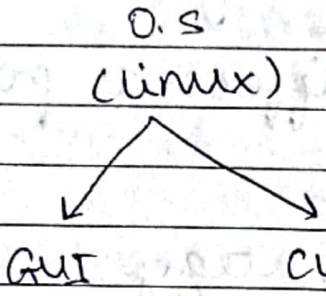
SHELL SCRIPTING WORKSHOP DAY-1

PAGE NO.:
DATE: 24/Apr/21



- On the serverside we generally use only CLI and remove GUI for optimization.

- For Backup
  I download
  II compress
  III copy

echo $?
↳ gives exit code of the last cmd

man <cmd>
↳ gives the manual of the cmd.

echo $SHELL
↳ environmental variable
↳ /bin/bash

x=5
echo x
↳ x
echo $x
↳ 5 ↳ variable

PAGE NO. :
DATE : / /

vim basic.sh
echo "hello vimal"

bash basic.sh

PATH=/root/shellscripting:$PATH

basic.sh
↳ Permission denied

chmod +x basic.sh

#! → hashbang/shebang
↳ gives instruction ~~to~~ ~~about~~
which shell we are using

vim basic.sh
#!/usr/bin/bash
user="vimal"
echo "hello $user"

# vim /root/.bashrc
PATH=/root/shellscripting:$PATH
↳ this will make it permanent

# TO make variables dynamic
user=$1

basic.sh rahul abhi
$0 $1 $2

PAGE NO.:
DATE: / /

user=$@
↳ all the arguments are stored in ~~the~~ ~~last~~ user var

- Iteration / Loop

```
# x=5
# for i in a b c d e
> do
> echo $i
> done
```

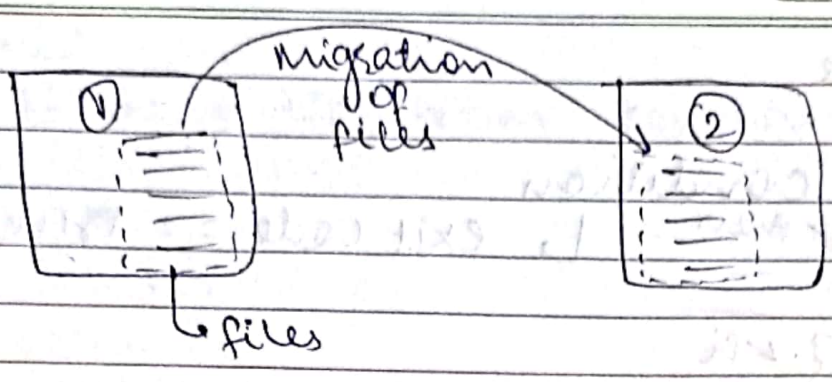
or

~~# for~~

```
# for i in a b c d e ; do echo $i ; done

# u="a b c d e f"
# for i in $u; do echo $i; done
```

or

```
#!/usr/bin/bash
user=$@
for i in $user
do
    useradd $i
    echo "hello $i, user created"
done
```

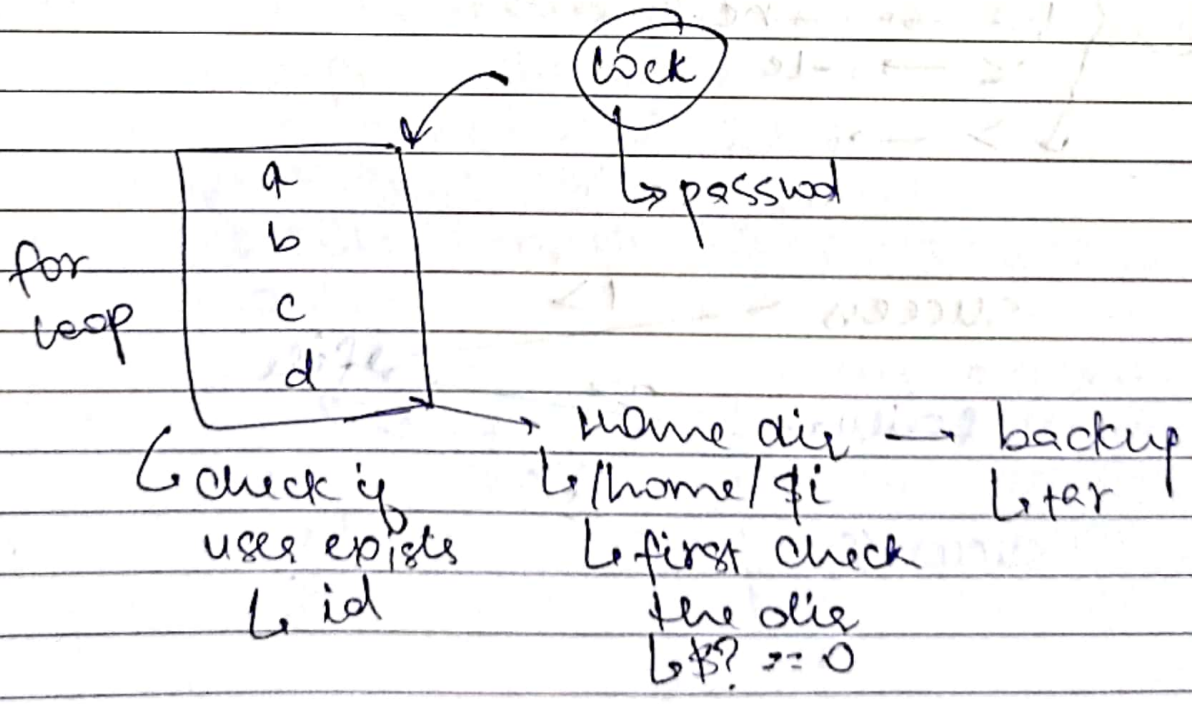



```
# vim db.csv
pop1
p2
user3
harry
eric
```

```
# x=$(cat db.csv)
# for i in $x; do echo $i; done
```

