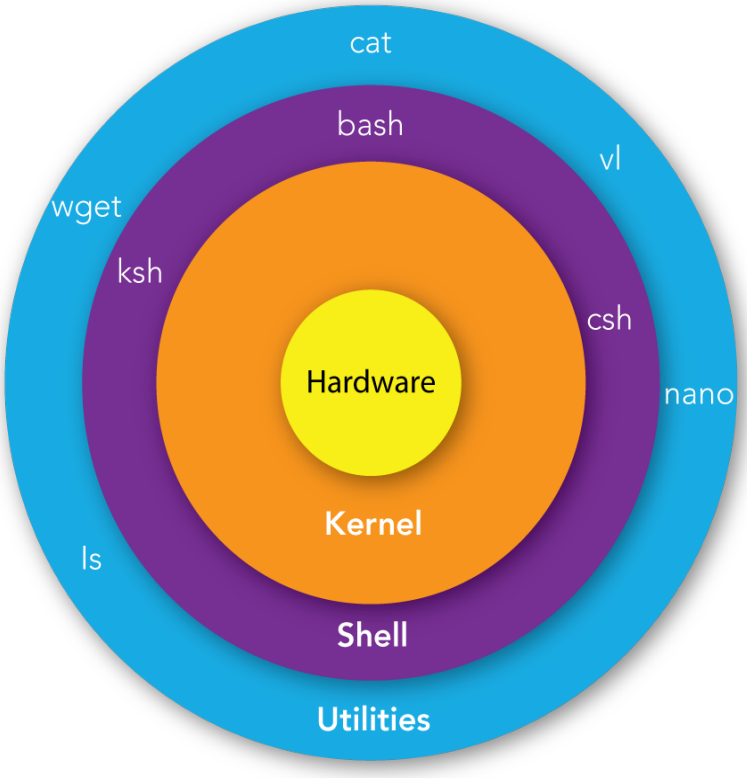
2. Shell Scripting

- Name: Om Deore
- PRN: 22110318
- Div: TY(A)
- RollNo: 321022

What is shell?

A shell is a special user program that provides an interface for the user to use operating system services. Shell accepts **human-readable commands** from users and converts them into something which **the kernel can understand**. It is a command language interpreter that executes commands read from input devices such as keyboards or from files. The shell gets started when the user logs in or starts the terminal.



Shell can be accessed by users using a command line interface. A special program called Terminal in Linux/macOS, or Command Prompt in Windows OS is provided to type in the human-readable commands such as “cat”, “ls” etc

Different types of shells

Shell	Complete path-name	Prompt for root user	Prompt for non root user
GNU Bourne-Again shell(bash)	/bin/bash	bash-VersionNumber#	bash-VersionNumber$
C shell (csh)	/bin/csh	#	%
Korn shell (ksh)	/bin/ksh	#	$
Z shell (zsh)	/bin/zsh	"hostname"#	"hostname"%

2a) Write a shell script to check user is root user or not (HINT: study "id" command in Linux)

`id` command outputs information about current user.
when user is normal user

```
$ id
uid=1000(om_d)gid=1000(om_d)groups=1000(om_d),24(cdrom),25(floppy),27(sudo),
29(audio),30(dip),44(video),46(plugdev),100(users),106(netdev),112(bluetooth
),114(lpadmin),118(scanner)
```

when user is root user

```
# id
uid=0(root) gid=0(root) groups=0(root)
```

with both of these outputs we can grep string root from these outputs. If grep doesn't return anything it will be treated as false in if statement and we can say that user is not root user, otherwise user is a root user.

```
# check_user.sh
#!/bin/bash
if [[ $(id | grep root) ]]; then
    echo "This is root user"
else
    command
    echo "This is not root user"
fi
```

output of above script is as following
when normal user

```
$ ./check_user.sh
This is not root user
```

when root user

```
# ./check_user.sh
This is root user
```

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

One of the most important features of Linux is availability of software via package managers. Package managers are command line tools which help use install applications, software via single command. Its much preferred way to install application than clicking in a GUI window multiple times.
Based on different Linux distributions, different package managers are used.All these package managers have different versions of packages available

Ex. if you are using Debian based distro you are using 'apt' package manager which is known to ship only stable packages. It means you will get stable, bug free software but you may have to wait to get latest and greatest features. For Arch based distros you use package manager named 'pacman'. It is a bleeding edge package manager which means you get latest features as soon as they are released by developers, on the risk of packages breaking your system and encountering bugs. Fedora you come across package manager called 'dnf' which false under bleeding edge side of package managers but not as fast as pacman so you get best of both worlds.

