

Cloud Computing
Assignment 2 – Shell Scripting

Name: Pranav Rao

Roll No: 333045

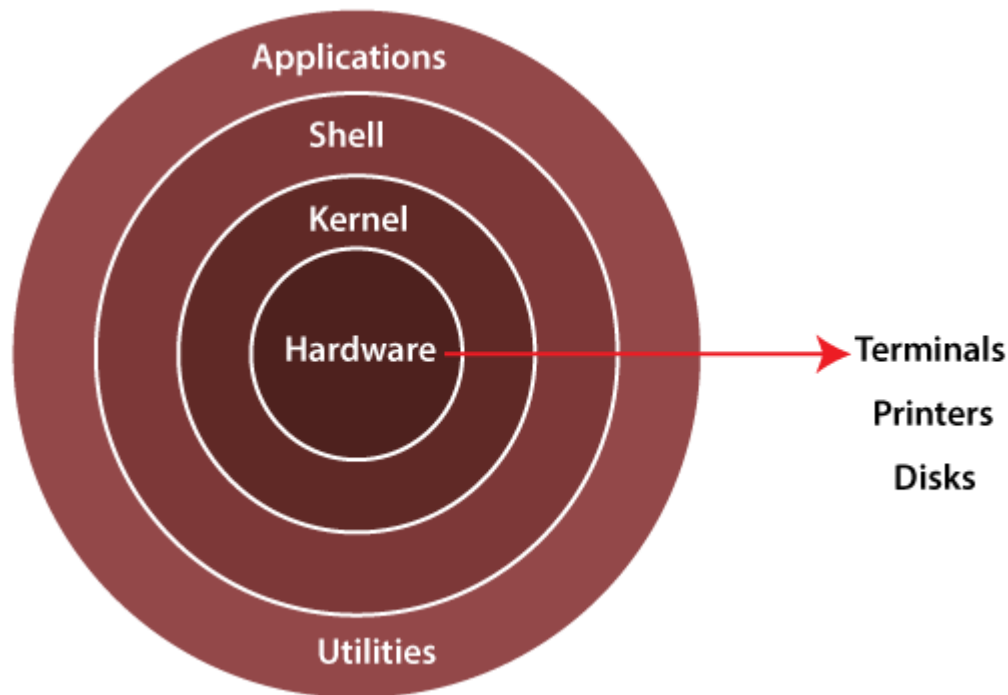
Division: IT-C

Batch: C2

Theory:

1) What is Shell (Linux kernel architecture diagram)

A Shell provides you with an interface to the Unix system. It gathers input from you and executes programs based on that input. When a program finish executing, it displays that program's output. Shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of a shell, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions. A Shell reads your input after you press Enter. It determines the command you want executed by looking at the first word of your input. A word is an unbroken set of characters. Spaces and tabs separate words.



2) Types of shell

The shell is mainly of two types, then these two types are further categorized: types of

shell is:

- Bourne Shell

- Bourne shell is known as the first shell to be introduced, it is represented by “sh”. This shell got popular because of its quite compact nature.

- It was made the default shell for the SOLARIS operating system and was used as a Solaris administration script. It has very high-speed operations

- Bourne’s shell was not able to handle logical and arithmetic operations. It was less interactive because of the lack of comprehensive features. Also, it is not able to recall previously used commands.

- The Bourne shell can be further divided into 5 types.

1. Bourne shell (sh)
2. Korn Shell (ksh)
3. Bourne Again shell (bash)
4. POSIX shell (sh)

- C Shell

- The C shell was designed by Bill Joy at the University of California. It is represented using “csh”.

- The C shell was designed with the purpose of supporting programming languages.

- It was specifically designed to support in-built features like solving arithmetic operations and syntax of programming languages like C.

- Unlike Bourne and other Linux shells, the C shell can maintain and history of previously used commands, and those commands can be used whenever required.

- Shell is the most important and powerful tool in the system.

