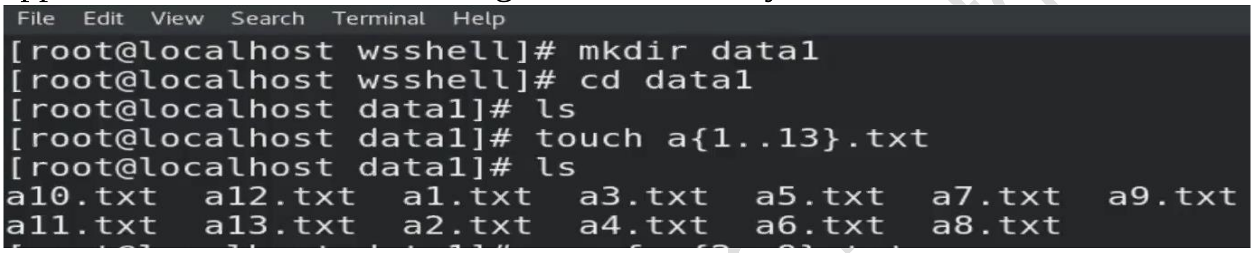
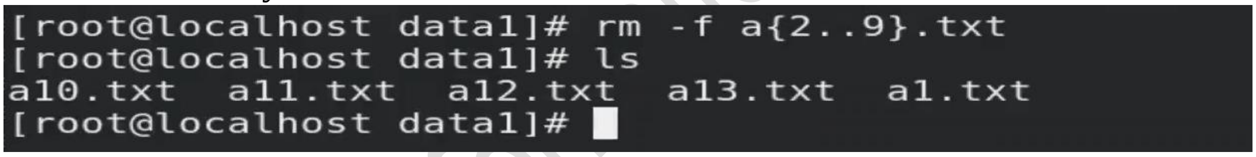
Shell and Shell Scripting Session No.2.2

- If my requirement is I want to launch to create some files in a specific order like A1, A2, A3, A4, A5….. and so on. So, we can use a different approach here rather than creating these files one by one.



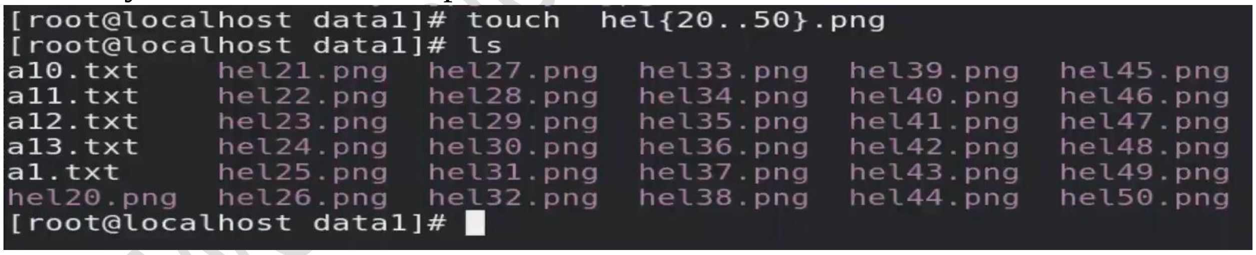
```
[root@localhost wsshell]# mkdir data1
[root@localhost wsshell]# cd data1
[root@localhost data1]# ls
[root@localhost data1]# touch a{1..13}.txt
[root@localhost data1]# ls
a10.txt  a12.txt  a1.txt  a3.txt  a5.txt  a7.txt  a9.txt
a11.txt  a13.txt  a2.txt  a4.txt  a6.txt  a8.txt
```

- This will create 13 files in order here.
- And the same way we can delete them also.



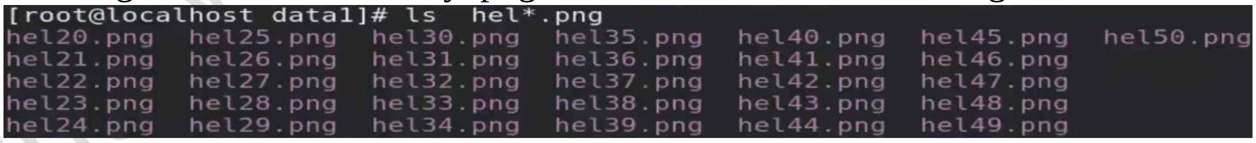
```
[root@localhost data1]# rm -f a{2..9}.txt
[root@localhost data1]# ls
a10.txt  a11.txt  a12.txt  a13.txt  a1.txt
[root@localhost data1]# 
```

- Similarly we can create multiple files like



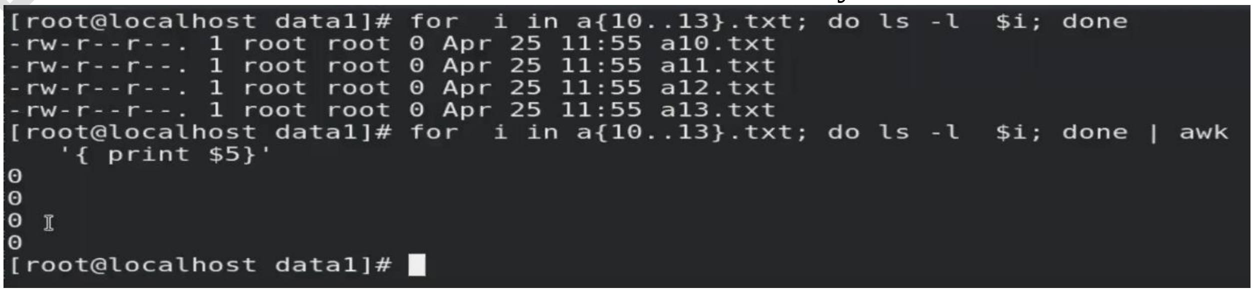
```
[root@localhost data1]# touch  hel{20..50}.png
[root@localhost data1]# ls
a10.txt     hel21.png  hel27.png  hel33.png  hel39.png  hel45.png
a11.txt     hel22.png  hel28.png  hel34.png  hel40.png  hel46.png
a12.txt     hel23.png  hel29.png  hel35.png  hel41.png  hel47.png
a13.txt     hel24.png  hel30.png  hel36.png  hel42.png  hel48.png
a1.txt      hel25.png  hel31.png  hel37.png  hel43.png  hel49.png
hel20.png   hel26.png  hel32.png  hel38.png  hel44.png  hel50.png
[root@localhost data1]# 
```

- And to get these files like only .png files, we can use the below regex.



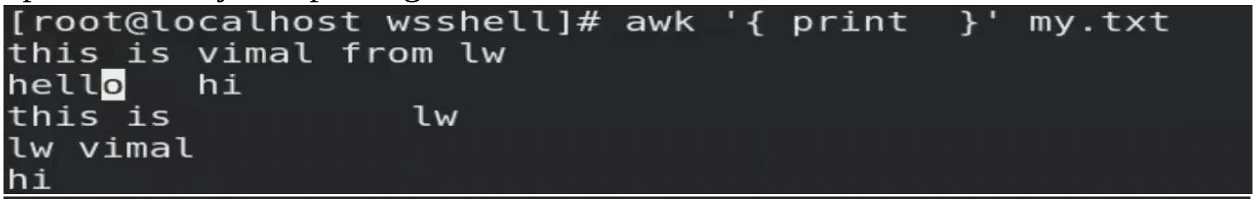
```
[root@localhost data1]# ls  hel*.png
hel20.png  hel25.png  hel30.png  hel35.png  hel40.png  hel45.png  hel50.png
hel21.png  hel26.png  hel31.png  hel36.png  hel41.png  hel46.png
hel22.png  hel27.png  hel32.png  hel37.png  hel42.png  hel47.png
hel23.png  hel28.png  hel33.png  hel38.png  hel43.png  hel48.png
hel24.png  hel29.png  hel34.png  hel39.png  hel44.png  hel49.png
```

- Similarly we can get a long list of the files and do further operations like what is the size of the data stored in those files and many more.

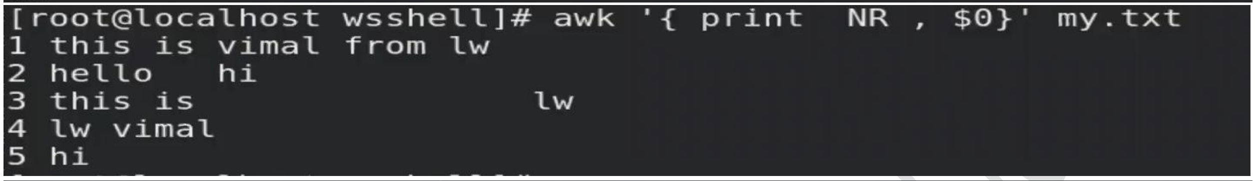


```
[root@localhost data1]# for  i in a{10..13}.txt; do ls -l  $i; done
-rw-r--r--. 1 root root 0 Apr 25 11:55 a10.txt
-rw-r--r--. 1 root root 0 Apr 25 11:55 a11.txt
-rw-r--r--. 1 root root 0 Apr 25 11:55 a12.txt
-rw-r--r--. 1 root root 0 Apr 25 11:55 a13.txt
[root@localhost data1]# for  i in a{10..13}.txt; do ls -l  $i; done | awk
   '{ print $5}'
0
0
0
0
[root@localhost data1]# 
```

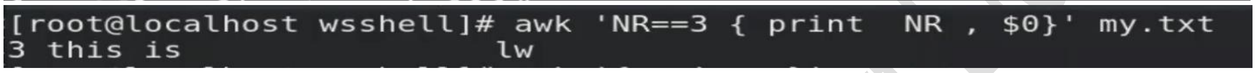
- Now we can also allot a number to each line that makes our future operations easy like passing conditions on them.



```
[root@localhost wsshell]# awk '{ print  }' my.txt
this is vimal from lw
hello   hi
this is          lw
lw vimal
hi
```

