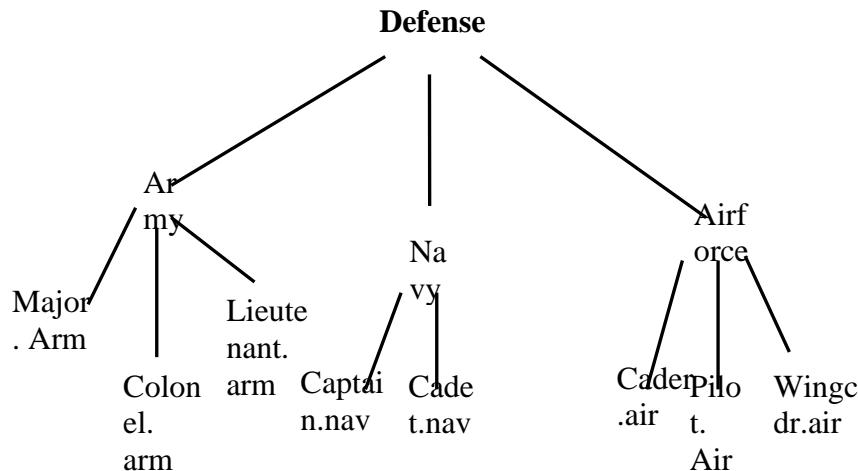


Lab Exercise - 1

Give the commands for the following:

1. Display the name of your home directory on the screen.
2. Create the following directory structure (Defense, army, navy and Airforce are directories, and under them are the files)



3. Add the following content to these files:

Major.arm – All the second lieutenant works under the major. It is a high post in the army.

Colonel.arm – Colonel is responsible for all the majors under his regiment.

Lieutenant.arm – He is responsible to look after all the cadets.

Captain.nav – One ship is managed by one captain.

Cadet.nav – Cadets are the trimness in Navy.

Cader.air – Cader are the cadets of Air force

Pilot.air – Pilot manages the higher plain.

Wingcdr.air – Wing Cdr manages all pilots under his regiment

4. Write the command to display the contents of these files.
5. Standing in the home directory, go to the Airforce subdirectory in a single command.
6. Create sub directories Operating and reporting under Airforce directory.
7. Go to the directory reporting, and create sub directories, secretary and sergeant.
8. Go to the directory defense and create sub directories wings, groups and squadron under operating sub directory using a single command.
9. Standing in the defense directory, append the content to the file captain.nav in navy subdirectory.
10. Copy the file major.arm to Wingcdr.air.
11. Display the content of the file Wingcdr.air.\
12. Concatenate the contents of the files Colonel.arm and Cader.air to a new file , Test, under Navy sub directory.
13. Copy the file, Test, to a new file, Tmpfile, under Groups sub directory.
14. Delete the file Test interactively.

Solutions Lab Exercise -1 :-

1.

```
[root@localhost Desktop]# pwd  
/root/Desktop
```

2.

```
[root@localhost Desktop]# mkdir defense  
[root@localhost Desktop]# cd defense  
[root@localhost defense]# mkdir army navy airforce  
[root@localhost defense]# cd army  
[root@localhost army]# touch major.arm colonel.arm lieutenant.arm  
[root@localhost army]# cd ..  
[root@localhost defense]# cd navy  
[root@localhost navy]# touch captain.nav cadet.nav  
[root@localhost navy]# cd ..  
[root@localhost defense]# cd airforce  
[root@localhost airforce]# touch cadet.air pilot.air wingcdr.air  
[root@localhost airforce]# █
```

3.

```
[root@localhost army]# cat > major.arm  
All the second lieutenant works under the major. It is a high post in the army.  
[root@localhost army]# cat > colonel.arm  
Colonel is responsible for all the majors under his regiment.  
[root@localhost army]# cat > lieutenant.arm  
He is responsible to look after all the cadets.  
[root@localhost army]# cd ..  
[root@localhost defense]# cd navy  
[root@localhost navy]# cat > captain.nav  
One ship is managed by one captain.  
[root@localhost navy]# cat > cadet.nav  
Cadets are the trimness in Navy.  
[root@localhost navy]# cd ..  
[root@localhost defense]# cd airforce  
[root@localhost airforce]# cat > cader.air  
Cader are the cadets of Air force  
[root@localhost airforce]# cat > pilot.air  
Pilot manages the higher plain.  
[root@localhost airforce]# cat > wingcdr.air  
Wing Cdr manages all pilots under his regiment  
[root@localhost airforce]# █
```

