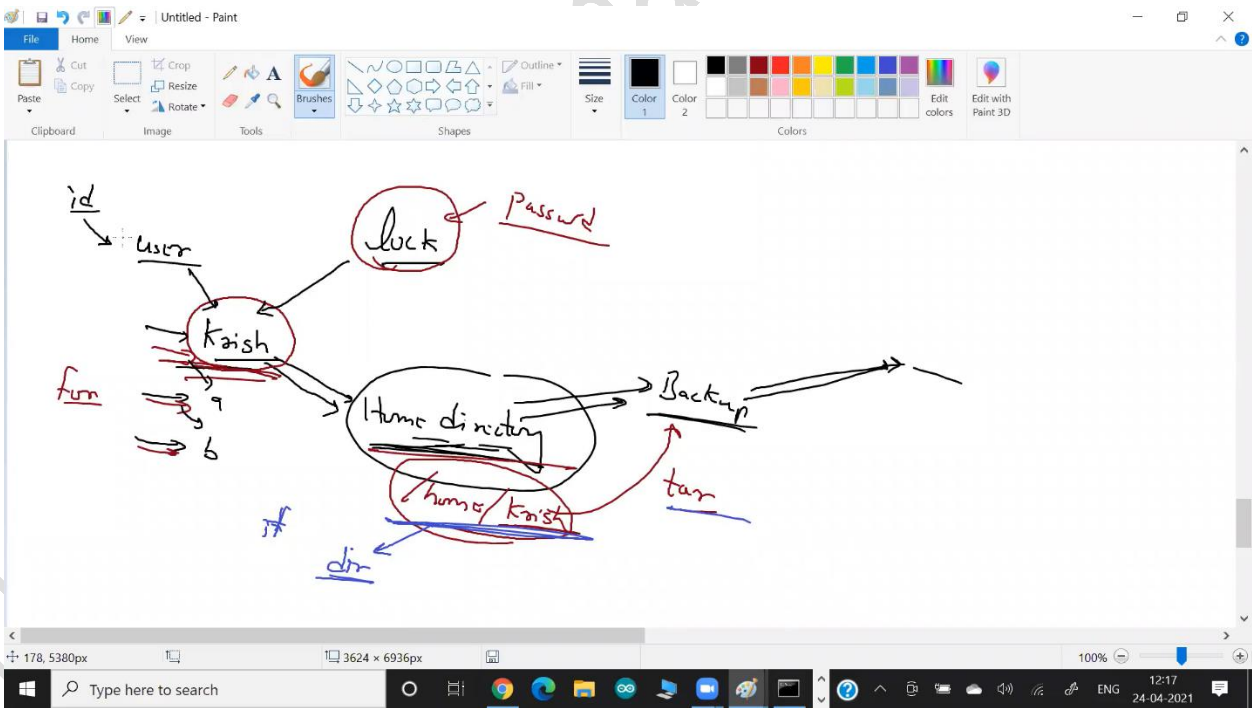
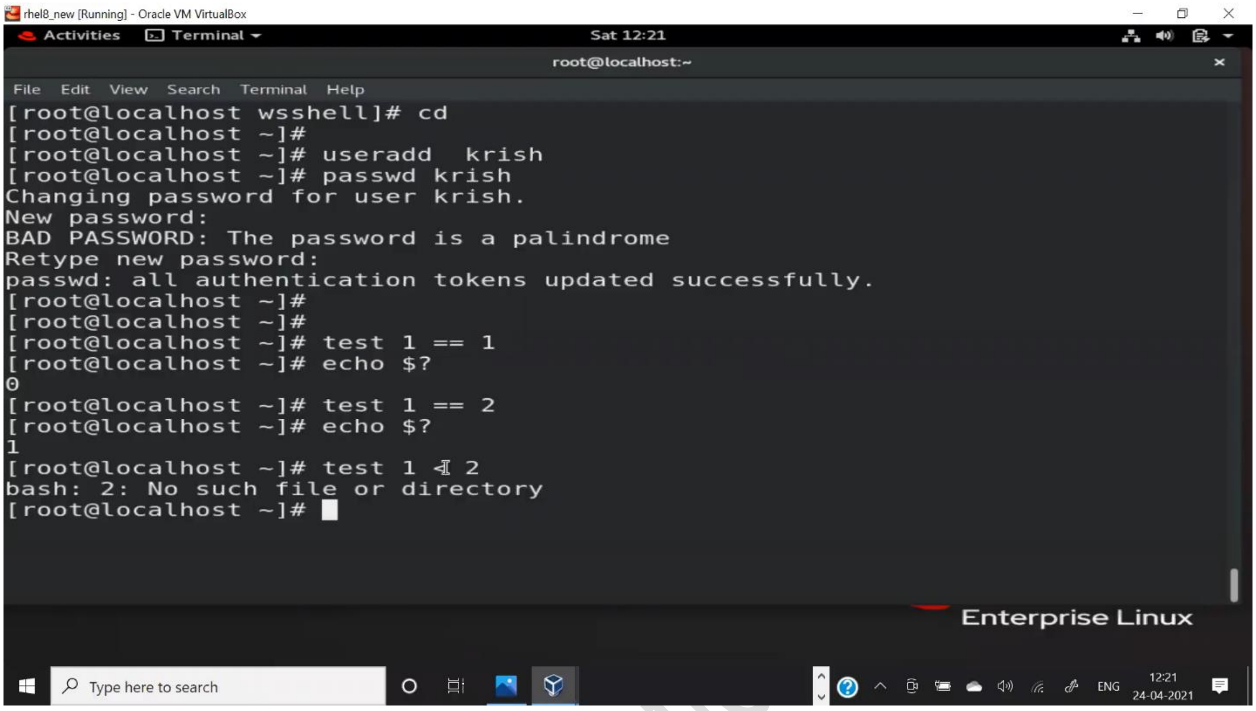
Shell and Shell Scripting Session No.1.4

- Let’s say there is a user named Krish and he is doing some malicious activities, we need to analyze his account for this, so we will first lock his account and do the backup of the entire data we have in the home directory. And send this data to our team for analysis.
- The home directory of every user is in the /home directory. In Linux, we can lock any account with the help of the “password” command. For this thing, we may have more than one user, so again we will use the “FOR” loop here. For backup of the home directory, there is a famous command in Linux that is the “tar” command.
- Before we do a backup, we first check whether this user exists or not, and for this we can use the if condition and the “id” command

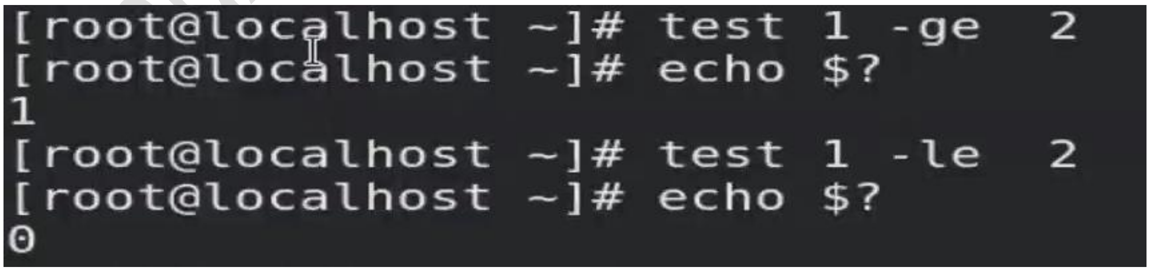


- The “If” condition has its syntax. We will give “if” a condition that if this condition is true then do this or else do this, but how to test the condition in Linux? So for this, we have a command `test`



