

13. Explain how arrays (1D, 2D, and 3D) are declared in bash scripting.

A: One-Dimensional (1D) Array

A 1D array is just a list of values.

How to declare:

fruits=("Apple" "Banana" "Orange")

Access elements:

echo ${fruits[0]} # Apple

echo ${fruits[@]} # All elements

2. Two-Dimensional (2D) Array

Bash doesn't support real 2D arrays, but we can make them look like they are present with associative arrays.

How to declare:

declare -A matrix

matrix[0,0]=1

matrix[0,1]=2

matrix[1,0]=3

matrix[1,1]=4

Access elements:

echo ${matrix[1,1]} # 4

3. Three-Dimensional (3D) Array

Bash does not support native 3D arrays, yet we can mimic the use of associative arrays with three indices.

Declaring:

declare -A cube

cube[0,0,0]=10

cube[0,0,1]=20

cube[0,1,0]=30

cube[1,0,0]=40

Accessing elements:

echo ${cube[0,0,1]} # 20

14. Write a shell script to display elements of an array.

A: Script: display_array.sh

#!/bin/bash

This script shows elements of an array

Declare an array

fruits=("Apple" "Banana" "Orange" "Mango")

Show all elements

echo "All fruits: ${fruits[@]}"

Show elements one by one using a loop

echo "Fruits individually:"

for fruit in "${fruits[@]}"

do

    echo "$fruit"

done

Steps to run:

Save the file as display_array.sh.

Make it executable:

chmod +x display_array.sh

Run the script:

./display_array.sh

Sample Output:

All fruits: Apple Banana Orange Mango

Fruits individually:

Apple

Banana

Orange

Mango

15. What is the purpose of cron in Linux?

A: Purpose of cron in Linux

cron is a Linux time-based job scheduler.

It is employed to run commands or scripts automatically at a specified time or interval.

It is helpful in performing tasks such as:

Making backups on a daily basis

Executing system maintenance

Sending automated emails

Updating logs on a regular basis

Important Points:

Executes recurring tasks automatically without human intervention.

Employing cron jobs, which are scheduled commands placed in a file named crontab.

Example of a Cron Job

0 7 * * * /home/user/backup.sh

This executes the script backup.sh daily at 7:00 AM

16. Write a cron job to run a backup script every day at midnight.

A: Cron Job

0 0 * * * /home/user/backup.sh

Explanation of the Fields

Field\tMeaning\tValue for this cron job

Minute\tWhen to run (0–59)\t0

Hour\tHour to run (0–23)\t0 (midnight)

Day\tDay of the month (1–31)\t* (every day)

Month\tMonth (1–12)\t* (every month)

Weekday\tDay of the week (0–7)\t* (every day)

Command\tScript to execute\t/home/user/backup.sh

Steps to Add the Cron Job

Open crontab for editing:

crontab -e

Add the line:

0 0 * * * /home/user/backup.sh

Save and exit.

The script /home/user/backup.sh will now execute automatically every day at midnight.

17. How do you schedule a one-time job using at command?

A: Scheduling a Single Job with at

The at command is utilized to execute a command once at a particular time.

Steps:

Open at and mention the time:

at 14:30

This will schedule a job at 2:30 PM today.

Mention the command to execute:

/home/user/backup.sh

End the input by pressing Ctrl+D.

Example

at 23:00

at> /home/user/backup.sh

at> <Ctrl+D>

This will execute the backup script once at 11:00 PM.

Check Scheduled Jobs

atq

Delete a Scheduled Job

atrm <job_number>

18. Write a script to display disk usage using df and du.

A: Script: disk_usage.sh

```
#!/bin/bash

# This script prints disk usage using df and du

echo "=== Disk Usage of File Systems (df) ==="

df -h   # Prints disk usage of all file systems in human-readable format

echo

echo "=== Disk Usage of Current Directory (du) ==="

du -sh. # Prints total size of current directory in human-readable format
```

Steps to run:

Save the file as disk_usage.sh.

Make it executable:

```
chmod +x disk_usage.sh
```

Run the script:

```
./disk_usage.sh
```

Sample Output:

```
=== Disk Usage of File Systems (df) ===

Filesystem      Size  Used Avail Use% Mounted on

/dev/sda1       50G   20G   28G  42% /

=== Disk Usage of Current Directory (du) ===

2.5G    .
```

Explanation:

.df -h → Displays disk usage of all mounted file systems in human-readable format (KB, MB, GB).

du -sh. → Displays total size of current directory.

-h → Human-readable format,

-s → Summarizes total size.

19. How can you log the output of a script using the tee command?

A: Logging Script Output Using tee

The tee command saves the output of a script on the screen as well as in a file.

Example

#!/bin/bash

sample_script.sh

echo "This is a test message"

echo "Logging script output"

Run the script and log output using tee:

./sample_script.sh | tee output.log

What Happens:

You see the output on the screen.

The same output is saved in output.log.

Append to a Log File

To append to the log file without overwriting:

./sample_script.sh | tee -a output.log

20. Explain with an example how shell scripting can automate system administration tasks.

A: Shell Scripting for System Admin Automation

System admin tasks range from:

Backups of files

Cleaning temp files

Disk space checks

Monitoring system performance

All these tasks can be automated by shell scripts so that you don't have to repeat them manually each time.Example: Backup Script

#!/bin/bash

Script for automatic backup of /home/user/data folder

Create a backup directory with today's date

```
backup_folder="/home/user/backup_$(date +%Y%m%d)"

mkdir -p "$backup_folder"

# Copy files to the backup directory

cp -r /home/user/data/* "$backup_folder"

echo "Backup done successfully at $(date)"
```

How it Automates Tasks

No manual labor: You don't have to copy files manually.

Schedules itself: This script can be run automatically every day using cron.

Logs activity: The script outputs a message with date and time.

Sample Run Output

Backup completed successfully at Tue Oct 7 14:45:12 IST 2025