4.

```
[root@localhost army]# cat major.arm  
All the second lieutenant works under the major. It is a high post in the army.  
[root@localhost army]# cat colonel.arm  
Colonel is responsible for all the majors under his regiment.  
[root@localhost army]# cat lieutenant.arm  
He is responsible to look after all the cadets.
```

```
[root@localhost defense]# cd navy
[root@localhost navy]# cat captain.nav
One ship is managed by one captain.
[root@localhost navy]# cat cadet.nav
Cadets are the trimness in Navy.
[root@localhost navy]# cd ..
[root@localhost defense]# cd airforce
[root@localhost airforce]# cat cader.air
Cader are the cadets of Air force
[root@localhost airforce]# cat pilot.air
Pilot manages the higher plain.
[root@localhost airforce]# cat wingcdr.air
Wing Cdr manages all pilots under his regiment
[root@localhost airforce]#
```

5.

```
[root@localhost ~]# cd /root/Desktop/defense/airforce
[root@localhost airforce]#
```

6.

```
[root@localhost airforce]# mkdir operating reporting
[root@localhost airforce]# ls
cader.air  operating  pilot.air  reporting  wingcdr.air
[root@localhost airforce]#
```

7.

```
[root@localhost airforce]# cd reporting
[root@localhost reporting]# mkdir secretary sergeant
[root@localhost reporting]# ls
secretary  sergeant
```

8.

```
[root@localhost defense]# mkdir airforce/operating/wings airforce/
operating/groups airforce/operating/squadron
[root@localhost defense]# ls airforce/operating/
groups  squadron  wings
[root@localhost defense]#
```

9.

```
[root@localhost defense]# cat >> navy/captain.nav
This is my appended text.
[root@localhost defense]# cat navy/captain.nav
One ship is managed by one captain.
This is my appended text.
```

10.
[root@localhost defense]# cp army/major.arm airforce/wingcdr.air
cp: overwrite `airforce/wingcdr.air'? y

11.
cp: overwrite `airforce/wingcdr.air'? y
[root@localhost defense]# cat airforce/wingcdr.air
All the second lieutenant works under the major. It is a high post
in the army.

12.
[root@localhost defense]# cat army/colonel.arm airforce/cader.air
>> navy/test
[root@localhost defense]# █

13.
[root@localhost defense]# cp navy/test airforce/operating/groups/t
mpfile
[root@localhost defense]# █

14.
[root@localhost defense]# rm -i navy/test
rm: remove regular file `navy/test'? y

Lab Exercise - 2

Refer to assignment 1 for directory structure.

Give the commands for the following:

15. Move the file Tmpfile from sub directory Groups to sub directory, Sergeant.
16. Rename the file Tmpfile as Subordinate. air
17. Create a directory Navalchief in Navy subdirectory .
18. Move this directory Navalchief from Navy subdirectory to Reporting subdirectory.
19. Rename this directory Navalchief, as Airchief.
20. Verify that directory has been renamed.
21. Display the name of all users logged in currently and determine your terminal name.
22. Now display only your terminal filename.
23. Create five empty files Empty1, Empty2, Empty3, Empty4 and Empty5 under defense directory.
24. Display the names of all ordinary files in your home directory.
25. Standing in home directory give command
who >oldfile

Now append the contents of a file called oldfile to newfile. Display the contents of oldfile and newfile to see if you have given the correct commands.

26. Suppose there are following files in present working directory:
art, part, part1, part2, part3, mozart, tart, quartz.