- Now we can edit basic.sh

```
user=$(cat db.csv)
```



- If / Else

```
if condition
{ ← then          └ exit code == True
   ===
} ← fi
else      →  exit code != True
{  ===

}
```

```
# if [1 -le 2]
> then            → exit code
> echo "hi"          if $? == 0
> fi                 ∴ condition is run
```

Shell operators:

```
== → -eq
!= → -ne
<  → -le
>  →
```

```
success        1>
                         → file
failure        2>
               &>
success/fail
```

PAGE NO. :
DATE : / /

read
↳ takes i/p from keyboard

read n
↳ hello
echo $n
↳ hello

read -p "enter name:" n

vim lock.sh
#!/usr/bin/bash
read -p "enter username:" user

if id $user &> /dev/null
then
passwd -l $user
else
echo "$user doesn't exists"
fi

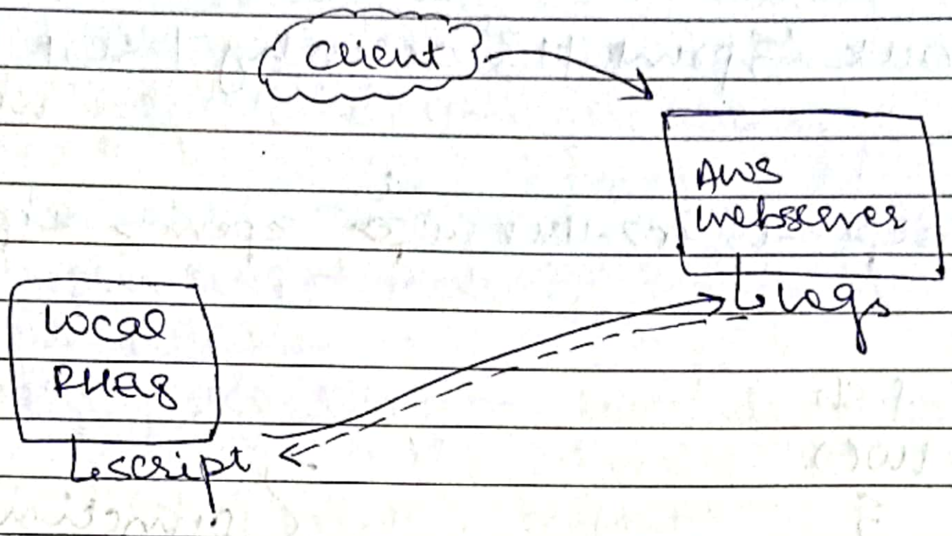
- while condition → 0
{ do
=
=
=
=
} done

when $?!=0
it exits/stops

```
# while:
do
  read ___
if [-z $user]
   ===
   ===

   exit 0

  ===

# grep "^j2:" /etc/passwd

# cut -f6 -d ":" /etc/passwd

# x=$(grep ^j2: /etc/passwd | cut -d ":"
                        -f 6 /etc/passwd)
# echo $x

# vim lock.sh
passwd -l $user
homeDir =$(grep ^${user}: /etc/passwd |
        cut -d ":" -f 6)
done
tar -cf $homeDir ${user}.tar
```

PAGE NO.:
DATE: 28Apr21

SHELL SCRIPTING WORKSHOP DAY-2



- We need to transfer the aws key to our local system.

~~# [illegible]~~

\# cd /root/shellscripting/

- Instead of cut cmd we can also use awk cmd. It is much more powerful. It is also a personal scripting language.

\# awk '{print}' access_log
\# awk '{print $1}' access_log
↳ 1st column
↳ $2 → 2nd column
↳ by default it is space seperated.

\# awk '{print $1 "---" $6}' access_log
↳ 1st cmn --- 6th cmn.

PAGE NO.:
DATE: / /

awk '{print $1}' access_log | sort | uniq | wc -l

scp -i ec2-user <pem> <ip> <Path>

f.sh

```
wc()
{
  echo "hi $1 ..."
}
```

} function

wc nimal

wc() { date; return 11;}
↳ fn with return code 11

unset wc
↳ removes the fn.

touch a{1..13}.txt
↳ .. means range
↳ creates a1.txt, a2.txt, ..., a13.txt

for i in a{10..13}.txt; do ls -l $i; done | awk '{print $5}'

PAGE NO.:
DATE: / /

awk '{ print NR, $0 }' my.txt
↳ prints the records with their respective line number.

awk 'NR==3 { print $0 }' my.txt
↳ prints 3rd record.

- NF → record no in descending order

awk 'END{ print }' my.txt
↳ prints last line of the file.

awk '/krish/{ print }' /etc/passwd
↳ searches the pattern
i.e we can use all regex symbols.

```
# if [1 -eq 2]
> then
> echo "ok"
> elif [2 -eq 2]
> then
>  echo "ok2"
> else
> echo "ok3"
> fi
```

*) ← default

```
# vim case.sh
option = "$1"
case $option in
    -f) echo "i know f" ;;
    -d) echo "i know d" ;;
esac  *) echo "i know nothing" ;;
```