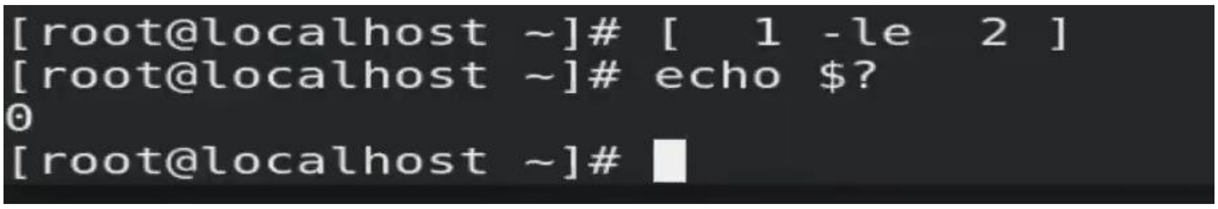
```
[root@localhost wsshell]# cd
[root@localhost ~]#
[root@localhost ~]# useradd  krish
[root@localhost ~]# passwd krish
Changing password for user krish.
New password:
BAD PASSWORD: The password is a palindrome
Retype new password:
passwd: all authentication tokens updated successfully.
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# test 1 == 1
[root@localhost ~]# echo $?
0
[root@localhost ~]# test 1 == 2
[root@localhost ~]# echo $?
1
[root@localhost ~]# test 1 < 2
bash: 2: No such file or directory
[root@localhost ~]#
```

- But greater than or less than symbol while giving condition does not work in the shell because < or > symbol has a different meaning here.
- (<) is the input redirection symbol and (>) is the output redirection symbol.
- That's why in the Linux or shell, we don't use these symbols for testing conditions. Instead, we have different symbols:
  - -eq = equals to
  - -ne = not equal to
  - -gt = greater than
  - -lt = less than
  - -le = less than equals to
  - -ge = greater than equals to



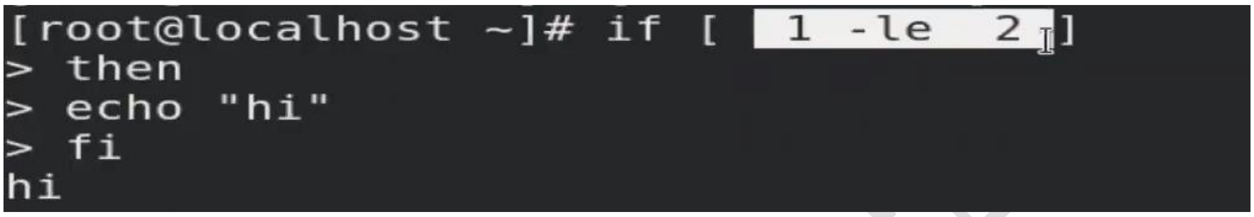
```
[root@localhost ~]# test 1 -ge  2
[root@localhost ~]# echo $?
1
[root@localhost ~]# test 1 -le  2
[root@localhost ~]# echo $?
0
```

- Instead of test keywords we can use square brackets also



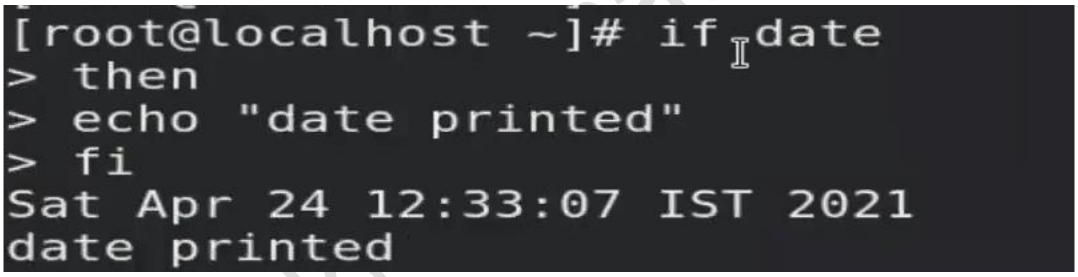
```
[root@localhost ~]# [  1 -le  2 ]
[root@localhost ~]# echo $?
0
[root@localhost ~]#
```

- The syntax for the "if" condition is



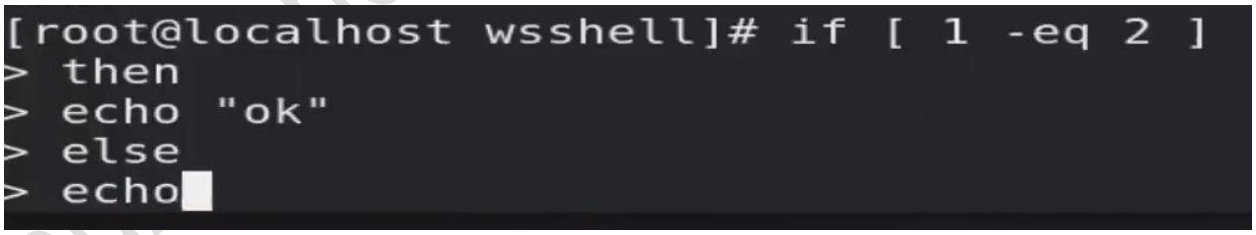
```
[root@localhost ~]# if [  1 -le  2 ]
> then
> echo "hi"
> fi
hi
```

- The "if" condition here is designed in such a way that it doesn't require Boolean, it needs status code for running its block of code.
- So, when the condition we give returns the status code 0, then "if" will run its block of code. So instead of giving conditions like these, we can also mention some commands here because every command returns the exit code



```
[root@localhost ~]# if date
> then
> echo "date printed"
> fi
Sat Apr 24 12:33:07 IST 2021
date printed
```

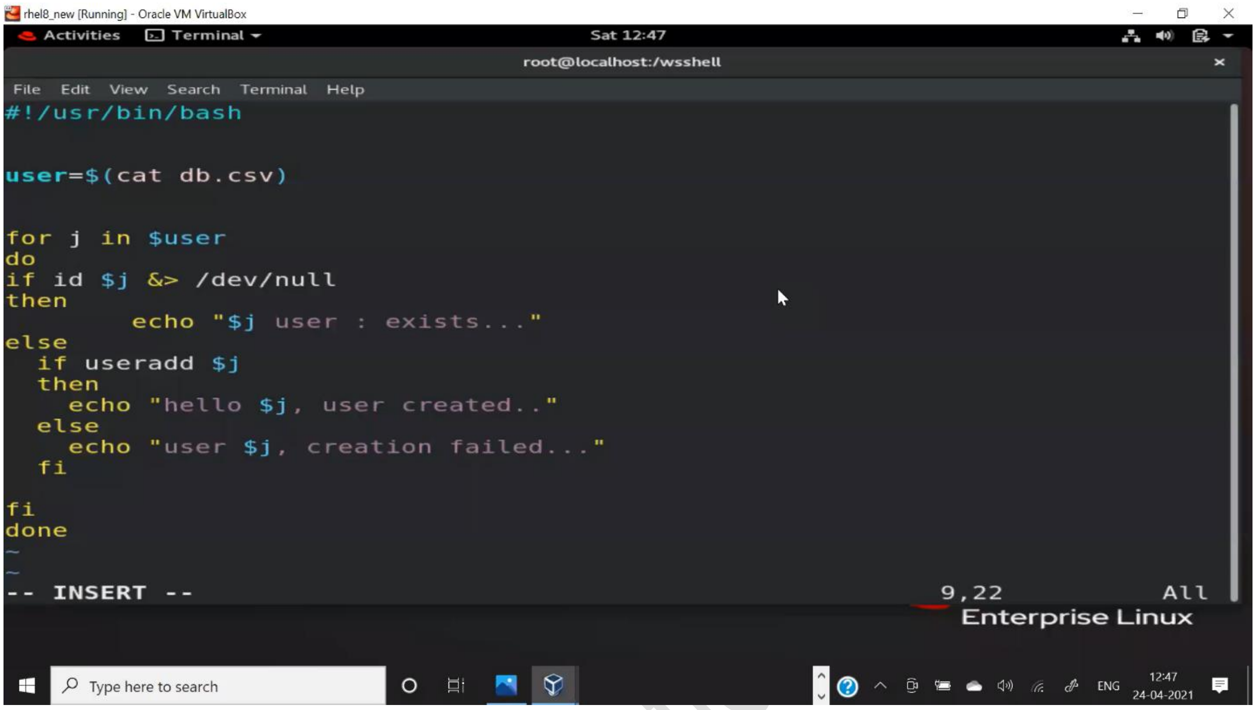
- The complete syntax of the "if" condition including both "then & else" is



```
[root@localhost wsshell]# if [ 1 -eq 2 ]
> then
> echo "ok"
> else
> echo
```