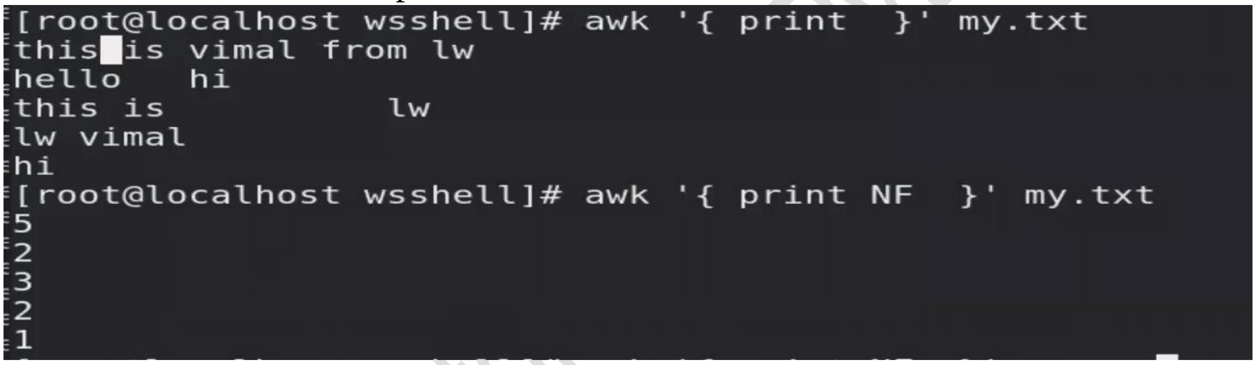


```
[root@localhost wsshell]# awk '{ print  NR , $0}' my.txt
1 this is vimal from lw
2 hello   hi
3 this is              lw
4 lw vimal
5 hi
```



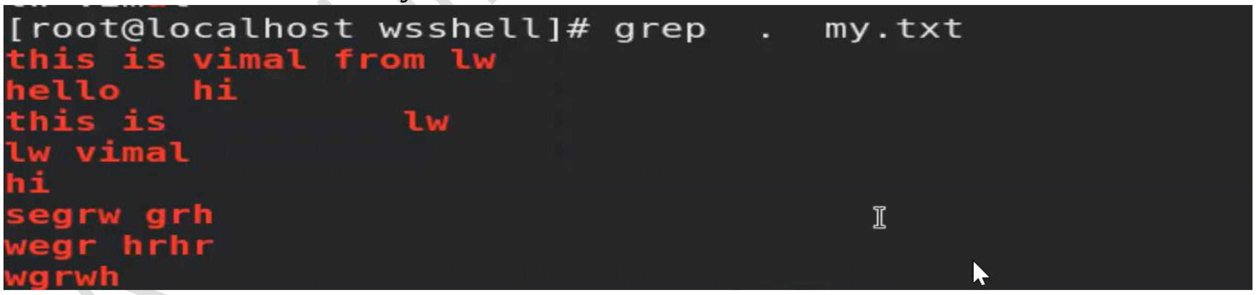
```
[root@localhost wsshell]# awk 'NR==3 { print  NR , $0}' my.txt
3 this is                   lw
```

- The below command will print the number of fields each line has,



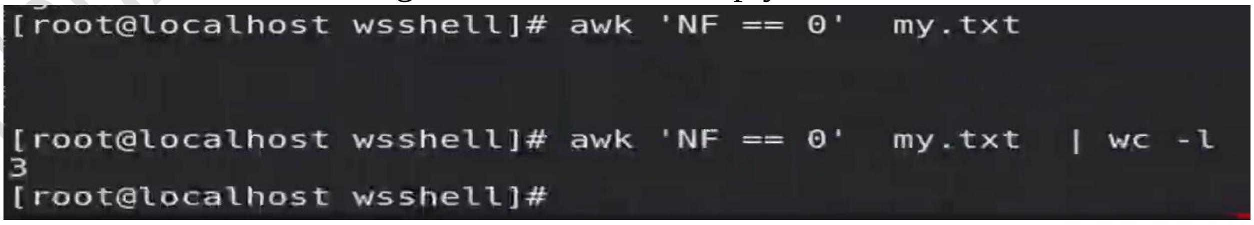
```
[root@localhost wsshell]# awk '{ print  }' my.txt
this is vimal from lw
hello   hi
this is          lw
lw vimal
hi
[root@localhost wsshell]# awk '{ print NF  }' my.txt
5
2
3
2
1
```

- The below command will remove all the empty lines from the file and return the data in continuity.



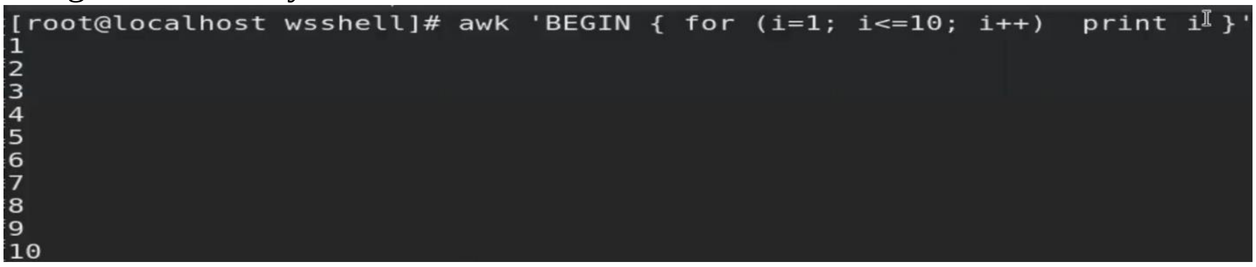
```
[root@localhost wsshell]# grep  .  my.txt
this is vimal from lw
hello   hi
this is          lw
lw vimal
hi
segrw grh
wegr hrhr
wgrwh
```

- The below command will give the number of empty lines from the file.



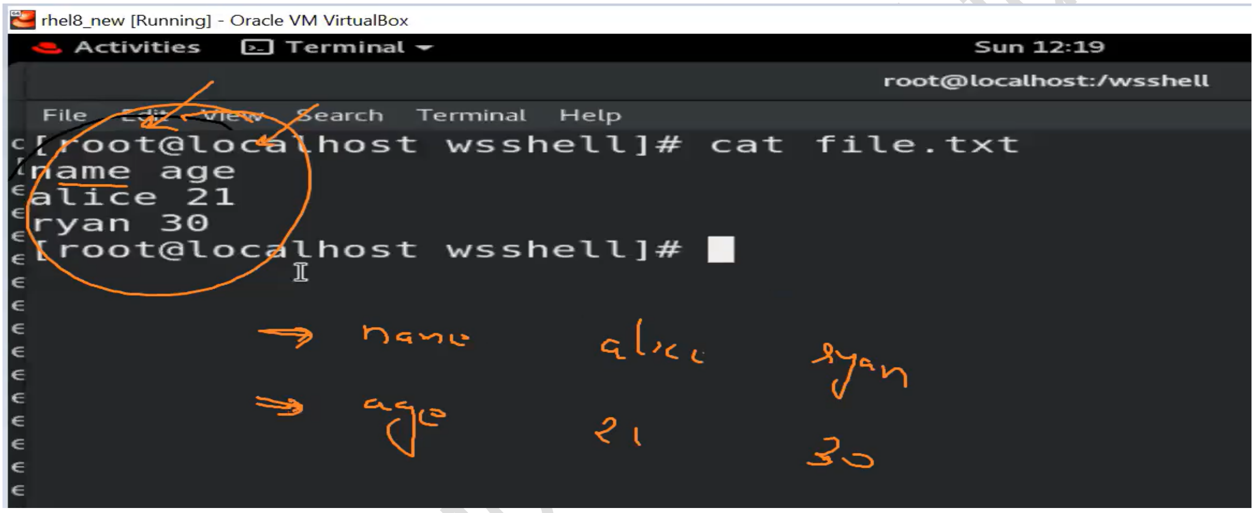
```
[root@localhost wsshell]# awk 'NF == 0'  my.txt



[root@localhost wsshell]# awk 'NF == 0'  my.txt  | wc -l
3
[root@localhost wsshell]#
```

- If we don't have any file as an argument for the "awk" command, then, we can use the "BEGIN" keyword to tell awk that you have to start from this given code only.



```
[root@localhost wsshell]# awk 'BEGIN { for (i=1; i<=10; i++)  print i }'
1
2
3
4
5
6
7
8
9
10
```

- Like we have the use-case where we have a database file in which there are 2 columns, one is name and the other is age. Now we want to convert both columns into rows.



rhel8_new [Running] - Oracle VM VirtualBox

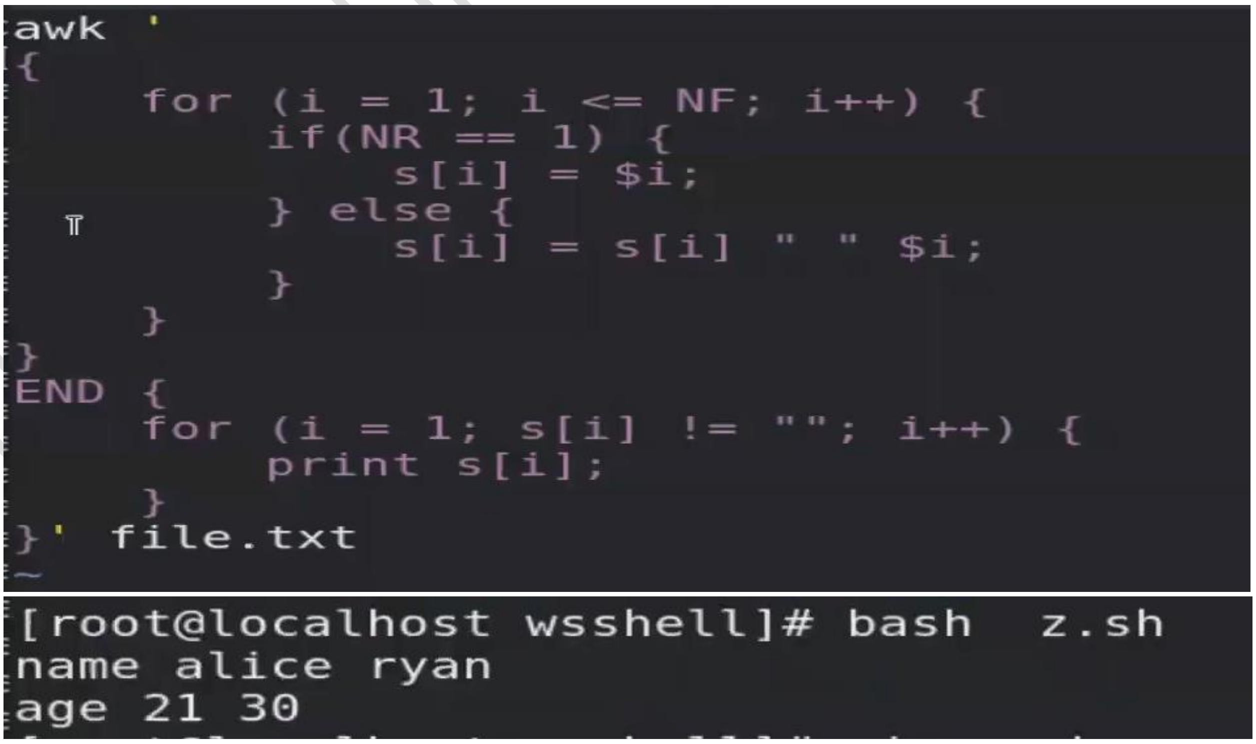
Activities   Terminal ▾   Sun 12:19

root@localhost:/wsshell

File Edit View Search Terminal Help

```
[root@localhost wsshell]# cat file.txt
name age
alice 21
ryan 30
[root@localhost wsshell]#
```