While writing a script to install a package we first must know on which distro we are so we can choose our package manger
The information about our distro is stored in `/etc/os-release` file.

```
$ cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"
NAME="Debian GNU/Linux"
VERSION_ID="12"
VERSION="12 (bookworm)"
VERSION_CODENAME=bookworm
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
```

here we can see attribute ID which will give us name of distro, we will try to grep distroname from this file.

```
# install.sh
#!/bin/bash
if [[ $(grep "debian" /etc/os-release) ]]; then
    echo "debian based system recognised"
    sudo apt update
    sudo apt upgrade
    sudo apt install python3

elif [[ $(grep "arch" /etc/os-release) ]]; then
    echo "arch based system recognised"
    sudo pacman -Syu
    sudo pacman -S python-pip

elif [[ $(grep "fedora" /etc/os-release) ]]; then
    echo "fedora based system recognised"
    sudo dnf upgrade
    sudo dnf install python
else
    echo "unknown system recognised"
fi
```

This universal script installs python on listed 3 distros (and their sub-distros)

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.

`du` command returns disk usage of specified directory

```
# gives disk used by user directory
$ du -sh ~
20G     /home/om_d
```

theres also `df` command which gives amount of space available on file system

```
$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
udev             8023760        0   8023760   0% /dev
tmpfs            1611424     1692   1609732   1% /run
```

```
/dev/nvme0n1p5 300278136 38525400 246426376  14% /
tmpfs            8057116     3684   8053432   1% /dev/shm
tmpfs               5120        8      5112   1% /run/lock
/dev/nvme0n1p1     98304    50418     47886  52% /boot/efi
tmpfs            1611420       76   1611344   1% /run/user/1000
```

our system is on `/dev/nvme0n1p5` file system, there we can see Use% being 14%, we can extract this percentage value from here. For that we will use grep, and sed.
To send mail, we will use mail command

```
$ echo "message you want to send" | mail -s "subject of mail"
"to_user@xyz.com"
```

So the script becomes following

```
#!/bin/bash
percentage_usage=$(df | grep "p5" | grep "[0-9]*%" --only-matching)
percentage_usage=$(echo "$percentage_usage" | sed s'/.$//')
echo "$percentage_usage"

if [[ $percentage_usage -gt 90 ]]; then
    # send mail
    message="Disk usage on the system has exceeded 90% threshold. Current
usage is $percentage_usage %"
    echo "$message" | mail -s "Disk usage alert" admin@email.com
fi
```

To run this script everyday at 8 AM add following lines to crontab file.

```
0 8 * * * /path/to/check_usage.sh
```

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

```
#!/bin/bash

# MySQL database credentials
DB_USER="your_username"
DB_PASS="your_password"
DB_NAME="your_database_name"
```

```
# Backup directory
BACKUP_DIR="/path/to/backup/directory"

# Date format for backup file
DATE=$(date +"%Y%m%d_%H%M%S")

# Dump the MySQL database
mysqldump -u "${DB_USER}" -p"${DB_PASS}" "${DB_NAME}" >
"${BACKUP_DIR}/${DB_NAME}_${DATE}.sql"

# Check if the backup was successful
if [ $? -eq 0 ]; then
    echo "MySQL database backup completed successfully."
else
    echo "Error: MySQL database backup failed."
fi
```

To schedule this script to run weekly on Sundays at 11:00 PM, you can set up a cron job. Add the following line to your crontab

```
0 23 * * 0 /path/to/mysql_backup.sh
```