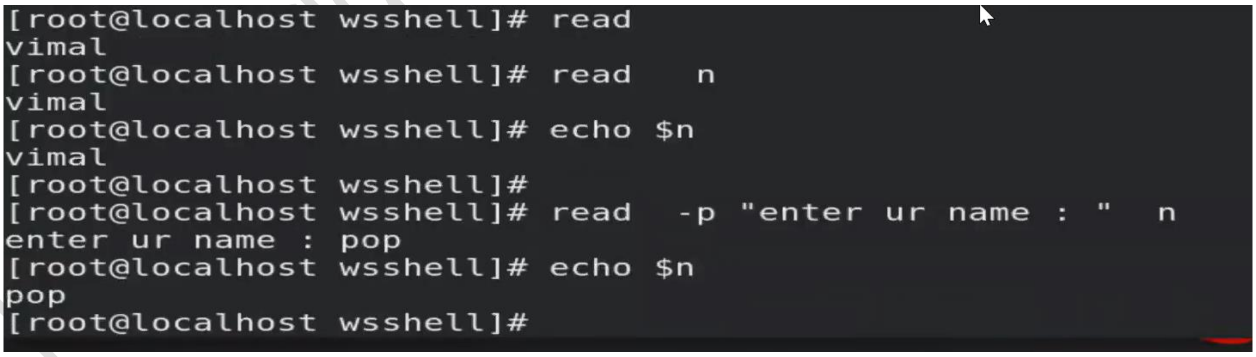
- What is a nested condition? -> When we use a condition inside one condition then it is called a nested condition.

- Now modify our previous code



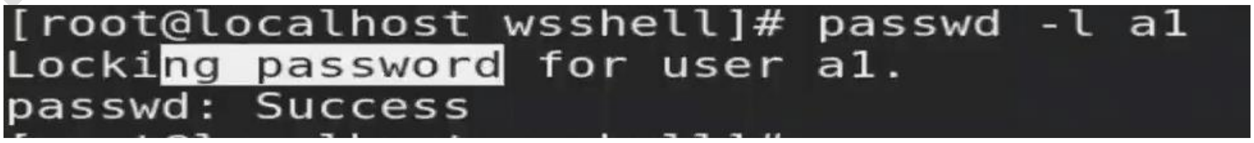
```
#!/usr/bin/bash

user=$(cat db.csv)

for j in $user
do
if id $j &> /dev/null
then
        echo "$j user : exists..."
else
  if useradd $j
  then
    echo "hello $j, user created.."
  else
    echo "user $j, creation failed..."
  fi

fi
done
~
~
-- INSERT --                                                  9,22          All
```

- Here we use nested "IF" condition. First, the user will be checked by the id command and if it is available then it will return the string "exists" and if it does not exist then it will go further and run the "else" statement. Also, we use the if condition again because if the useradd command fails due to some other reason then it will return this string.
- "Read" is one of the keywords of the shell that gives you a prompt to enter your data and store it in a variable on the fly



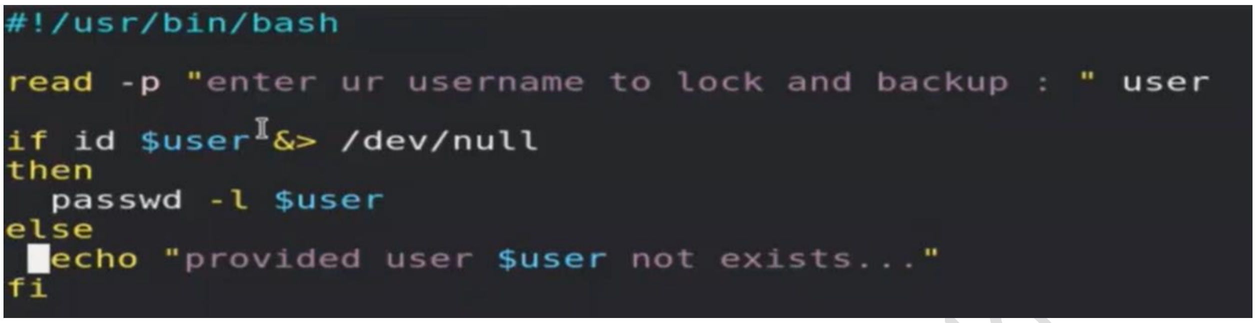
```
[root@localhost wsshell]# read
vimal
[root@localhost wsshell]# read   n
vimal
[root@localhost wsshell]# echo $n
vimal
[root@localhost wsshell]#
[root@localhost wsshell]# read  -p "enter ur name : "  n
enter ur name : pop
[root@localhost wsshell]# echo $n
pop
[root@localhost wsshell]#
```

- For locking the user, we use the password command:



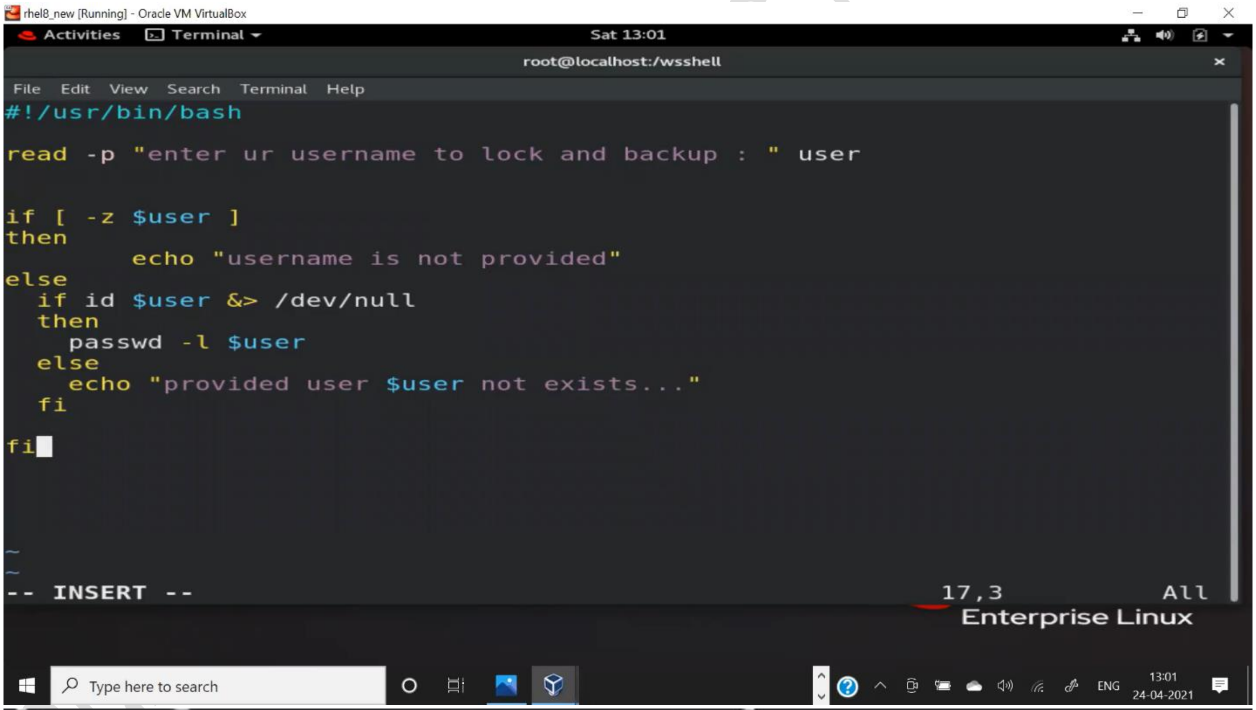
```
[root@localhost wsshell]# passwd -l a1
Locking password for user a1.
passwd: Success
```

- Now let's create the script for our use-case of locking and backing up the command

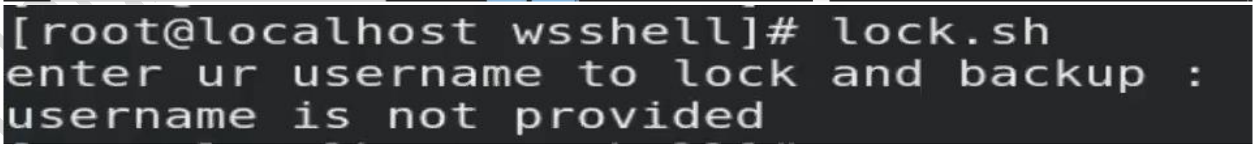


```
#!/usr/bin/bash

read -p "enter ur username to lock and backup : " user

if id $user &> /dev/null
then
  passwd -l $user
else
  echo "provided user $user not exists..."
fi
```

- This is how we can start our code for locking the user we use the if condition to check whether the user exists or not.
- It is possible that the user may not provide any username when it is asking for it. In this case, we first check whether the variable "user" has some data or not. And for this, we use the `-z option`

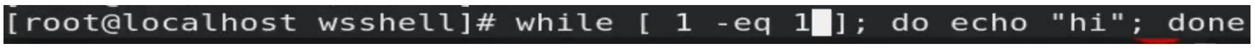


```
#!/usr/bin/bash

read -p "enter ur username to lock and backup : " user


if [ -z $user ]
then
        echo "username is not provided"
else
  if id $user &> /dev/null
  then
    passwd -l $user
  else
    echo "provided user $user not exists..."
  fi

fi
~
~
-- INSERT --                                                    17,3          All
```



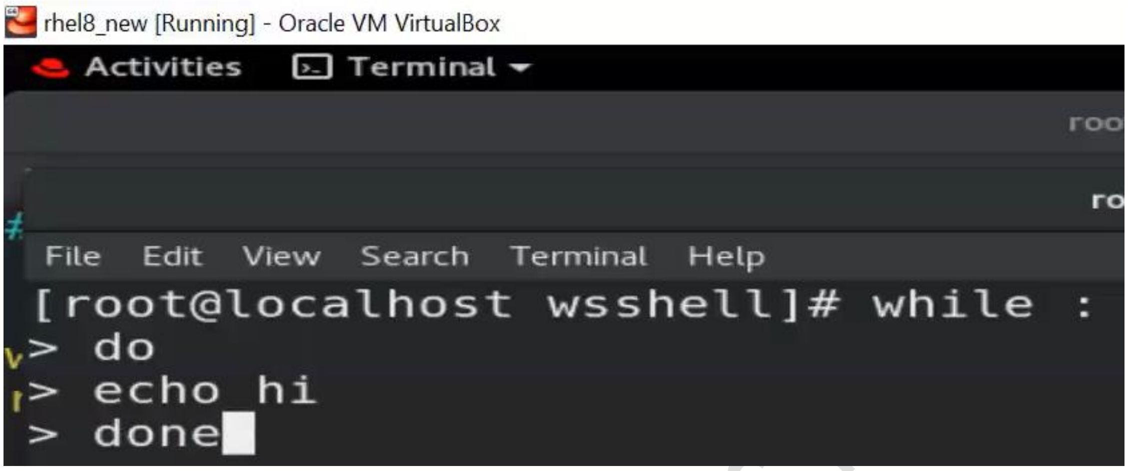
```
[root@localhost wsshell]# lock.sh
enter ur username to lock and backup :
username is not provided
```

- Here whenever no input has been provided, the script ends and terminates and we have to run it again. So, we want that on failure it will again prompt us for the username. For this use case, we use the "while" loop. While the loop will again and again run the code until it is asked to stop. It is also known as an infinity loop



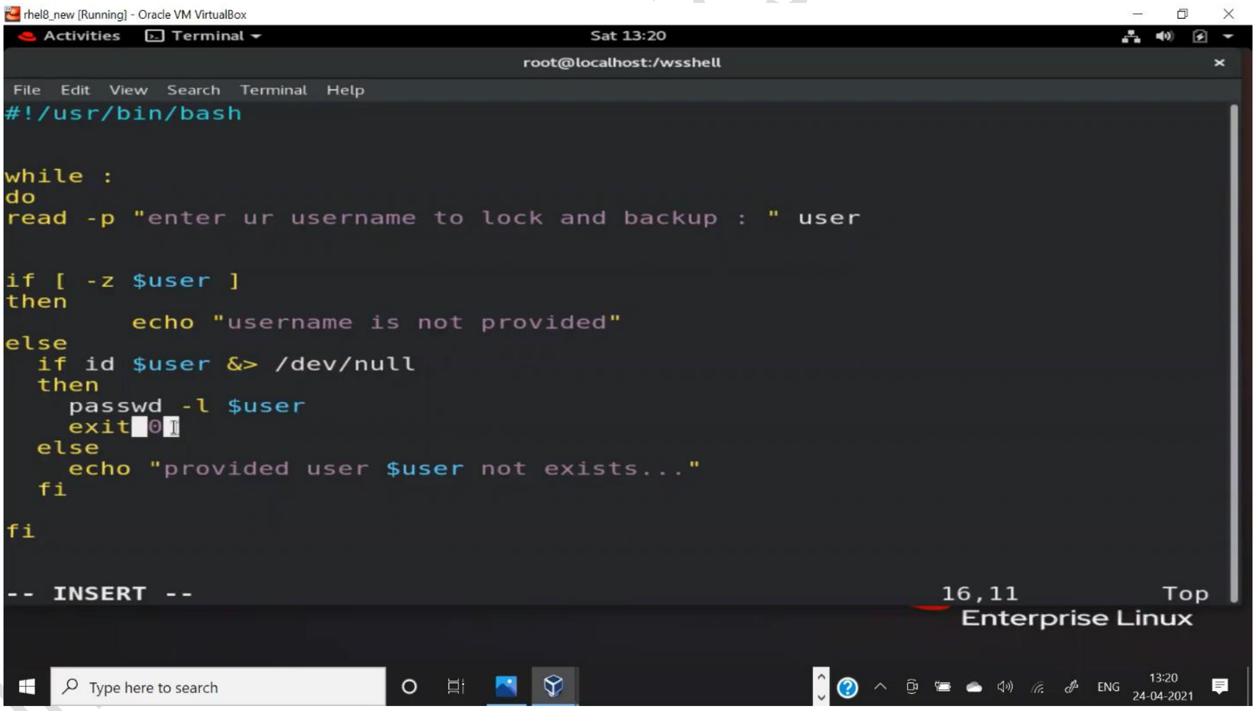
```
[root@localhost wsshell]# while [ 1 -eq 1 ]; do echo "hi"; done
```