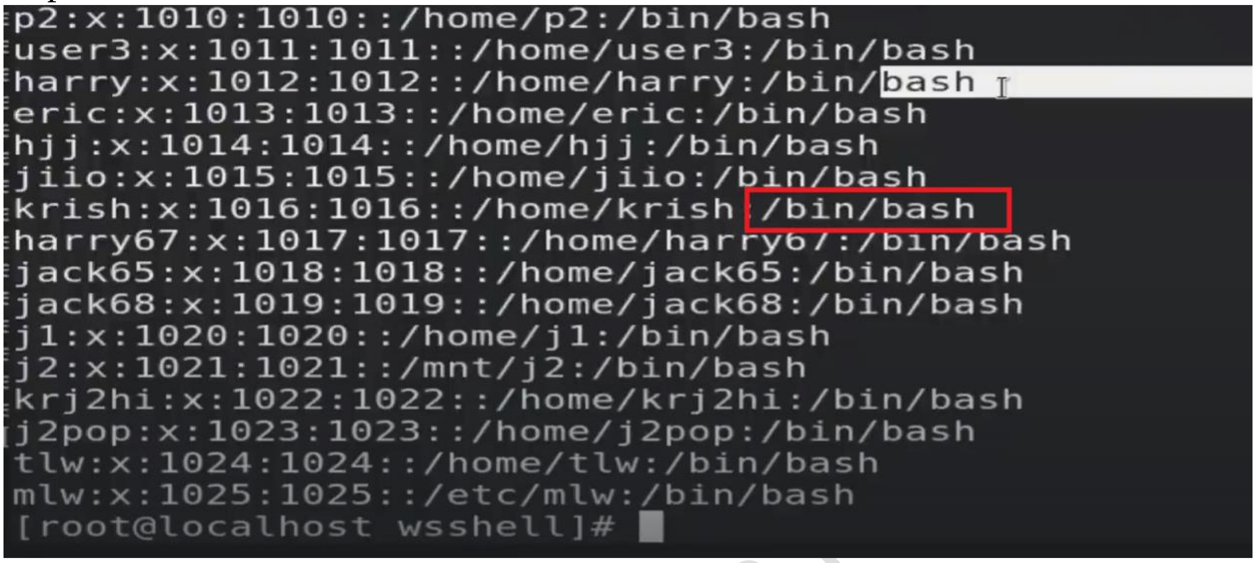
- There is a script that is created with the awk command which will convert these columns into rows.



```
awk '
{
        for (i = 1; i <= NF; i++) {
                if(NR == 1) {
                        s[i] = $i;
                } else {
                        s[i] = s[i] " " $i;
                }
        }
}
END {
        for (i = 1; s[i] != ""; i++) {
                print s[i];
        }
}' file.txt
~
```

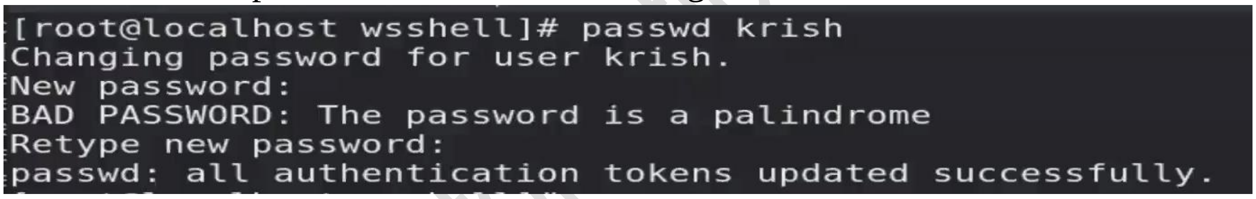
```
[root@localhost wsshell]# bash  z.sh
name alice ryan
age 21 30
```

- Who will decide which user gets which shell? The passwd file is responsible for this task.



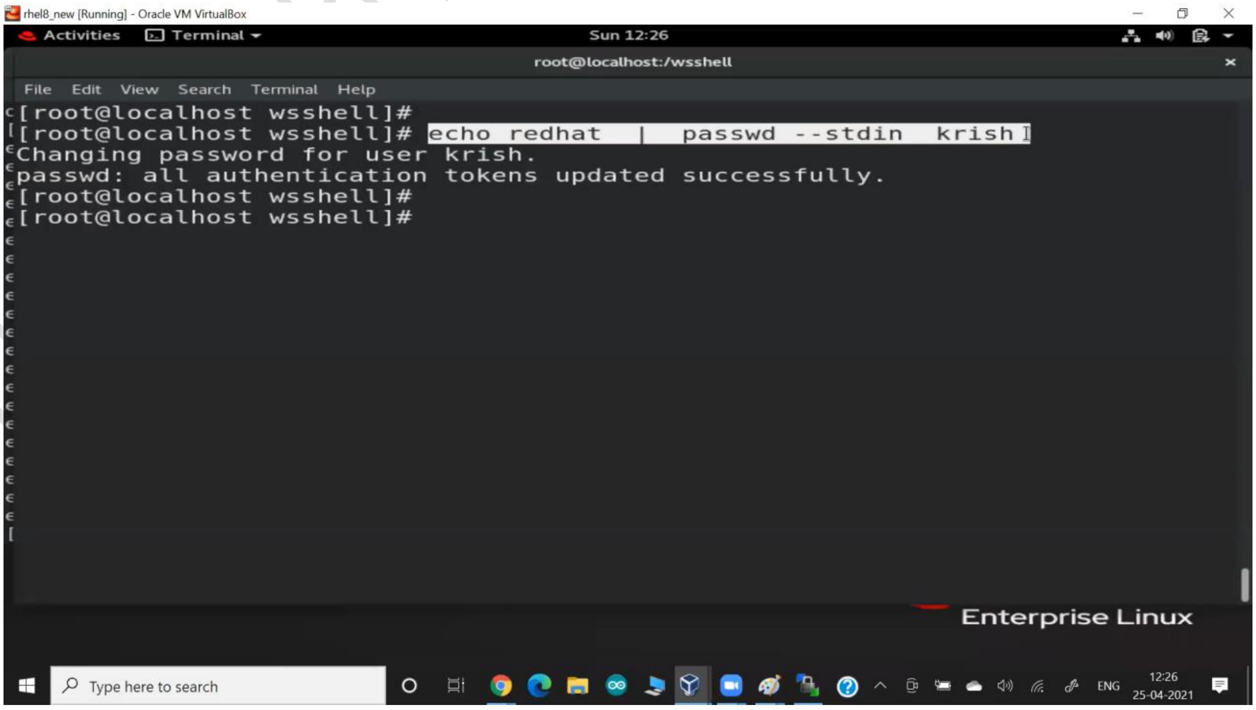
```
p2:x:1010:1010::/home/p2:/bin/bash
user3:x:1011:1011::/home/user3:/bin/bash
harry:x:1012:1012::/home/harry:/bin/bash
eric:x:1013:1013::/home/eric:/bin/bash
hjj:x:1014:1014::/home/hjj:/bin/bash
jiio:x:1015:1015::/home/jiio:/bin/bash
krish:x:1016:1016::/home/krish:/bin/bash
harry67:x:1017:1017::/home/harry67:/bin/bash
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
krj2hi:x:1022:1022::/home/krj2hi:/bin/bash
j2pop:x:1023:1023::/home/j2pop:/bin/bash
tlw:x:1024:1024::/home/tlw:/bin/bash
mlw:x:1025:1025::/etc/mlw:/bin/bash
[root@localhost wsshell]#
```

- So now I want one of my users Krish will get sh shell and for this first, we have to set a password for Krish user to log in to it.



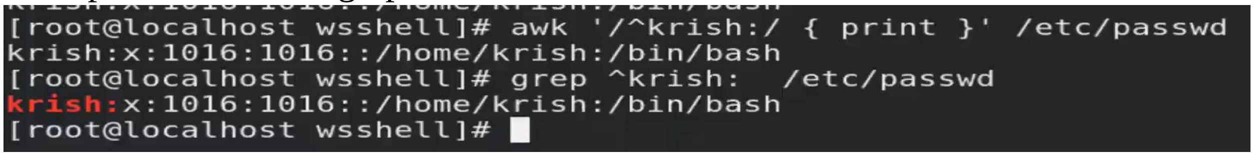
```
[root@localhost wsshell]# passwd krish
Changing password for user krish.
New password:
BAD PASSWORD: The password is a palindrome
Retype new password:
passwd: all authentication tokens updated successfully.
```

- If we want to set the password for our user via script then how can we do this because the “passwd” file is an interactive command? So for this, we use the below command.



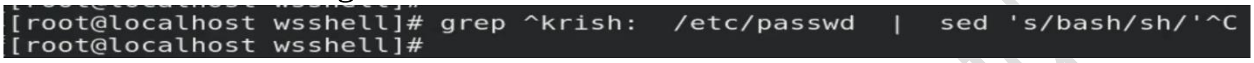
```
[root@localhost wsshell]#
[root@localhost wsshell]# echo redhat  |  passwd --stdin  krish
Changing password for user krish.
passwd: all authentication tokens updated successfully.
[root@localhost wsshell]#
[root@localhost wsshell]#
```

- To get the entire record of the Krish user from the password file, we have two options either the grep command or the awk command.



