Lab Assignment 2

Name: Vivek Sapkal | Roll No.: B22AI066

Q.1)

➢

```
#!/bin/bash

check_cpu() {
    echo "CPU Usage:"
    top -bn1
}

check_memory() {
    echo "Memory Usage:"
    free -m
}

check_disk() {
    echo "Disk Space:"
    df -h
}

main() {
    echo "System Health Check:"
    check_cpu
    echo
    check_memory
    echo
    check_disk
}
main
```

➢ **Description:**

➢ ‘#!/bin/bash’: This is called a shebang, and it indicates that the script should be executed using the Bash shell.
➢ ‘check_cpu(){ … }’ defines a function to check cpu usage using the top command; top -bn1 runs the top command in batch mode for one iteration to show cpu usage information.
➢ ‘check_memory(){ … }’ defines a function to display memory usage using the ‘free -m’ command which shows memory usage in megabytes.
➢ ‘check_disk(){ … }’ defines a function to display disk space information using the ‘df -h’ command.
➢ ‘main(){ … }’ defines the main function which calls the above-mentioned three functions to display overall system health checks.
➢ Last line calls the main function.

Q.2)

➢

```
#!/bin/bash

echo "User Account Information"

printf "%-20s %-15s %-15s %-15s\n" "Username" "User ID" "Home
Directory" "Shell"
echo "------------------------------------------------------------"

while IFS=: read -r username password uid gid info home shell; do
    if [ "$uid" -ge 1000 ]; then
        printf "%-20s %-15s %-15s %-15s\n" "$username" "$uid"
"$home" "$shell"
    fi
done < /etc/passwd
```

➢ **Description:**

➢ First prints a simple header using echo.
➢ The next line prints a formatted header for the user account information. It specifies the width and alignment for each field (‘%-20s’ means left-aligned string with a width of 20 characters).
➢ Next prints a separator using echo.
➢ Next it starts a while loop that reads each line of the ‘/etc/passwd’ file and assigns values to variables (username, password, uid, gid, info, home, shell) using `:` as a delimiter.
➢ The -r option disables backslash escaping, ensuring that backslashes are treated as literal characters.
➢ Next it checks if the uid (User ID) is greater than or equal to 1000, because regular users have uids greater than or equal to 1000. If the condition is satisfied then it prints the formatted user account information for each regular user.
➢ Last line ends the while loop, indicating that it should continue reading lines from the ‘/etc/passwd’ file until there are no more lines.

Q.3)

➢

```
#!/bin/bash

target_directory="/home/vivek/PCS_2/archives"

archive_directory="/home/vivek/PCS_2/Assignment_2"

mkdir -p "$archive_directory"

find "$target_directory" -type f -mtime +30 -exec mv {}
"$archive_directory" \;
```

```
echo "Cleanup complete."
```

- **Description:**

- Paths to the directory to clean up and to the archive directory are stored in variables.
- The ‘-p’ option used with mkdir ensures that the specified directory and its parent directories are created if they don't exist.
- Then the ‘find’ utility is used to locate files(-type f) in the specified directory that are older than 30 days(-mtime +30).
- Then the ‘-exec mv {} “$archive_directory” \; ’ part of the command executes the move command to move file to the archive directory for each file round.

Q.4)

-

```
#!/bin/bash

if [ "$#" -ne 1 ]; then
        echo "Usage: $0 <path_to_filesystem>"
        exit 1
fi

check_disk_usage() {
    echo "Disk Usage:"
    df -h "$1"
    echo
    echo "File space usage:"
    du -h --max-depth=1 "$1"
}

check_file_count() {
    echo "File Count:"
    find "$1" -type f | wc -l
}

check_directory_count() {
    echo "Directory Count:"
    find "$1" -type d | wc -l
}

main() {

    filesystem_path="$1"

    echo "File System Report for: $filesystem_path"
    echo

    check_disk_usage "$filesystem_path"
    echo
    check_file_count "$filesystem_path"
    echo
    check_directory_count "$filesystem_path"
}

main "$1"
```

- **Description:**

- First the if condition checks if the number of arguments provided is exactly one or not. If there is not exactly one argument then it prints a usage message and exits the script with an error code (1).
- check_disk_usage() { ... }: This defines a function named check_disk_usage that takes one argument (the path to the filesystem) and prints the disk usage information using the df -h command. The du -h command prints the file space usage of the specified directory.
- check_file_count() { ... }: This defines a function named check_file_count that takes one argument (the path to the filesystem) and prints the count of files using the find command piped to wc -l.
- Similarly, check_directory_count prints the number of directories.
- Then a main function is defined which calls the three functions to display disk usage, file count and directory count.
- filesystem_path="$1" : This assigns the first command-line argument (the filesystem path) to the variable filesystem_path.
- The last line calls the main function with the provided filesystem path.