- This is the syntax of the while loop, and since the condition we give here is true, it will run the code infinite times until we manually terminate the loop by “ctrl+c”
- But if we don’t give any condition, that means it is by default True


rhel8_new [Running] - Oracle VM VirtualBox
Activities
Terminal
File Edit View Search Terminal Help

```
[root@localhost wsshell]# while :
> do
> echo hi
> done
```

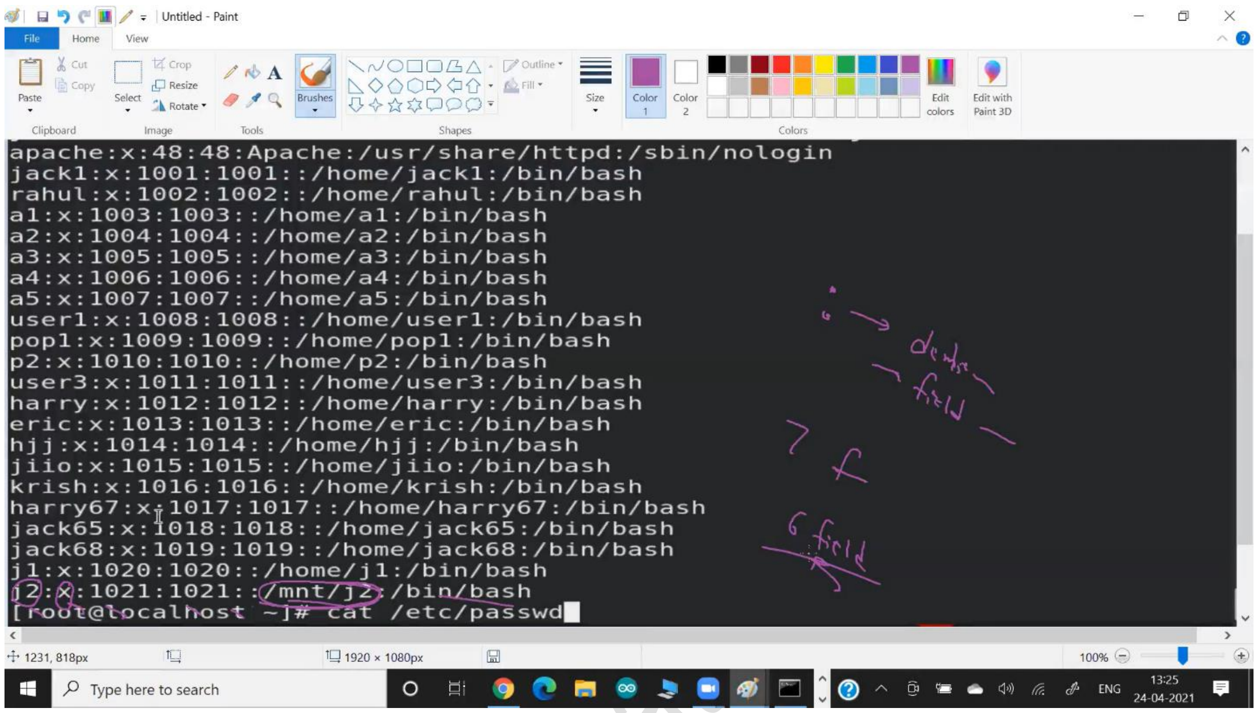
- Now we will add this while looping in our script



```
#!/usr/bin/bash

while :
do
read -p "enter ur username to lock and backup : " user

if [ -z $user ]
then
        echo "username is not provided"
else
  if id $user &> /dev/null
  then
    passwd -l $user
    exit 0
  else
    echo "provided user $user not exists..."
  fi

fi
-- INSERT --                                          16,11          Top
```

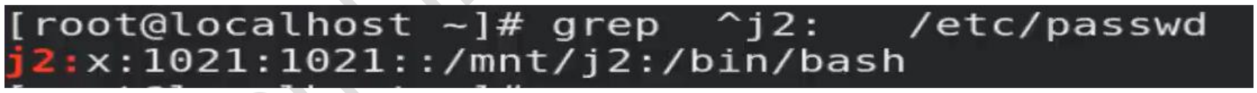
- Now we successfully create the script for locking the user. Now our next task is to back that user.
- So for backup of the data we have to find out the home directory of that user and it is not fixed that the home directory is always in /home directory.

- So to find the home directory of a particular user we take the help from "/etc/passwd" file. Only this file contains the entire info about a user



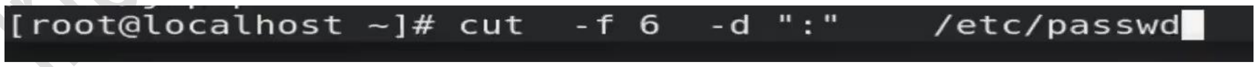
```
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
jack1:x:1001:1001::/home/jack1:/bin/bash
rahul:x:1002:1002::/home/rahul:/bin/bash
a1:x:1003:1003::/home/a1:/bin/bash
a2:x:1004:1004::/home/a2:/bin/bash
a3:x:1005:1005::/home/a3:/bin/bash
a4:x:1006:1006::/home/a4:/bin/bash
a5:x:1007:1007::/home/a5:/bin/bash
user1:x:1008:1008::/home/user1:/bin/bash
pop1:x:1009:1009::/home/pop1:/bin/bash
p2:x:1010:1010::/home/p2:/bin/bash
user3:x:1011:1011::/home/user3:/bin/bash
harry:x:1012:1012::/home/harry:/bin/bash
eric:x:1013:1013::/home/eric:/bin/bash
hjj:x:1014:1014::/home/hjj:/bin/bash
jiio:x:1015:1015::/home/jiio:/bin/bash
krish:x:1016:1016::/home/krish:/bin/bash
harry67:x:1017:1017::/home/harry67:/bin/bash
jack65:x:1018:1018::/home/jack65:/bin/bash
jack68:x:1019:1019::/home/jack68:/bin/bash
j1:x:1020:1020::/home/j1:/bin/bash
j2:x:1021:1021::/mnt/j2:/bin/bash
[root@localhost ~]# cat /etc/passwd
```

- This file has a total of 7 fields and each field is separated by (:) also known as delimiter and the home directory is the 6th field.
- And "grep" command helps us to find out the user we are searching for inside the file. Grep is the command used to search for a particular pattern inside a file