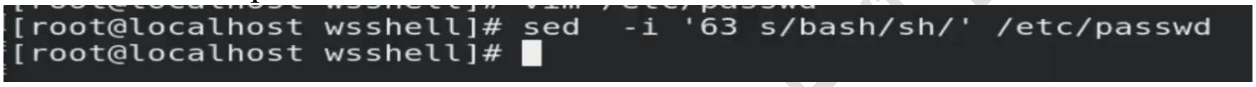
```
[root@localhost wsshell]# awk '/^krish:/ { print }' /etc/passwd
krish:x:1016:1016::/home/krish:/bin/bash
[root@localhost wsshell]# grep ^krish:  /etc/passwd
krish:x:1016:1016::/home/krish:/bin/bash
[root@localhost wsshell]#
```

- Now to edit this record we will use the “sed” command which is a string editor. This will go inside this record and change the specified string to what we ask it to change.



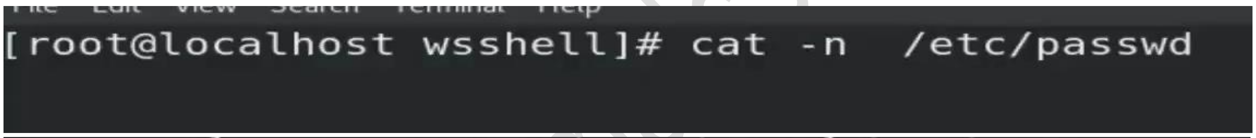
```
[root@localhost wsshell]# grep ^krish:  /etc/passwd  |  sed 's/bash/sh/'^C
[root@localhost wsshell]#
```

- But this command will not update the file internally. For updating the file we can use the option “-i ”

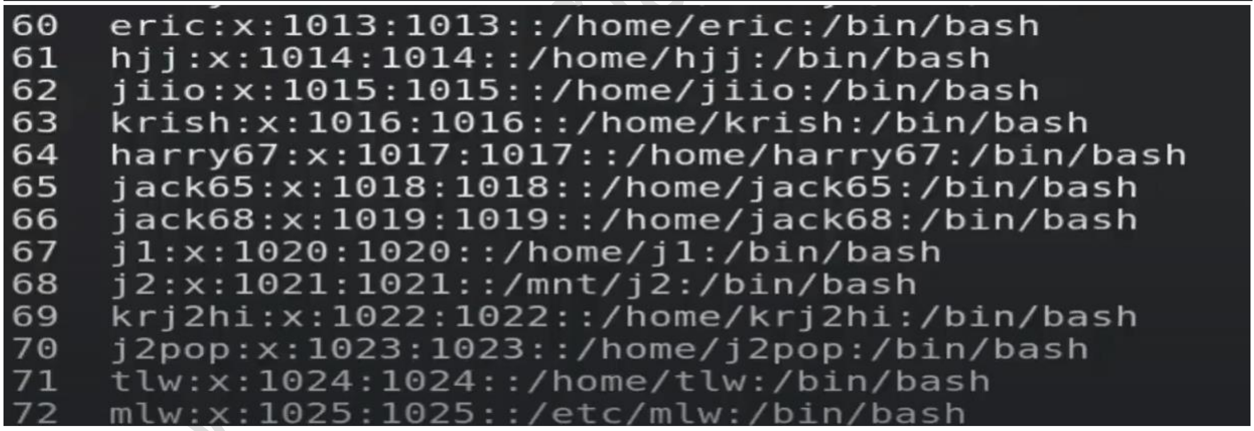


```
[root@localhost wsshell]# sed  -i '63 s/bash/sh/' /etc/passwd
[root@localhost wsshell]#
```

- There is also another way to do this same task. First, we will find the line numbers of each line from this file.

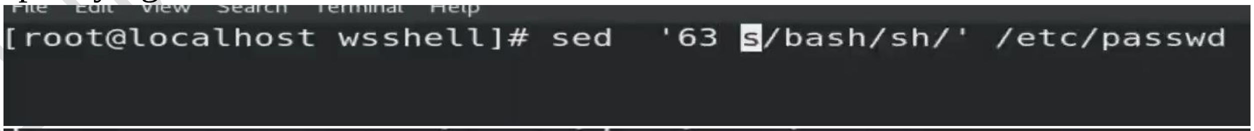


```
[root@localhost wsshell]# cat -n  /etc/passwd
```

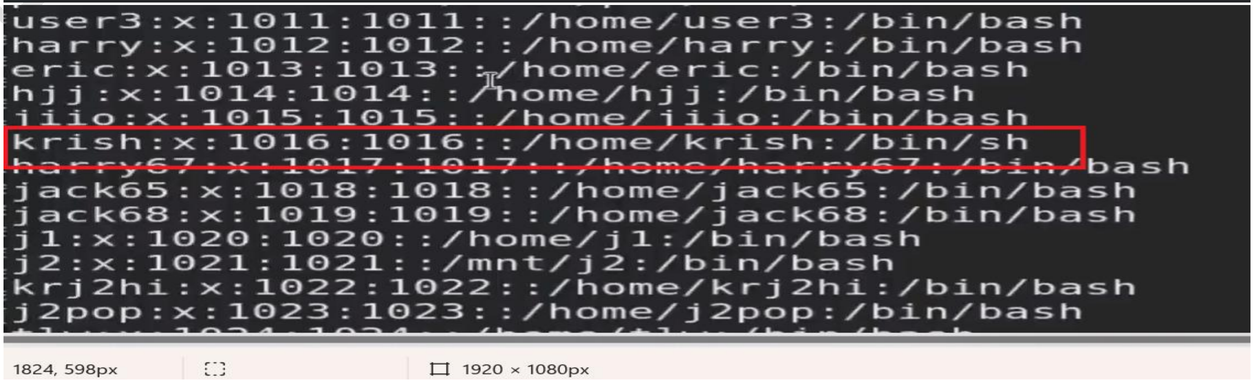


```
60  eric:x:1013:1013::/home/eric:/bin/bash
61  hjj:x:1014:1014::/home/hjj:/bin/bash
62  jiio:x:1015:1015::/home/jiio:/bin/bash
63  krish:x:1016:1016::/home/krish:/bin/bash
64  harry67:x:1017:1017::/home/harry67:/bin/bash
65  jack65:x:1018:1018::/home/jack65:/bin/bash
66  jack68:x:1019:1019::/home/jack68:/bin/bash
67  j1:x:1020:1020::/home/j1:/bin/bash
68  j2:x:1021:1021::/mnt/j2:/bin/bash
69  krj2hi:x:1022:1022::/home/krj2hi:/bin/bash
70  j2pop:x:1023:1023::/home/j2pop:/bin/bash
71  tlw:x:1024:1024::/home/tlw:/bin/bash
72  mlw:x:1025:1025::/etc/mlw:/bin/bash
```

- And then we will do the operation directly with the sed command only by specifying the line number.



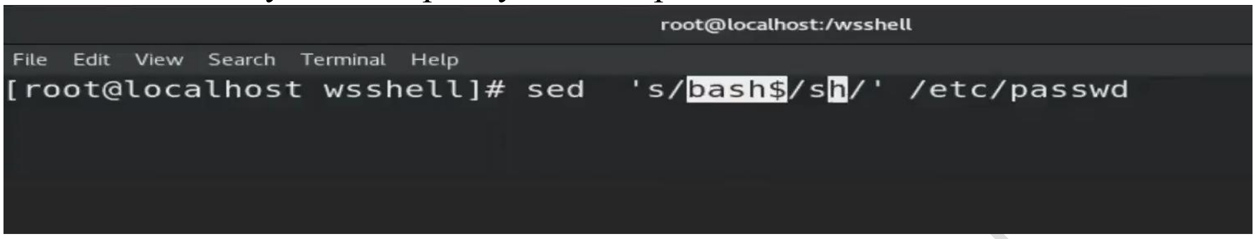
```
[root@localhost wsshell]# sed  '63 s/bash/sh/' /etc/passwd
```



```
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/sh
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
krj2hi:x:1022:1022::/home/krj2hi:/bin/bash
j2pop:x:1023:1023::/home/j2pop:/bin/bash
```

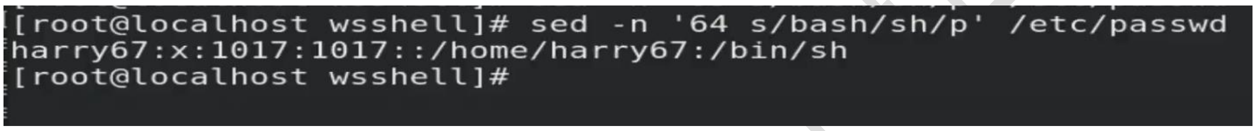
1824, 598px   1920 × 1080px

- And if we want to change the shell of all the users, we can again use the sed command and since we know that shell comes in the last field then we use the " $ " symbol to specify that this pattern comes in the last.