Which of the above files would qualify for the following searches:

- (a) ls a?
- (b) ls a*
- (c) ls *.*
- (d) ls [!abc]art
- (e) ls [a!bc]art
- (f) ls [b-dku-z]*
27. Suppose the path dir1/dir2/dir3/dir4 exists in your directory. All these directories are empty. Write down the command to remove all of them at one shot?
28. Create a hidden file hidden1 in defense directory.
29. Verify that the hidden file is created.
30. Standing in the defense directory list all the files starting with 'e'
31. Standing in the defense directory list all the files starting with any single character but end with 'ing'
32. Standing in the Navy subdirectory list all files starting with a, b, c and ending with any no of characters.
33. Standing in the Navy subdirectory list all files NOT starting with a, b, and c
34. Standing in the Army subdirectory list all three character files whose first character can be anything from a to z, second character any vowel and third character can be anything.
35. Display the long listing of all files present in the directory Navy. Note the entries for Directories and files.
36. Display the names of all files that start with a lower case alphabet followed by a digit and ending with '.c' or '.o' in army subdirectory.
37. Move all the files whose names end with '.c' or '.o' to the directory Olddir in secretary sub directory.
38. In one command, remove all the files in Olddir and the directory itself.
39. Append the following content in the empty file, Empty1
A line is any group of characters not containing a newline
A word is a group of characters not containing a space, tab or newline.

- A character is the smallest unit of information, and includes a space, tab and newline.
40. Display the no of lines, words and characters present in the file, Empty1.
 41. Redirect the output of file Empty1 to a new file, infile, in the same directory.
 42. Compare the contents of these two files.
 43. Append the content to the file, infile.
wc offers three options to make a specific count.
 44. Now compare the content of these two files, Empty1 and infile.
 45. Display what is common in these two files. Observe the display.

Solutions of Lab Exercise 2:-

15.

```
[root@localhost Desktop]# mv defense/airforce/operating/groups/  
tmpfile defense/airforce/reporting/sergeant  
[root@localhost Desktop]# █
```

16.

```
[root@localhost Desktop]# defense/airforce/reporting/sergeant m  
v tmpfile subordinate.air
```

17.

```
[root@localhost Desktop]# mkdir defense/navy/navalchief  
[root@localhost Desktop]# █
```

18.

```
[root@localhost Desktop]# mv defense/navy/navalchief defense/airfor  
ce/reporting  
[root@localhost Desktop]# █
```

19.

```
[root@localhost Desktop]# mv defense/airforce/reporting/navalchi  
ef defense/airforce/reporting/airchief  
[root@localhost Desktop]# █
```

20.

```
[root@localhost Desktop]# ls defense/airforce/reporting  
airchief secretary sergeant  
[root@localhost Desktop]# █
```

21.

```
[root@localhost Desktop]# who  
root    tty1        2023-12-13 01:46 (:0)  
root    pts/0        2023-12-13 02:51 (:0.0)  
[root@localhost Desktop]# █
```

22.

```
[root@localhost Desktop]# tty  
/dev/pts/0  
[root@localhost Desktop]# █
```

23.

```
[root@localhost defense]# touch empty1 empty2 empty3 empty4 empty5  
[root@localhost defense]# █
```

24.

```
[root@localhost ~]# ls  
anaconda-ks.cfg  Downloads          Music    Templates  
Desktop          install.log        Pictures  Videos  
Documents         install.log.syslog Public  
[root@localhost ~]# █
```

25.

```
[root@localhost ~]# who > oldfile  
[root@localhost ~]# cat oldfile >> newfile  
[root@localhost ~]# cat oldfile  
root    tty1          2023-12-13 01:46 (:0)  
root    pts/0          2023-12-13 02:51 (:0.0)  
[root@localhost ~]# cat newfile  
root    tty1          2023-12-13 01:46 (:0)  
root    pts/0          2023-12-13 02:51 (:0.0)  
[root@localhost ~]# █
```

26.

a) ls a

Ans: No such files present

```
[root@localhost test]# ls a  
ls: cannot access a: No such file or directory
```

b) ls a*

Ans: art

```
[root@localhost test]# ls a*  
art
```

c) ls*.*

Ans: anaconda-ks.cfg install.log install.log.syslog

```
[root@localhost test]# ls *.*  
ls: cannot access *.*: No such file or directory
```

d) ls [!abc]art

Ans: part, tart

```
[root@localhost ~]# ls [!abc]art
ls: cannot access [!abc]art: No such file or directory
[root@localhost ~]#
```

e) ls [a!bc]art

Ans: event not found.

```
[root@localhost test]# ls [a!bc]art
bash: !bc]art: event not found
```

f) ls [b-dku-z]*

Ans: No such file or directory

```
[root@localhost test]# ls [b-dku-z]*
ls: cannot access [b-dku-z]*: No such file or directory
[root@localhost test]#
```