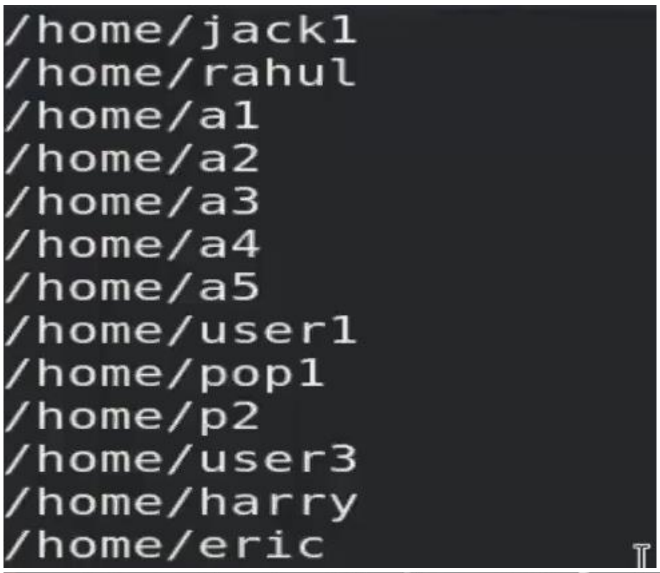


```
[root@localhost ~]# grep  ^j2:   /etc/passwd
j2:x:1021:1021::/mnt/j2:/bin/bash
```

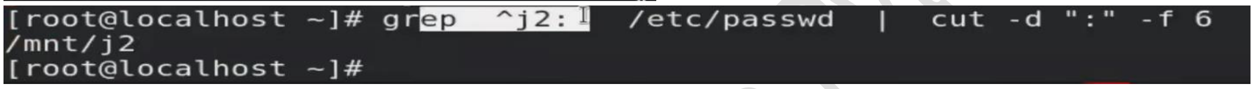
- ^ symbol will search for the given word at the start of the line and that word should end with the (:) symbol. These symbols come under the topic regex.
- But still we want the 6th field in this line also so for this we will use the cut command



```
[root@localhost ~]# cut  -f 6  -d ":"    /etc/passwd
```

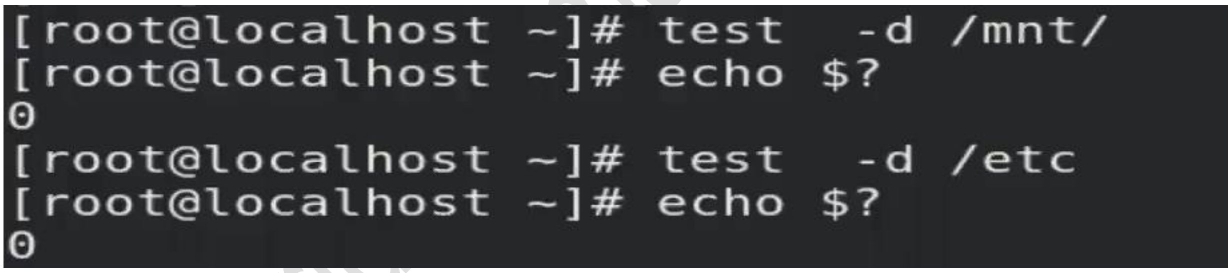


```
/home/jack1
/home/rahul
/home/a1
/home/a2
/home/a3
/home/a4
/home/a5
/home/user1
/home/pop1
/home/p2
/home/user3
/home/harry
/home/eric
```



```
[root@localhost ~]# grep  ^j2:  /etc/passwd  |  cut -d ":" -f 6
/mnt/j2
[root@localhost ~]#
```

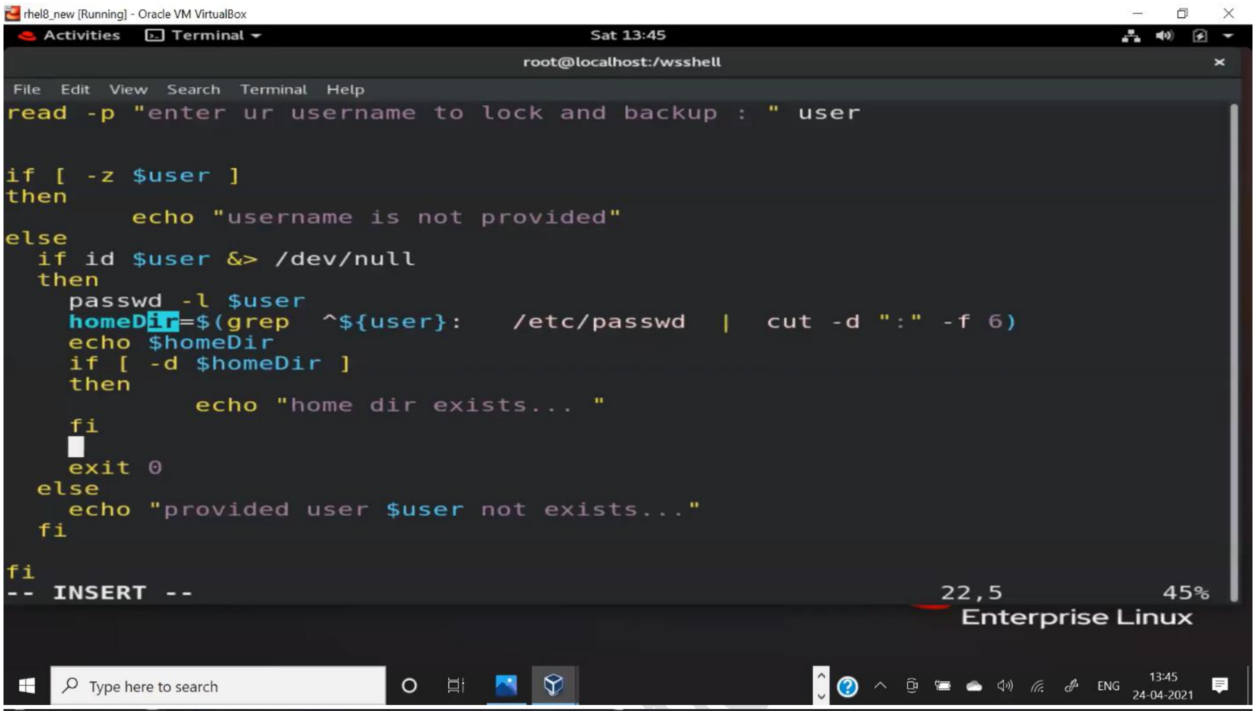
- But this doesn't mean that the home directory always exists, so for this we have to first check whether the file exists or not. And for this, we can use the "test" command or square brackets [ ], but the option we have to use is "-d"



```
[root@localhost ~]# test  -d /mnt/
[root@localhost ~]# echo $?
0
[root@localhost ~]# test  -d /etc
[root@localhost ~]# echo $?
0
```