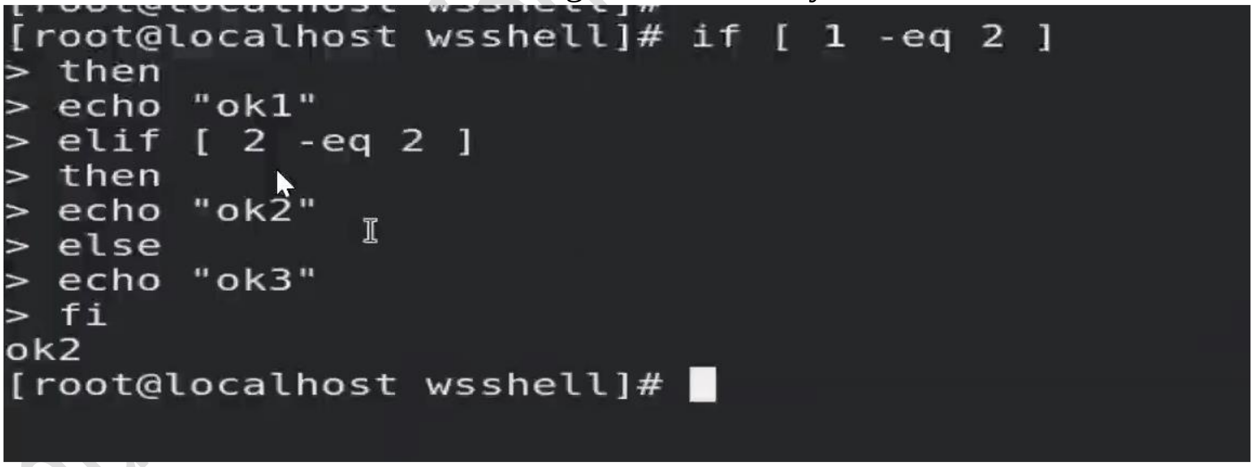
```
[root@localhost wsshell]# sed  's/bash$/sh/' /etc/passwd
```

- And if you want to run this command for a testing purpose and want to see if the changes you made were successful or not then we have the below command



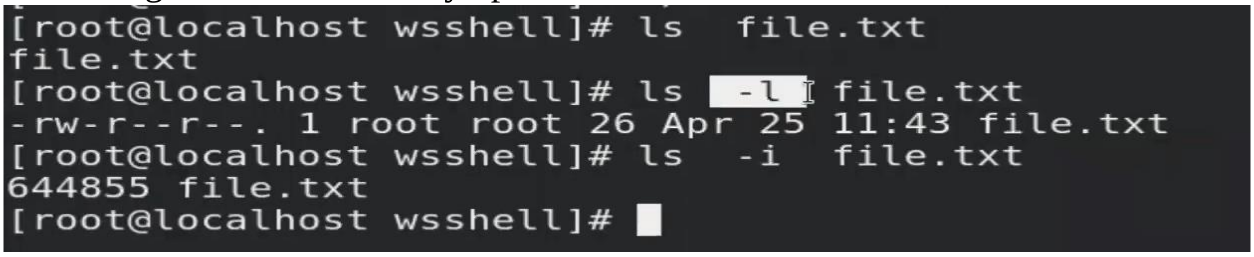
```
[root@localhost wsshell]# sed -n '64 s/bash/sh/p' /etc/passwd
harry67:x:1017:1017::/home/harry67:/bin/sh
[root@localhost wsshell]#
```

- This will make the specified changes in your file and print only those lines where changes have been made successfully.
- Now let's come to the condition part again. So if you have multiple conditions then you can use the "elif" keyword, which means else if. When if the condition doesn't match then it will go to elif and check it and if elif does not match then it will go to the else keyword and run it.



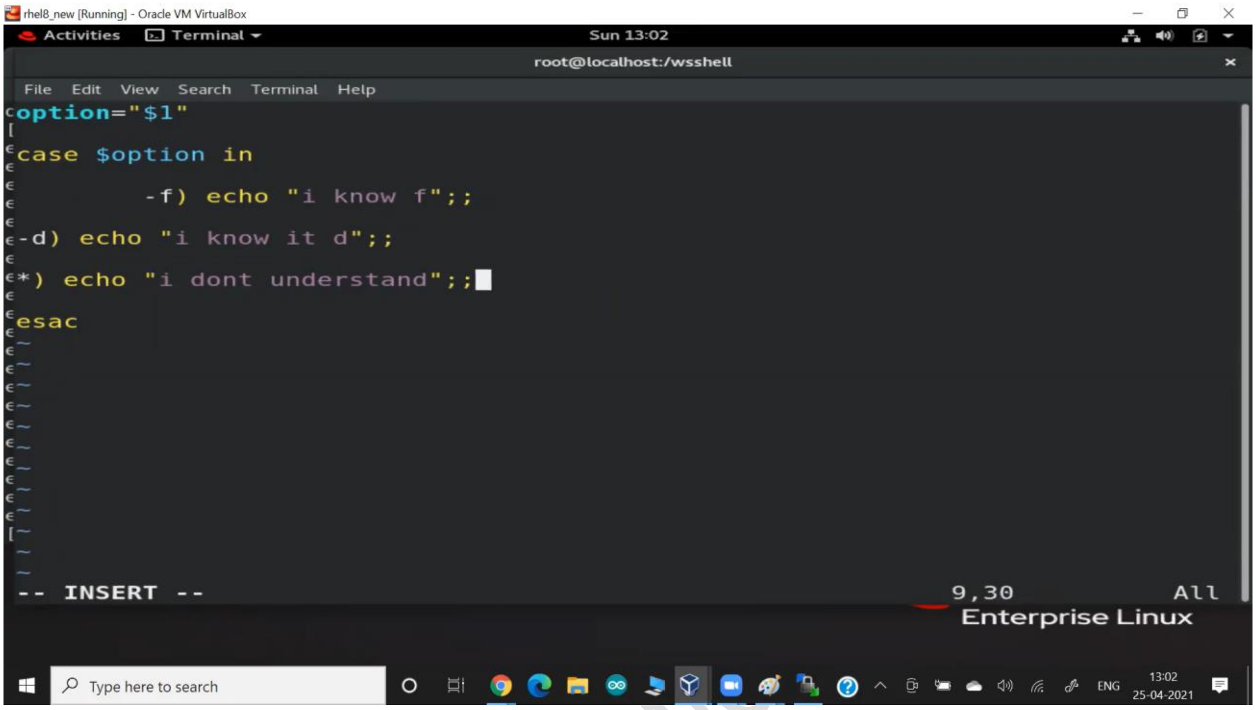
```
[root@localhost wsshell]# if [ 1 -eq 2 ]
> then
> echo "ok1"
> elif [ 2 -eq 2 ]
> then
> echo "ok2"
> else
> echo "ok3"
> fi
ok2
[root@localhost wsshell]#
```

- Now our next task is that I want to make a command that can accept options. Just like if we pass any argument to ls command and what to do on that argument is decided by options like -l.



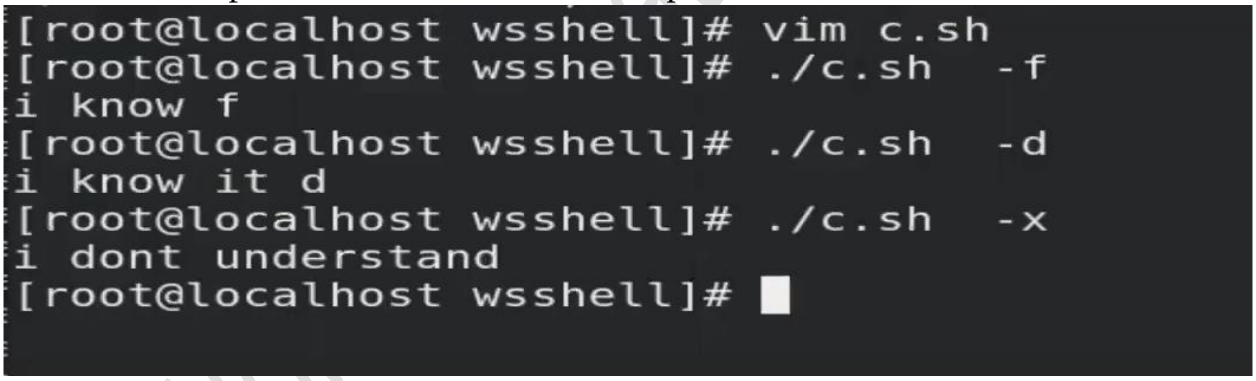
```
[root@localhost wsshell]# ls  file.txt
file.txt
[root@localhost wsshell]# ls  -l  file.txt
-rw-r--r--. 1 root root 26 Apr 25 11:43 file.txt
[root@localhost wsshell]# ls  -i  file.txt
644855 file.txt
[root@localhost wsshell]#
```

- So below is the script for this use case



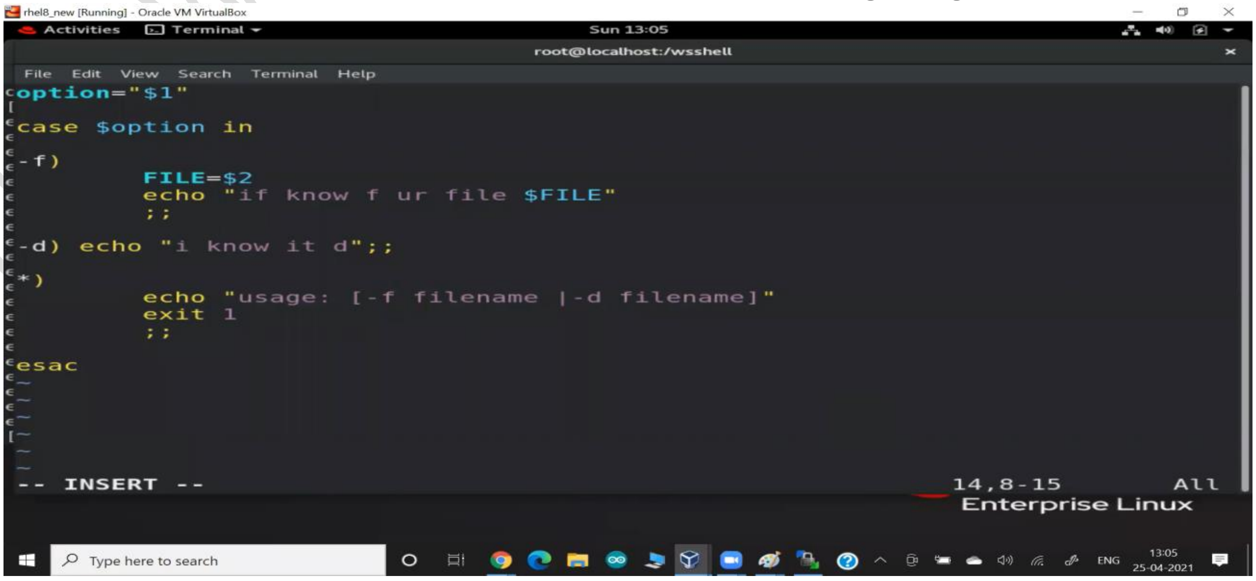
```
option="$1"

case $option in

        -f) echo "i know f";;

-d) echo "i know it d";;

*) echo "i dont understand";;

esac
-- INSERT --                                   9,30          All
```