Without a shell, it's

impossible to utilize the system's features and functionality to its fullest.

Problem statements for shell scripting:

2a) Write a shell script to check user is root user or not

CODE:

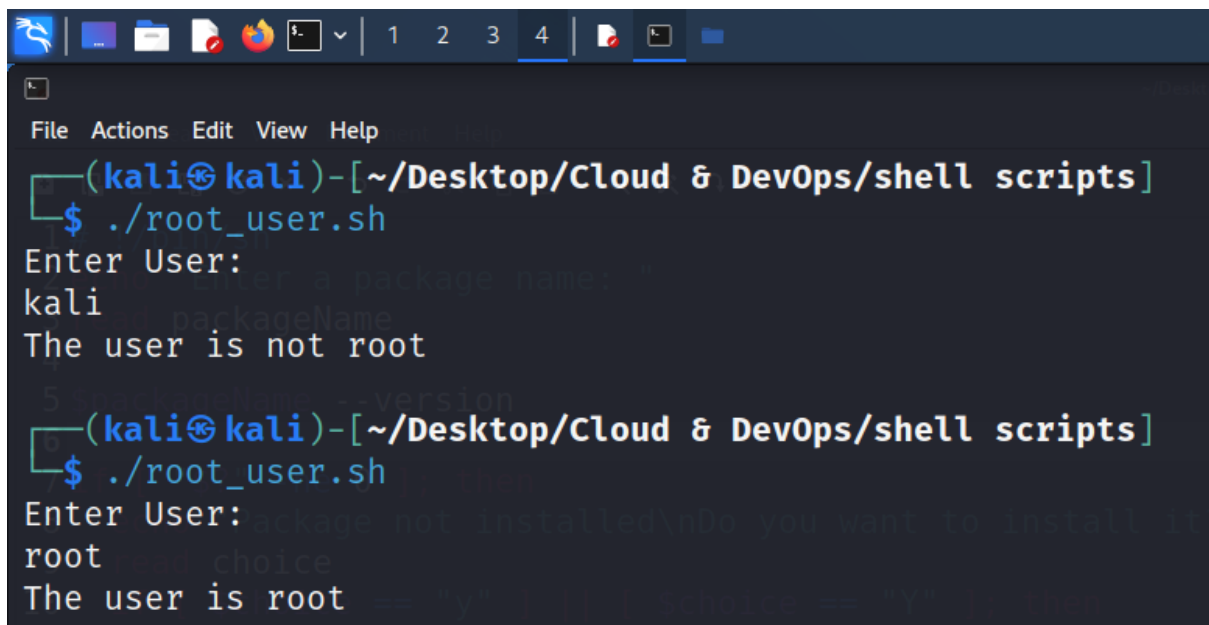
```
#!/bin/bash

echo "Enter User:"

read name

if [ `id -u $name` -eq 0 ]
then echo "The user is root"
else
echo "The user is not root"
fi
```

OUTPUT:

A screenshot of a terminal window with a dark background. The window title bar shows icons for a file manager, a web browser, and a terminal. The terminal content shows a user at a kali machine in the directory ~/Desktop/Cloud & DevOps/shell scripts. The user runs the command ./root_user.sh. The script prompts for a user name, and the user enters 'kali'. The script outputs 'The user is not root'. The user runs the script again, enters 'root', and the script outputs 'The user is root'.

```
(kali@kali)-[~/Desktop/Cloud & DevOps/shell scripts]
$ ./root_user.sh
Enter User:
kali
The user is not root

(kali@kali)-[~/Desktop/Cloud & DevOps/shell scripts]
$ ./root_user.sh
Enter User:
root
The user is root
```

2b) Write a shell script to install any particular software (ex: java or python)

CODE:

```
# !/bin/sh

echo "Enter a package name: "
read packageName
$packageName --version
if [ "$?" -ne 0 ]; then
echo "Package not installed\nDo you want to install it?[Y/N]"
read choice
if [ $choice == "y" ] || [ $choice == "Y" ]; then
sudo apt-get update
sudo apt-get install $packageName
else
exit
fi
else
echo "Package already installed"
exit
fi
```