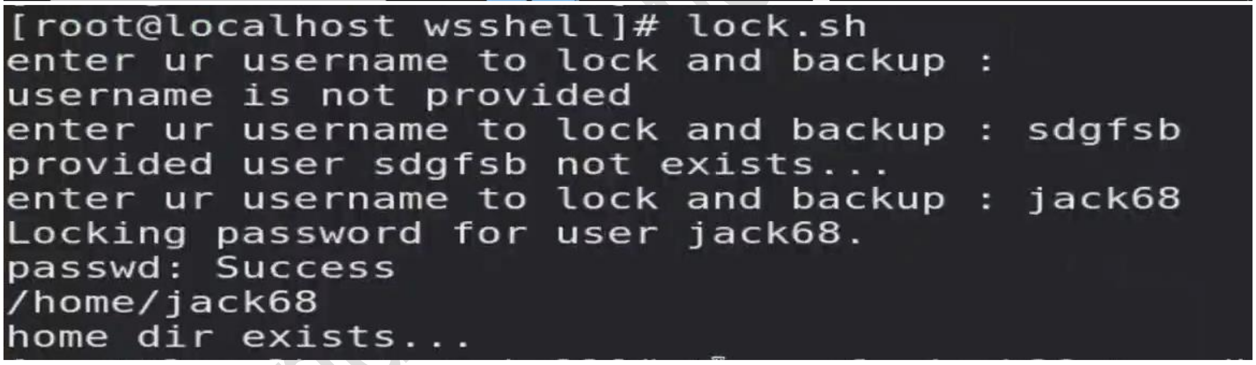
- There are more options like this:
    - -e = file exist
    - -x = file is executable
    - -s = file exists and is not empty
    - -r = file is readable
    - -w = file is writable
    - -f = file exists and is a regular file

- Now we will put all this info into our script



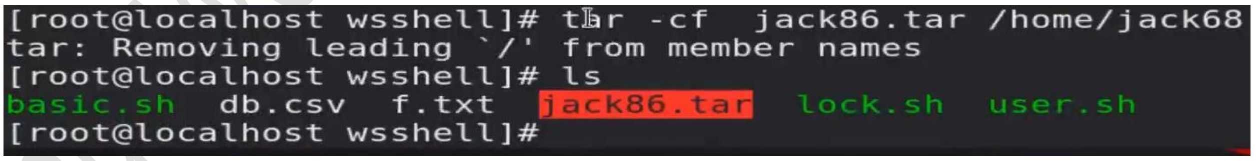
rhel8_new [Running] - Oracle VM VirtualBox
Activities
Terminal
Sat 13:45
root@localhost:/wsshell
File Edit View Search Terminal Help

```
read -p "enter ur username to lock and backup : " user

if [ -z $user ]
then
        echo "username is not provided"
else
  if id $user &> /dev/null
  then
    passwd -l $user
    homeDir=$(grep  ^${user}:   /etc/passwd  |  cut -d ":" -f 6)
    echo $homeDir
    if [ -d $homeDir ]
    then
            echo "home dir exists... "
    fi

    exit 0
  else
    echo "provided user $user not exists..."
  fi

fi
-- INSERT --                                                    22,5          45%
```



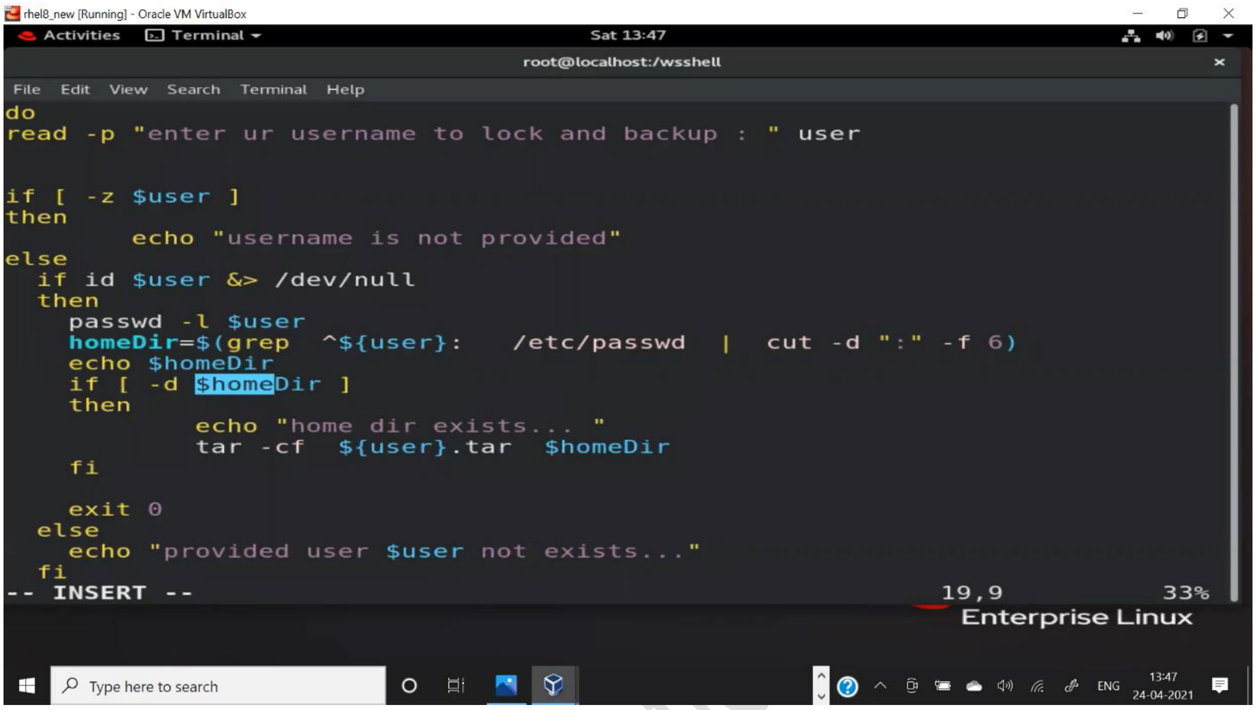
```
[root@localhost wsshell]# lock.sh
enter ur username to lock and backup :
username is not provided
enter ur username to lock and backup : sdgfsb
provided user sdgfsb not exists...
enter ur username to lock and backup : jack68
Locking password for user jack68.
passwd: Success
/home/jack68
home dir exists...
```

- Now our next and final step is to create the backup of that directory and for this, we have the command “tar”