- Now when we pass -f or -d it will do these operations

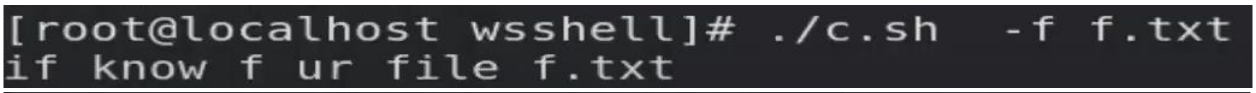


```
[root@localhost wsshell]# vim c.sh
[root@localhost wsshell]# ./c.sh  -f
i know f
[root@localhost wsshell]# ./c.sh  -d
i know it d
[root@localhost wsshell]# ./c.sh  -x
i dont understand
[root@localhost wsshell]#
```

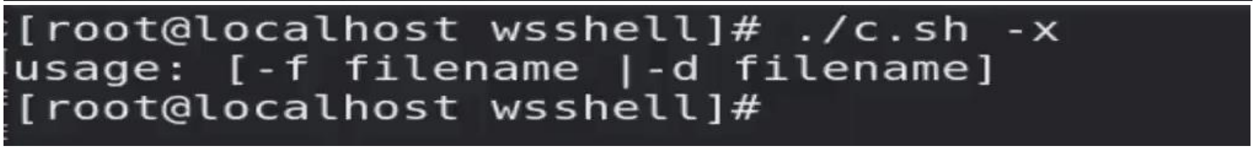
- Now to make it more customized, we can do the following things



```
option="$1"

case $option in

-f)
        FILE=$2
        echo "if know f ur file $FILE"
        ;;

-d) echo "i know it d";;

*)
        echo "usage: [-f filename |-d filename]"
        exit 1
        ;;

esac
-- INSERT --                                   14,8-15       All
```

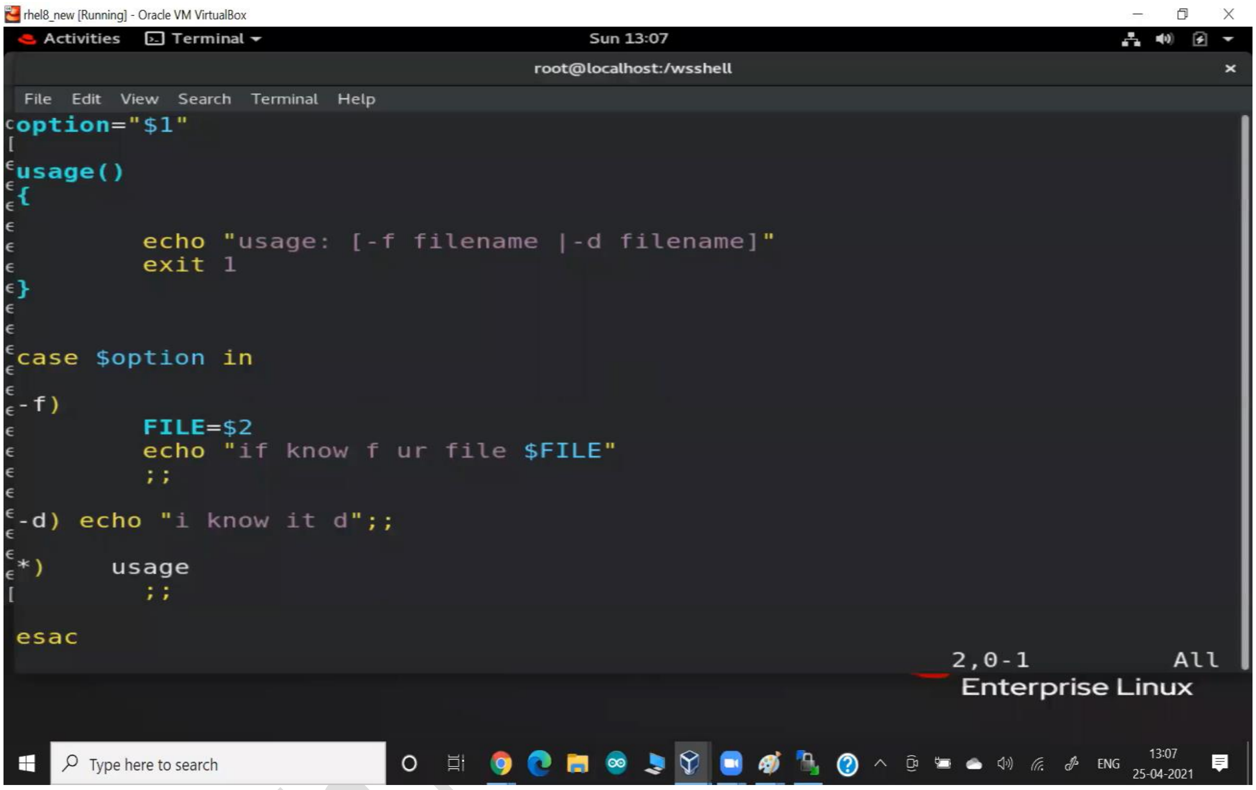


```
[root@localhost wsshell]# ./c.sh  -f f.txt
if know f ur file f.txt
```



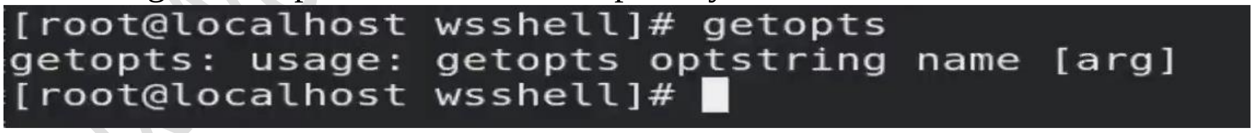
```
[root@localhost wsshell]# ./c.sh -x
usage: [-f filename |-d filename]
[root@localhost wsshell]#
```

- Or you can also create a separate function for usage here



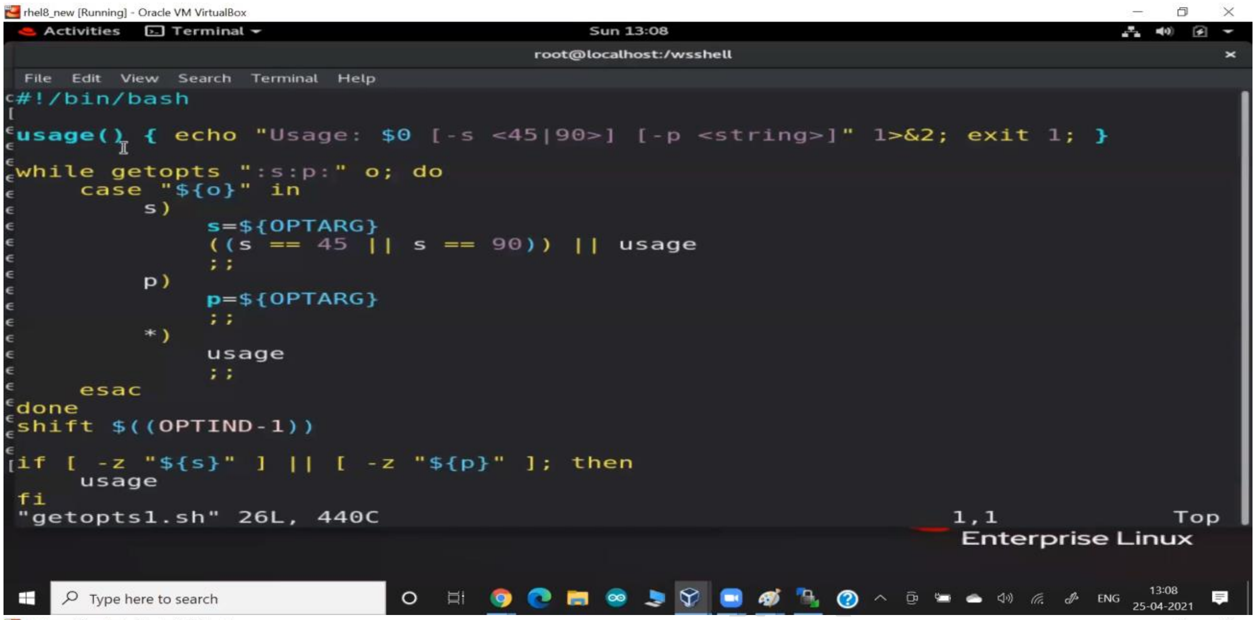
```
option="$1"

usage()
{
        echo "usage: [-f filename |-d filename]"
        exit 1
}


case $option in

-f)
        FILE=$2
        echo "if know f ur file $FILE"
        ;;

-d) echo "i know it d";;

*)      usage
        ;;

esac
```

- And in the script, sometimes we have some compulsory options or some options that are not compulsory. So this shell has the command “getopts”. This will give the options which are compulsory or not.