OUTPUT:

```
kali@kali: ~/Desktop/Cloud & DevOps/shell scripts
File Actions Edit View Help
(kali@kali)-[~/Desktop/Cloud & DevOps/shell scripts]
└─$ bash software_download.sh
Enter a package name:
neovim
software_download.sh: line 5: neovim: command not found
Package not installed
Do you want to install it?[Y/N]
Y
[sudo] password for kali:
Get:1 http://kali.download/kali kali-rolling InRelease [30.6 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [19.2 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [44.0 MB]
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [111 kB]
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [164 kB]
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [237 kB]
Get:7 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [922 kB]
Fetched 64.7 MB in 13s (4,949 kB/s)
Reading package lists ... Done
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following additional packages will be installed:
  libmsgpackc2 libtermkey1 libtree-sitter0 libunibilium4 libvterm0 lua-luv
  neovim-runtime python3-pynvim xclip
Suggested packages:
  ctags vim-scripts
The following NEW packages will be installed:
  libmsgpackc2 libtermkey1 libtree-sitter0 libunibilium4 libvterm0 lua-luv
  neovim neovim-runtime python3-pynvim xclip
0 upgraded, 10 newly installed, 0 to remove and 1727 not upgraded.
```

```
(kali@kali)-[~/Desktop/Cloud & DevOps/shell scripts]
└─$ bash software_download.sh
Enter a package name:
python
Python 3.10.5
Package already installed
```

2c) Write a shell script to check disk usage of the system and if disk usage is more than 90% it should send an email to system admin. This script should run everyday at 8:00 AM.

CODE:

```
# !bin/sh

df -Ph | grep -vE '/Filesystem|tmpfs|cdrom' | awk '{ print $5,$1 }' |
while read output;
do
echo $output
used=$(echo $output | awk '{print $1}' | sed s/%//g)
partition=$(echo $output | awk '{ print $2 }')
if [ $used -gt 90 ]; then
echo "The partition \"$partition\" on $(hostname) has used $used%
at $(date)" | mail -s "Disk Space Alert: $used% Used On
$(hostname)"
kali@kali
echo "Email sent regarding $partition"
else
echo "Disk space usage is in under control"
fi
done
```

OUTPUT:

```
(kali㉿kali)-[~/Desktop/Cloud & DevOps/shell scripts]
$ bash disk_usage.sh
Use% Filesystem
disk_usage.sh: line 8: [: Use: integer expression expected
Disk space usage is in under control
0% udev
Disk space usage is in under control
18% /dev/sda1
Disk space usage is in under control
```

2d) write a shell script to take mysql database server backup. This script should run weekly on every sunday at 11:00 PM.