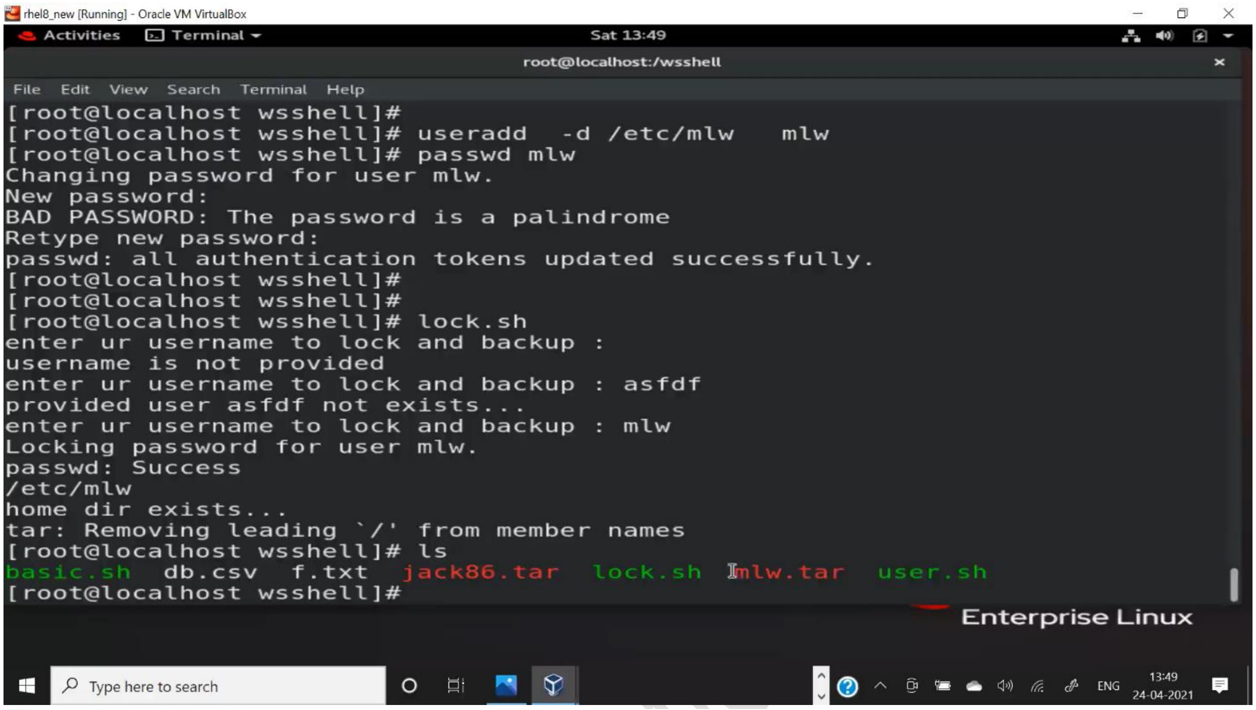
```
[root@localhost wsshell]# tar -cf  jack86.tar /home/jack68
tar: Removing leading `/' from member names
[root@localhost wsshell]# ls
basic.sh   db.csv   f.txt   jack86.tar   lock.sh   user.sh
[root@localhost wsshell]#
```

- Modifying our script

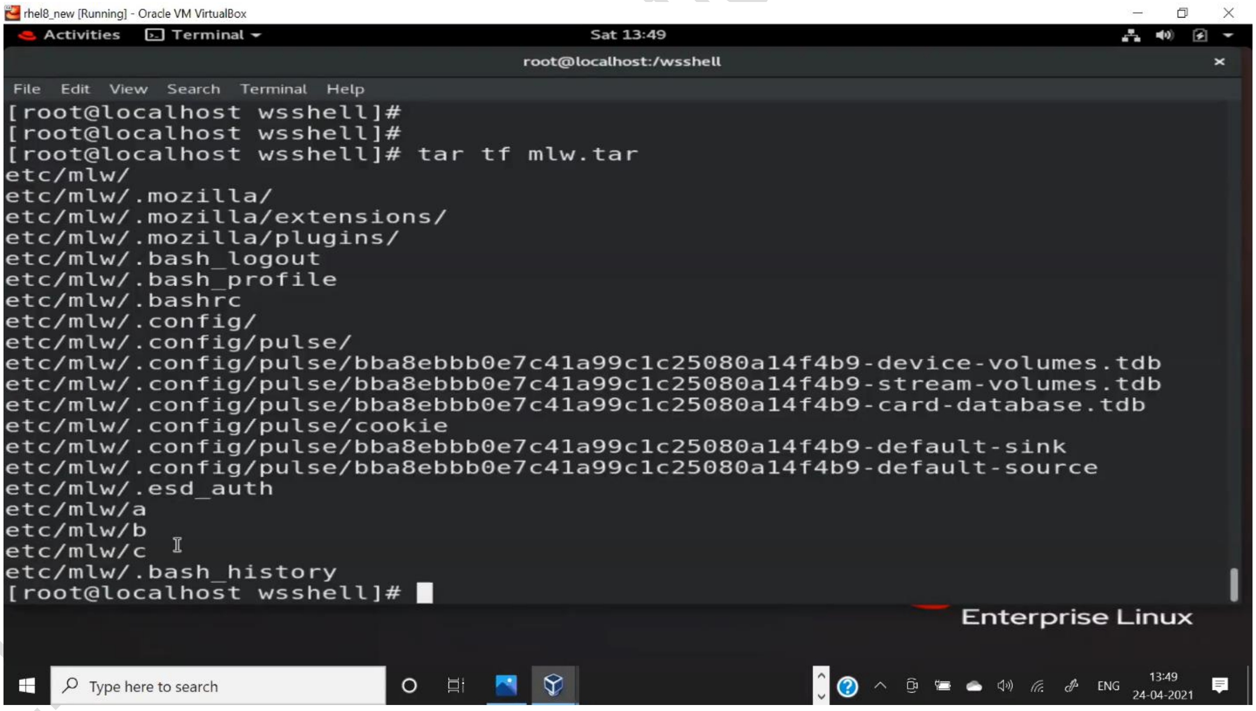


```
do
read -p "enter ur username to lock and backup : " user

if [ -z $user ]
then
        echo "username is not provided"
else
  if id $user &> /dev/null
  then
    passwd -l $user
    homeDir=$(grep  ^${user}:   /etc/passwd  |  cut -d ":" -f 6)
    echo $homeDir
    if [ -d $homeDir ]
    then
            echo "home dir exists... "
            tar -cf  ${user}.tar  $homeDir
    fi

    exit 0
  else
    echo "provided user $user not exists..."
  fi
-- INSERT --                                                  19,9          33%
```

- Now our script is ready and we can verify it



```
[root@localhost wsshell]#
[root@localhost wsshell]# useradd  -d /etc/mlw   mlw
[root@localhost wsshell]# passwd mlw
Changing password for user mlw.
New password:
BAD PASSWORD: The password is a palindrome
Retype new password:
passwd: all authentication tokens updated successfully.
[root@localhost wsshell]#
[root@localhost wsshell]#
[root@localhost wsshell]# lock.sh
enter ur username to lock and backup :
username is not provided
enter ur username to lock and backup : asfdf
provided user asfdf not exists...
enter ur username to lock and backup : mlw
Locking password for user mlw.
passwd: Success
/etc/mlw
home dir exists...
tar: Removing leading `/' from member names
[root@localhost wsshell]# ls
basic.sh  db.csv  f.txt  jack86.tar  lock.sh  mlw.tar  user.sh
[root@localhost wsshell]#
```



```
[root@localhost wsshell]#
[root@localhost wsshell]#
[root@localhost wsshell]# tar tf mlw.tar
etc/mlw/
etc/mlw/.mozilla/
etc/mlw/.mozilla/extensions/
etc/mlw/.mozilla/plugins/
etc/mlw/.bash_logout
etc/mlw/.bash_profile
etc/mlw/.bashrc
etc/mlw/.config/
etc/mlw/.config/pulse/
etc/mlw/.config/pulse/bba8ebbb0e7c41a99c1c25080a14f4b9-device-volumes.tdb
etc/mlw/.config/pulse/bba8ebbb0e7c41a99c1c25080a14f4b9-stream-volumes.tdb
etc/mlw/.config/pulse/bba8ebbb0e7c41a99c1c25080a14f4b9-card-database.tdb
etc/mlw/.config/pulse/cookie
etc/mlw/.config/pulse/bba8ebbb0e7c41a99c1c25080a14f4b9-default-sink
etc/mlw/.config/pulse/bba8ebbb0e7c41a99c1c25080a14f4b9-default-source
etc/mlw/.esd_auth
etc/mlw/a
etc/mlw/b
etc/mlw/c
etc/mlw/.bash_history
[root@localhost wsshell]#
```