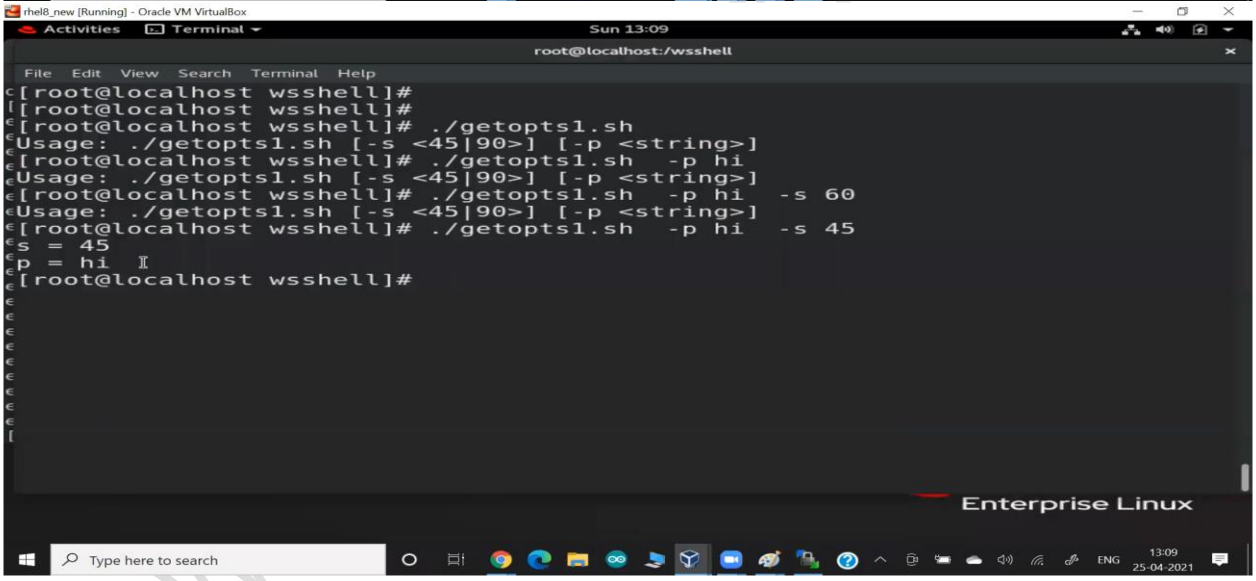


```
[root@localhost wsshell]# getopts
getopts: usage: getopts optstring name [arg]
[root@localhost wsshell]#
```

- Below is the code for getopts

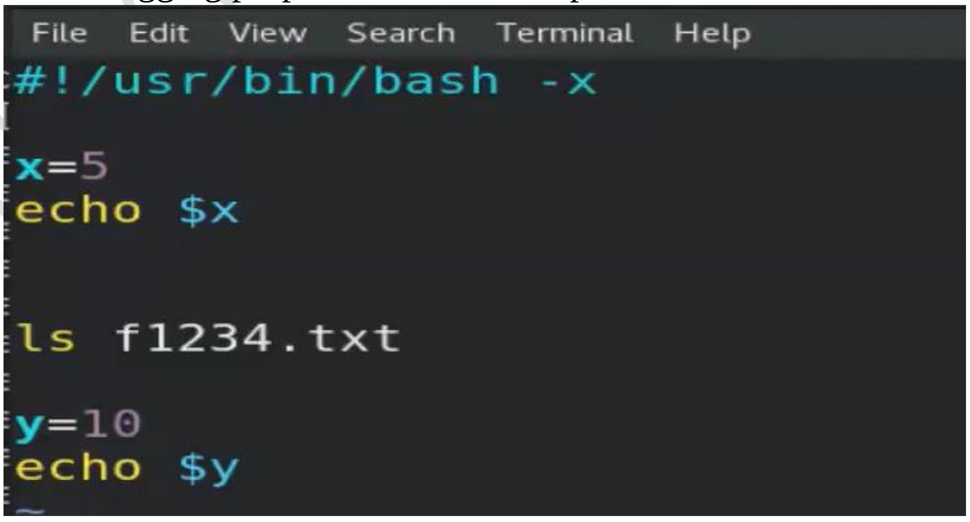


```
#!/bin/bash

usage() { echo "Usage: $0 [-s <45|90>] [-p <string>]" 1>&2; exit 1; }

while getopts ":s:p:" o; do
    case "${o}" in
        s)
            s=${OPTARG}
            ((s == 45 || s == 90)) || usage
            ;;
        p)
            p=${OPTARG}
            ;;
        *)
            usage
            ;;
    esac
done
shift $((OPTIND-1))

if [ -z "${s}" ] || [ -z "${p}" ]; then
    usage
fi
"getopts1.sh" 26L, 440C                                    1,1           Top
```



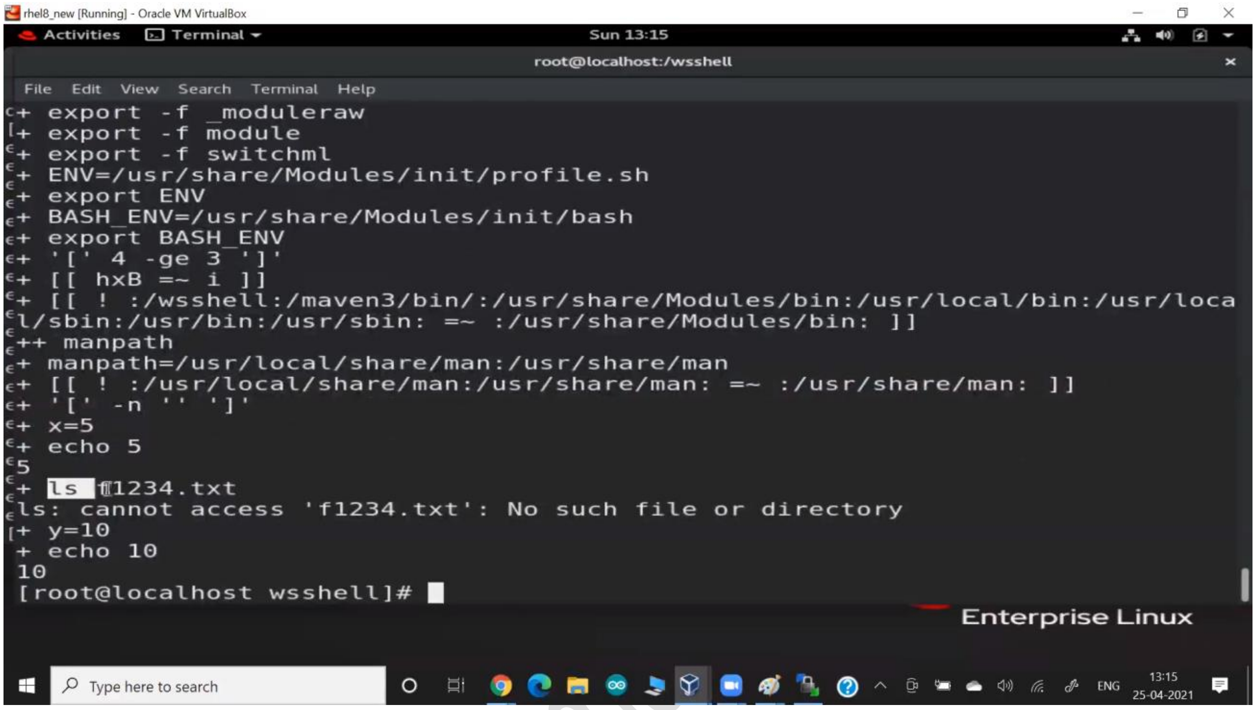
```
[root@localhost wsshell]#
[root@localhost wsshell]#
[root@localhost wsshell]# ./getopts1.sh
Usage: ./getopts1.sh [-s <45|90>] [-p <string>]
[root@localhost wsshell]# ./getopts1.sh  -p hi
Usage: ./getopts1.sh [-s <45|90>] [-p <string>]
[root@localhost wsshell]# ./getopts1.sh  -p hi  -s 60
Usage: ./getopts1.sh [-s <45|90>] [-p <string>]
[root@localhost wsshell]# ./getopts1.sh  -p hi  -s 45
s = 45
p = hi
[root@localhost wsshell]#
```

- For debugging purposes we can one option -x at the first line.



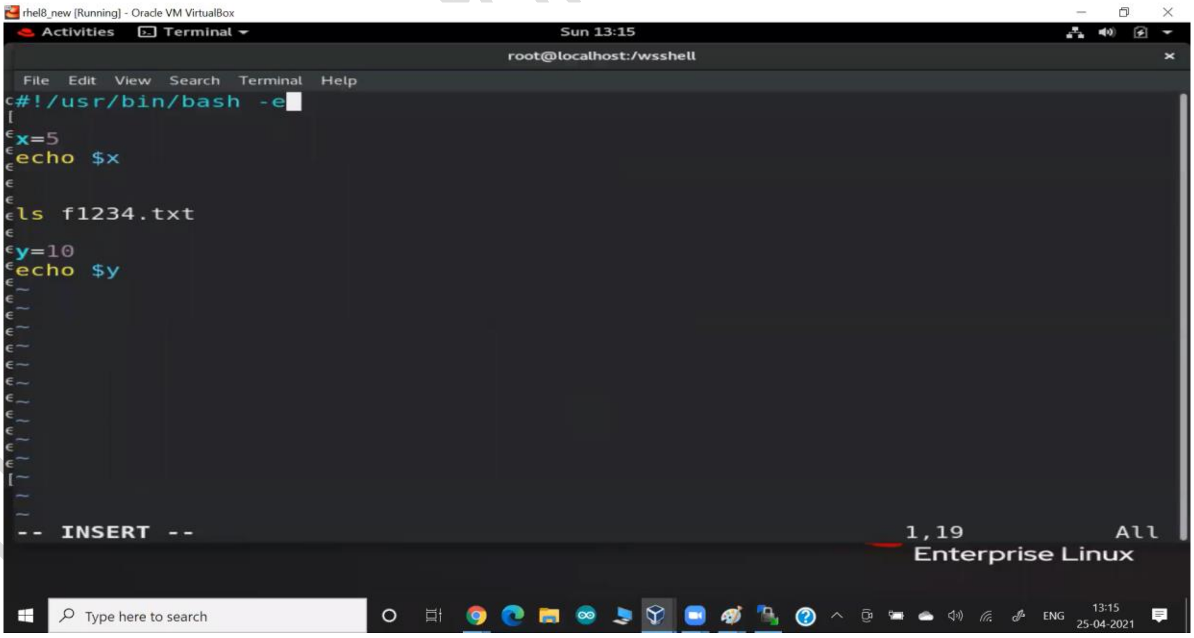
```
#!/usr/bin/bash -x

x=5
echo $x


ls f1234.txt

y=10
echo $y
```

- So, when we run this script, you will see many things that are happening behind the scenes. It will show us the error in detail and where we are doing wrong.