27.

```
[root@localhost ~]# rm -r dir1
```

28.

```
[root@localhost defense]# touch .hidden1
[root@localhost defense]#
```

29.

```
[root@localhost defense]# touch .hidden1
[root@localhost defense]# ls -a
.  airforce  empty1  empty3  empty5    navy
..  army     empty2  empty4  .hidden1
[root@localhost defense]#
```

30.

```
[root@localhost defense]# ls e*
empty1  empty2  empty3  empty4  empty5
[root@localhost defense]#
```

31.

```
[root@localhost defense]# ls ?ing
ls: cannot access ?ing: No such file or directory
[root@localhost defense]#
```

32.

```
[root@localhost navy]# ls [a-c]*
cadet.nav captain.nav
[root@localhost navy]#
```

33.

```
[root@localhost navy]# ls [!abc]*
ls: cannot access [!abc]*: No such file or directory
```

34.

```
[root@localhost army]# ls
colonel.arm lieutenant.arm major.arm
[root@localhost army]#
```

35.

```
[root@localhost army]# ls -l
total 12
-rw-r--r--. 1 root root 62 Dec 12 22:45 colonel.arm
-rw-r--r--. 1 root root 48 Dec 12 22:46 lieutenant.arm
-rw-r--r--. 1 root root 80 Dec 12 22:45 major.arm
[root@localhost army]#
```

36.

```
[root@localhost army]# ls [a-z][0-9]*[{"c","o"}]
ls: cannot access [a-z][0-9]*[.c: No such file or directory
ls: cannot access [a-z][0-9]*[.o: No such file or directory
[root@localhost army]#
```

37.

```
a directory
[root@localhost Desktop]# mv defense/army/*[{"c","o"}] defense/airforce/reporting/secretary/olddir/
```

38.

```
[root@localhost secretary]# rm -r olddir
rm: remove directory `olddir'? y
[root@localhost secretary]#
```

39.

```
[root@localhost defense]# cat >> empty1
A line is any group of characters not containing a newline.
A word is a group of characters not containing a space, tab or n
ewline.
A character is the smallest unit of information, and includes a
space, tab and newline.
[root@localhost defense]#
```

40.

```
[root@localhost defense]# wc empty1
 3 40 220 empty1
[root@localhost defense]#
```

41.

```
[root@localhost defense]# cat empty1 > infile
[root@localhost defense]# cat infile
A line is any group of characters not containing a newline.
A word is a group of characters not containing a space, tab or n
ewline.
A character is the smallest unit of information, and includes a
space, tab and newline.
[root@localhost defense]#
```

42.

```
[root@localhost defense]# cmp empty1 infile
[root@localhost defense]#
```

43.

```
[root@localhost defense]# cat >> infile
wc offers three options to make a specific count.
[root@localhost defense]#
```

44.

```
[root@localhost defense]# cmp empty1 infile
cmp: EOF on empty1
```

45.

```
[root@localhost defense]# comm empty1 infile
                  A line is any group of characters not containing
a newline.
                  A word is a group of characters not containing a
space, tab or newline.
                  A character is the smallest unit of information,
and includes a space, tab and newline.
      wc offers three options to make a specific count.
[root@localhost defense]#
```

Lab Exercise - 3

- I. Construct pipelines to carry out the following jobs:
- (a) List all files beginning with the character “P” on the screen and also store them in a file called File1.
 - (b) List all files beginning with the character “P” on the screen twice in succession.
 - (c) Merge the contents of the files a.txt, b.txt and c.txt, sort them and display the sorted output on the screen page by page.
 - (d) Display the list of last 20 files present in the current directory. Also store this list in a file ‘profile’.
 - (e) Write a command to transfer 3 lines from one file (FILE1) to (FILE2) and vice versa.
 - (f) Write a command to append the data of FILE1, FILE2 to FILE3.
- II. What Regular Expression would you use to do the following:
- (a) List all the records having either woodhouse or wodehouse.
 - (b) List all the records having either trueman or truman.
 - (c) List all the records having Wilcox or wilcocks.
 - (d) List the records having first name as p. followed by any numbers of characters and last name as woodhouse.
 - (e) List the records which begin with employee code 2.....
 - (f) Inverse your regular expression i.e. list all except the ones starting with 2.
 - (g) List all records having salary between 70000 to 79999.
 - (h) List records having p.g. woodhouse as name
 - (i) List only directories in your current directories.