CODE:

```
#!/bin/sh

echo "starting database backup"

db_backup="mydb.gz"

sudo mysqldump -uroot -p test | gzip -c >
./db_backups/${db_backup}

if [ "$?" -eq 0 ]; then
echo "db backup complete"
else
echo "db backup failed"
fi

# sudo chmod +x ./backup.sh (Run in terminal from this folder)

# sudo crontab -e

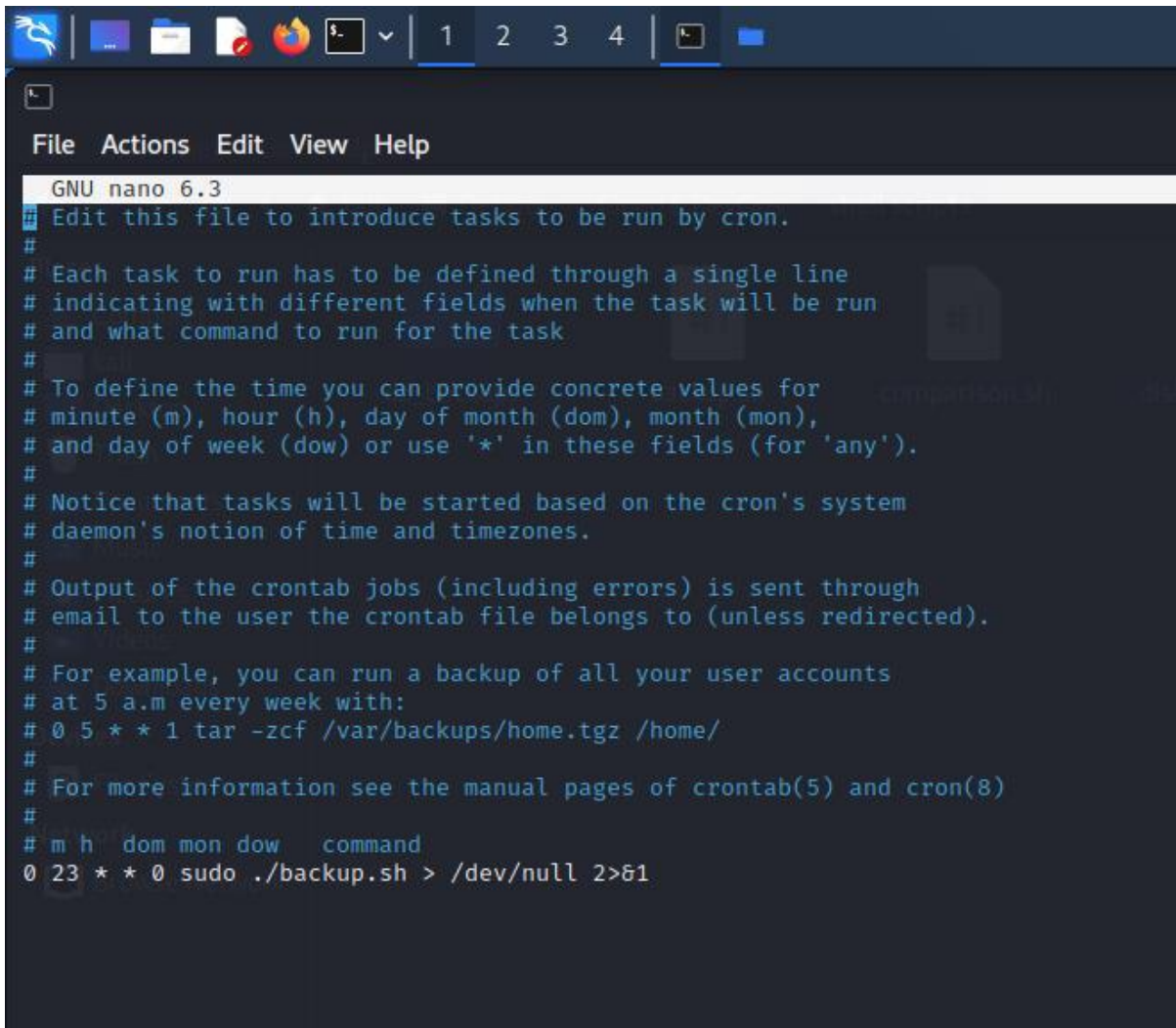
# 0 23 * * 0 sudo ./backup.sh > /dev/null 2>&1
```


OUTPUT:

```
(kali@kali)-[~/Desktop/Cloud & DevOps/shell scripts]
└─$ ls
backup.sh  comparison.sh  db_backups  disk_usage.sh  root_user.sh  software_download.sh

(kali@kali)-[~/Desktop/Cloud & DevOps/shell scripts]
└─$ bash backup.sh
starting database backup
Enter password:
mysqldump: Got error: 2002: "Can't connect to local server through socket '/run/mysqld/mysqld.sock' (2)" when trying to connect
db backup complete
```

For 2d) The executable files are added to crontab



```
GNU nano 6.3
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
0 23 * * 0 sudo ./backup.sh > /dev/null 2>&1
```