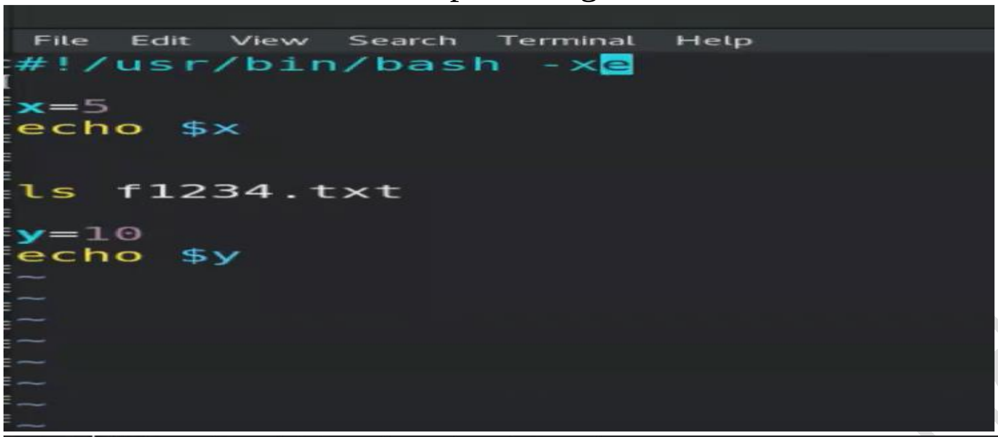
```
+ export -f _moduleraw
+ export -f module
+ export -f switchml
+ ENV=/usr/share/Modules/init/profile.sh
+ export ENV
+ BASH_ENV=/usr/share/Modules/init/bash
+ export BASH_ENV
+ '[' 4 -ge 3 ']'
+ [[ hxB =~ i ]]
+ [[ ! :/wsshell:/maven3/bin/:/usr/share/Modules/bin:/usr/local/bin:/usr/loca
l/sbin:/usr/bin:/usr/sbin: =~ :/usr/share/Modules/bin: ]]
++ manpath
+ manpath=/usr/local/share/man:/usr/share/man
+ [[ ! :/usr/local/share/man:/usr/share/man: =~ :/usr/share/man: ]]
+ '[' -n '' ']'
+ x=5
+ echo 5
5
+ ls f1234.txt
ls: cannot access 'f1234.txt': No such file or directory
+ y=10
+ echo 10
10
[root@localhost wsshell]#
```

- We have one other and similar option "-e". It will stop the script whenever an error occurs and don't go further.

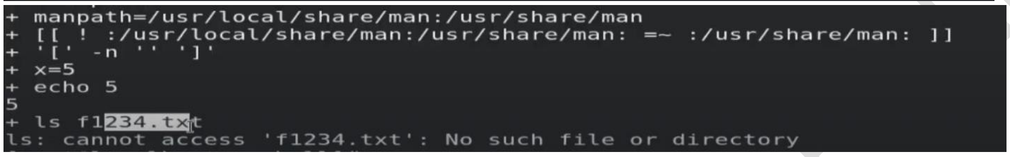


```
#!/usr/bin/bash -e

x=5
echo $x

ls f1234.txt

y=10
echo $y
~
-- INSERT --                                         1,19          All
```

- And We can also write both options together -xe

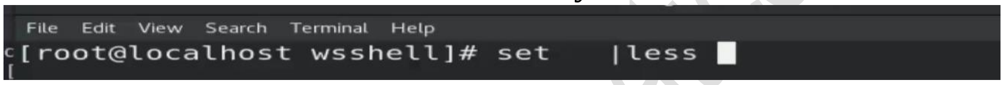


```
#!/usr/bin/bash -xe

x=5
echo $x

ls f1234.txt

y=10
echo $y
~
~
~
~
~
~
~
~
```

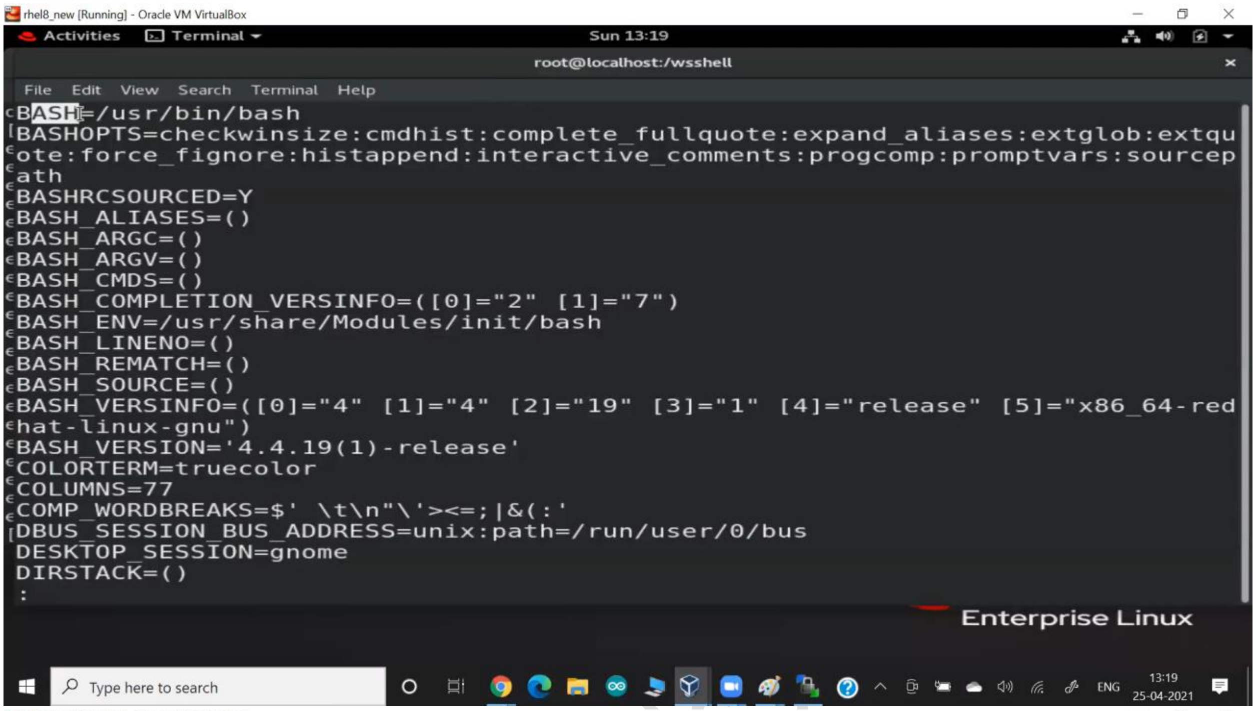


```
+ manpath=/usr/local/share/man:/usr/share/man
+ [[ ! :/usr/local/share/man:/usr/share/man: =~ :/usr/share/man: ]]
+ '[' -n '' ']'
+ x=5
+ echo 5
5
+ ls f1234.txt
ls: cannot access 'f1234.txt': No such file or directory
```

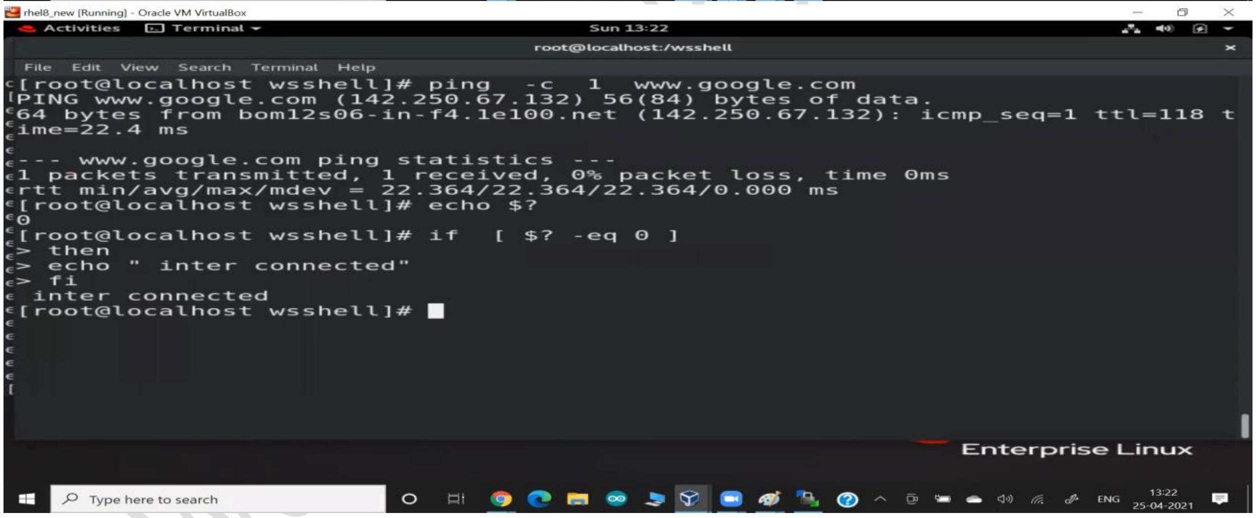
- And there is a command in the shell that is “set” which shows you all the internal functions and variables already created



```
[root@localhost wsshell]# set   |less
```



```
BASH=/usr/bin/bash
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:expand_aliases:extglob:extqu
ote:force_fignore:histappend:interactive_comments:progcomp:promptvars:sourcep
ath
BASHRCSOURCED=Y
BASH_ALIASES=()
BASH_ARGC=()
BASH_ARGV=()
BASH_CMDS=()
BASH_COMPLETION_VERSINFO=([0]="2" [1]="7")
BASH_ENV=/usr/share/Modules/init/bash
BASH_LINENO=()
BASH_REMATCH=()
BASH_SOURCE=()
BASH_VERSINFO=([0]="4" [1]="4" [2]="19" [3]="1" [4]="release" [5]="x86_64-red
hat-linux-gnu")
BASH_VERSION='4.4.19(1)-release'
COLORTERM=truecolor
COLUMNS=77
COMP_WORDBREAKS=$' \t\n"\'><=;|&(:'
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/0/bus
DESKTOP_SESSION=gnome
DIRSTACK=()
:
```



```
[root@localhost wsshell]# ping  -c  1  www.google.com
PING www.google.com (142.250.67.132) 56(84) bytes of data.
64 bytes from bom12s06-in-f4.1e100.net (142.250.67.132): icmp_seq=1 ttl=118 t
ime=22.4 ms

--- www.google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 22.364/22.364/22.364/0.000 ms
[root@localhost wsshell]# echo $?
0
[root@localhost wsshell]# if  [ $? -eq 0 ]
> then
> echo " inter connected"
> fi
 inter connected
[root@localhost wsshell]# 
```