Solutions of Lab Exercise 2 :-

I.

a)

```
[root@localhost ~]# ls P* | tee file1  
Pictures:
```

```
Public:
```

```
[root@localhost ~]# cat file1  
Pictures:
```

```
Public:
```

```
[root@localhost ~]# █
```

b)

```
[root@localhost ~]# ls P* |more  
Pictures:
```

```
Public:
```

```
[root@localhost ~]# █
```

c)

```
[root@localhost ~]# cat a.txt b.txt c.txt > abc.txt && cat abc.txt | > sort|more
[root@localhost ~]# cat abc.txt
apple
ball
cat
[root@localhost ~]# █
```

d)

```
[root@localhost ~]# cat a.txt b.txt c.txt > abc.txt && cat abc.txt | > sort|more
[root@localhost ~]# cat abc.txt
apple
ball
cat
[root@localhost ~]# ls -l | tail -20 | tee profile
anaconda-ks.cfg
a.txt
b.txt
c.txt
Desktop
Documents
Downloads
file1
install.log
install.log.syslog
Music
newfile
oldfile
Pictures
profile
Public
sort
Templates
test
Videos
[root@localhost ~]# █
```

e)

```
[root@localhost ~]# head -3 file1 | tee file2
Pictures:

Public:
[root@localhost ~]# head -3 file2 | tee file1
Pictures:

Public:
[root@localhost ~]# █
```

f)

```
[root@localhost ~]# cat file1 file2 | tee file3
Pictures:
Public:
Pictures:
Public:
[root@localhost ~]#
```

II.

a)

```
[root@localhost Desktop]# grep "wo[od][de]house" file.txt
woodhouse wodehouse name is the good workwoodhouse wodehousetttr
huigh woodhouse
good woodhouse
bad wodehouse
tty woodhousemmn
[root@localhost Desktop]#
```

b)

```
[root@localhost Desktop]# grep -E "tru(e|)man" file.txt
trueman
truman
nihtruman
yes trueman
[root@localhost Desktop]#
```

c)

```
[root@localhost Desktop]# grep -E "[wW]ilco(x|cks)" file.txt
wilcox
wilcocks
hgwilcocks
for wilcocks
[root@localhost Desktop]#
```

d)

```
[root@localhost Desktop]# grep -i "p.*woodhouse" file.txt
p.m woodhouse
p.q woodhouse
high p woodhouse
[root@localhost Desktop]#
```

e)

```
[root@localhost Desktop]# grep "^2" file.txt
21 emp code
202 emp code
205 emp code
[root@localhost Desktop]#
```

f)

```
[root@localhost Desktop]# grep "^\w" filename.txt
grep: filename.txt: No such file or directory
[root@localhost Desktop]# grep "^\w" file.txt
21 emp code
202 emp code
205 emp code
[root@localhost Desktop]# grep -v "^\w" file.txt
woodhouse wodehouse name is the good workwoodhouse wodehousettr
good
very bad
huigh woodhouse
good woodhouse
bad wodehouse
tty woodhousemmn
trueman
truman
nihtruman
yes trueman
wilcox
wilcocks
hgwilcocks
for wilcocks
p.m woodhouse
p.q woodhouse
high p woodhouse
[root@localhost Desktop]#
```

g)

```
[root@localhost Desktop]# grep "7...." file.txt
salary: 70000
salary: 71000
salary: 75000
[root@localhost Desktop]#
```

h)

```
[root@localhost Desktop]# grep -i "p.g. woodhouse" file.txt
p.g. woodhouse
p.g. woodhouse high
low p.g. woodhouse good
[root@localhost Desktop]#
```

