# Linux

Page No.
Date

Think of linux as the brain of a compu-ter (operating system) that controls everything.

It's like Android for mobile but used in servers, websites and cloud platform.

In devops, you use linux to control servers (big computers running websites, apps, etc.

## Why devops Developers Need linux ?

- Run Docker containers
- Set up kubernetes clusters.
- deploy websites / app.
- Manage cloud servers (like AWS, Azure)
- Automate things with scripts.
- Monitor server health, logs and performance.

## Understanding Linux structure ▷

linux is like a house with rooms and drawers.

/ Main gate (root of everything)

| | |
|---|---|
| /home/user | your personal room. |
| /etc | electrical panel. |
| /var/log | CCTV footage (logs) |
| /bin | Toolbox (basic commands) |
| /tmp | Trash bin (temporary stuff) |

Talking to linux: (Terminal = Remote control).

- You type commands to make linux do things.
- This is called using the command line or Terminal.

## Must-Know Commands

| Action | command | Laymen |
|---|---|---|
| show current location | pwd | "Where I am" |
| List files | ls | "what's in this folder" |
| Move to folder | cd folder | "Go to the room" |
| Create folder | mkdir name | "Build a room" |

Page No.
Date

Create file — touch file.txt — "Create a document"

Copy file — cp file1 file2 — "Photocopy file"

Move / rename — mv old new — "Shift or rename file"

Delete — rm file — "Throw it in trash".

file permissions (lock & key)

files / folders can be locked or shared using permissions.

r — Read (can see it)

w — write (can write it)

x — Execute (can run it like a prog).

chmod 755 file = owner can do anything. others can only read / run.

→ Users and Groups (Family in House)

- Users: People in the house (you, admin, guest).
- Groups :- Teams sharing things.

Important :-

sudo — Boss power (do anything)

Page No.
Date

- Add user samit → Add user.
- Give sudo : usermod -aG sudo samit

⇒ Networking In Linux (Phone System)

You connect to other servers like calling someone.

ping Google.com — Test internet is working

curl website — See websites from command line.

ssh user@ip — Remote login to another computer.

⇒ Package manger :—

Install software / tools.

apt update — Refresh opp list

apt install nginx — Install web server.

apt remove nginx — Uninstall.

Scripting in Bash :-

Instead of typing 10 commands every day write them in a script and run once.

Page No.
Date

```
#! /bin/bash

echo "Hello Devops"
uptime
df -h
```

Save as myscript.sh
Run with bash myscript.sh

⇒ Crontab (Alarm clock).

Schedule task (backup: everyday at 2AM)

crontab -e  Edit schedule.

0 2 * * * /backup.sh  Run at 2AM daily

⇒ Log

# Bash shell scripting :-

What is Bash?
- A shell language
- Bourne Again shell
- Easy commands.

Why learn Bash?
- Most used shell
- Stands the test of time
- Comes with linux

→ WSL :- we will use WSL on windows (Ubuntu).

→ Echo :- will display the text as it is an argument.

→ vim :- it is used to create textfile.txt

:w ⇒ to write & save the changes that we have made

:q → to exit without saving. the change.

:wq ⇒ to save the change & exit.

:q! → without saving.

I → to insert & write into the textfile.

A → Append into the textfile.

cat. textfile.txt → to show the content in the textfile.

→ vim .shell test.sh → Vim you can type append mode.

ls → list all the files in our current directory.

pwd : we can verify that we are indeed in the home directory.

pwd → print working directory.

~~bash~~ *

bash shell test.sh → This will output the command we entered in the shell script using the bash.

---

→ Not Good

cp /my/location/from/my/location/to

cp/my/location/from/here /my/location/to/there.

→ Good

MY_LOCATION_FROM = /my/location/from

MY_LOCATION_TO = /my/location/to

vim interactiveshell.sh

```
#!/bin/bash

echo what is your first name?
read FIRST_NAME
echo what is your last name?
read LAST_NAME

echo HELLO $FIRST_NAME $LAST NAME
```

esc + :wq

chmod u+x interactiveshell.sh
./interactiveshell.sh

---

Positional Arguments:-
commands can take in arguments at a specific position.

echo (0) ↓ space Hello (1) ↓ Space. THERE (2) - position is specified by space.

---

vim posargu.sh

```
#!/bin/bash
echo Hello $1 $2
```

esc + :wq

chmod u+x posargu.sh
./posargu.sh Samit Singh

Page No.
Date

→ **Piping :-**

Command one :-

$ ls -l /usr/bin

ls -l /usr/bin | grep bash

piping symbol

**Output redirection :- > >>**

echo Hello world ! > hello.txt

It also overwrite it. cat Hello.txt

echo Good day to you ! > hello.txt

cat hello.txt

---

remove ~~echo~~ rm hello.txt

echo Hello world >> hello.txt

It appends cat hello.txt

echo Good day to you >> hello.txt

cat hello.txt

---

word Count ~ wc -w Hello.txt

also shows the file name.

Direct word count

```
wc -w < hello.txt
```

```
cat << EOF
> I will
> write some
> text here
> EOF
I will
write some
text here
```

String should be 3 in double cout "" otherwise it will not show the word count.

```
wc -w <<< "Hello there word count !"
```

## Test operators :-

Exit code 0 mean command exit with no errors.

```
[ hello = hello ]   ( _ = space)
echo $?
```

```
[ 1 = 0 ]
echo $?
```

```
[ 1 -eq 1 ]
echo $?
```

## If / Elif / Else :-

vim ifelifelse.sh

```
#!/bin/bash
if [ $# , ]
if [ ${1, , } = samit ]; then
        echo "Oh, you're the boss
        here. Welcome!"
elif [ ${1, , } = help ]; then
    echo "Just enter your username,
          duh!"
else
    echo "I don't know who you
          are. But you're not the
          boss of me!".
fi
```

./ifelifelse.sh samit

./ifelifelse.sh help

Case statement :-

vim login.sh

```
#!/bin/bash
```

```
case ${1,,} in
    samit | administrator )
        echo "Hello, you're the boss
              here!"
        ;;
help )
        echo "Just enter your username
              !" ;;
* )
    echo "Hello there. You're not
    the boss of me."
esac.
```

chmod u+x login.sh

```
./login.sh Samit
./login.sh help
./login.sh help
./login.sh administrator.
```

Array :-

```
MY_FIRST_LIST=(one two three four five)
echo $MY_FIRST_LIST
one
echo ${MY_FIRST_LIST[@]}
one two three four five
echo ${MY_FIRST_LIST[0]}
one
```

⇒ for loop :-

```
for item in ${MY_FIRST_LIST[@]}; do
 echo -n $item
 | wc -c; done
3
3
5
4
4
```

function :-

```
#!/bin/bash

 showuptime() {
local up=$(uptime -p | cut -c4-)
local since=$(uptime -s)
    cat << EOF

-----

This machine has been up for ${up}
It has been running since ${since}

-----
EOF
}
showuptime
echo $up
echo $since
```

up = "before"
since = "function"
echo $up
echo $since

vim functionposargu

```
#!/bin/bash
showname () {
        echo hello $1
}
showname Herbert
```

chmod u+x functionposargu.sh
./functionpesargu.sh
~~hello~~

Exit codes :-

vim functionposargu.sh

```
#!/bin/bash
showname () {
        echo hello $1
        if [ ${1,,} = herbert ]; then
            return 0
        else
            return 1
        fi
}
        $1
if [ $? = 1 ]; then
    echo "someone unknown called the
        function!"
fi
```