Assignment 8

Linux Programming

Name : Yathish N R

Class: 3B CY

Roll No: 66

USN : ENG25CY1007

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

A user-defined function is a **reusable block of code** within a shell script that performs a specific task, improving modularity and readability.

Example:

```
# Function Definition
greeting () {
echo "Hello, $1!" # $1 is the first argument passed
}

# Function Call greeting "Alex"
# Output: Hello, Alex!
```

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

```
#!/bin/bash

# Function to multiply two numbers
multiply_numbers () {
# Perform multiplication using arithmetic expansion ($(( )) )
local product=$(( $1 * $2 ))
echo "The product of $1 and $2 is: $product"
}

# Execute the function
multiply_numbers 12 5 # Output: The product of 12 and 5
is: 60
```

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

scripting.

Bash natively supports only **1D arrays** (indexed by numbers or keys). Multi-dimensional arrays (2D, 3D) must be **simulated** using complex indexing schemes or by embedding data strings within a 1D array.

1D Indexed: my_array=(element1 element2 30)

1D Associative: declare -A my_map; my_map[key]="value"

2D / 3D: Simulation is required, as there is no direct declaration syntax.

4. **Write a shell script to display elements of an array.**

```
#!/bin/bash
planets=("Mars" "Jupiter" "Saturn"
"Uranus")
echo "--- Array
Elements ---"
# Iterate through the array elements using the @
symbol for planet in "${planets[@]}"; do   echo
"Found planet: $planet" done
```

5. **What is the purpose of cron in Linux?**

Cron is the time-based job scheduler daemon in Linux. Its purpose is to **automate the execution of recurring tasks** (cron jobs or scripts) at predetermined times, dates, or intervals for system maintenance (like backups and log rotation).

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

A cron job runs at midnight (0 minutes, 0 hours) every day of the month, every month, and every day of the week.
Crontab Entry:

```
0 0 * * * /bin/bash
/path/to/backup_script.sh
```

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

The at command is used to schedule a task to run **once** at a specified time in the future.
Start the scheduling session: at 02:00 PM tomorrow
Enter the command(s) at the at> prompt: cleanup_temp_files.sh
Press **Ctrl+D** to save the job to the queue.

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

```
#!/bin/bash
echo "--- Filesystem Disk Space (df -h) ---" # df: Shows
overall disk usage on mounted filesystems df -h   echo -e
"\n--- Directory Disk Usage Summary (du -sh /var) ---" #
du: Shows disk usage for a specific file or directory du
-sh /var
```

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

The **tee command** allows the output of a command or script to be **both displayed on the terminal** (Standard Output) **and saved to a file** simultaneously.
Example:

```
# Run ls -l, display results, and append (-a) the output
to script_log.txt
```

```
ls -l /etc | tee -a
script_log.txt
```

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

Shell scripting automates repetitive, tedious, or scheduled system administration tasks, reducing errors and ensuring consistency.

Example: Automated User Account Creation

A script can read a list of new users from a file and automatically create accounts, set passwords, and assign groups.

```
#!/bin/bash

USER_LIST="new_users.txt"

# Loop through each line (username) in
the file while IFS= read -r USERNAME; do
# Check if the user already exists   if !
id "$USERNAME" &>/dev/null; then
# Create the user account with a temporary
password     useradd -m "$USERNAME"     echo "User
$USERNAME created."   fi done < "$USER_LIST"
```