i)

```
[root@localhost ~]# ls -l | grep "^\w"
drwxr-xr-x. 4 root root 4096 Dec 13 04:38 Desktop
drwxr-xr-x. 2 root root 4096 Jan 29 2015 Documents
drwxr-xr-x. 2 root root 4096 Jan 29 2015 Downloads
drwxr-xr-x. 2 root root 4096 Jan 29 2015 Music
drwxr-xr-x. 2 root root 4096 Jan 29 2015 Pictures
drwxr-xr-x. 2 root root 4096 Jan 29 2015 Public
drwxr-xr-x. 2 root root 4096 Jan 29 2015 Templates
drwxr-xr-x. 2 root root 4096 Dec 13 03:31 test
drwxr-xr-x. 2 root root 4096 Jan 29 2015 Videos
drwxr-xr-x. 2 root root 4096 Dec 13 04:29 wewoodhouse
drwxr-xr-x. 2 root root 4096 Dec 13 04:30 wodehouseoo
[root@localhost ~]#
```

Lab Exercise - 4

- 1) Devise suitable wild-card patterns to match the following filenames:
 - (i) foo1, foo2 and foo5
 - (ii) quit.c, quit.o and quit.h
 - (iii) tutorial.ps and tutorial.pdf
 - (iv) all filenames beginning with a dot and ending with .swp
- 2) Frame wild-card patterns to match all filenames where the first character is numeric and the last character is not alphabetic.
- 3) Devise a command that copies all files named chap01, chap02, chap03 and so forth, and up to chap26 to the parent directory. Can a single wild-card pattern match them all?
- 4) Can we write the following command in a more compact form?
Cat<file1 | grep John > result
- 5) What is the difference between the commands:
wc -l < file1
wc -l file1

Solutions of Lab Exercise – 4 :-

1.

i)

```
[root@localhost lab4]# ls foo?
foo1  foo2  foo5
[root@localhost lab4]#
```

ii)

```
[root@localhost lab4]# ls quit.?
quit.c  quit.h  quit.o
[root@localhost lab4]#
```

iii)

```
[root@localhost lab4]# ls tutorial.*
tutorial.pdf  tutorial.ps
[root@localhost lab4]#
```

iv)

```
[root@localhost lab4]# ls *.swp
fa1.swp  fa2.swp  fa3.swp
[root@localhost lab4]#
```

2.

```
[root@localhost lab4]# ls [0-9]*[!a-zA-Z]
4asus1  6gfg8  7span5
[root@localhost lab4]#
```

3.

```
[root@localhost lab4]# cp chap{01..26} ../
[root@localhost lab4]# cd ..
[root@localhost Desktop]# ls
chap01  chap06  chap11  chap16  chap21  chap26      lab3
chap02  chap07  chap12  chap17  chap22  defense      lab4
chap03  chap08  chap13  chap18  chap23  file1
chap04  chap09  chap14  chap19  chap24  file.txt
chap05  chap10  chap15  chap20  chap25  lab 1 linux
[root@localhost Desktop]#
```

Yes, a single wild-card pattern like chap?? Or chap* can match all these files.

```
[root@localhost Desktop]# ls chap??
chap01  chap05  chap09  chap13  chap17  chap21  chap25
chap02  chap06  chap10  chap14  chap18  chap22  chap26
chap03  chap07  chap11  chap15  chap19  chap23
chap04  chap08  chap12  chap16  chap20  chap24
[root@localhost Desktop]#
```

4.

```
[root@localhost Desktop]# grep John < file1> result
[root@localhost Desktop]#
```

5.

Ans: Both commands provide the line count for file1, but the first command (wc -l < file1) does not include the filename in the output, while the second command (wc -l file1) does include the filename in the output.

```
[root@localhost Desktop]# wc -l < file1
2
[root@localhost Desktop]# wc -l file1
2 file1
[root@localhost Desktop]#
```

Lab Exercise - 5

1. What is the difference between the commands:
2. cat<file1>file2
3. cat>file2<file1

4. You are given a file myfile. Without opening this file how would you make a fair estimate about its contents?
5. What will be the effect of following commands?
 - (ii) umask
 - (iii) chmodu+w g-w abcd.out
 - (iv) chmod 777 aaa.c
 - (v) chmodug+rw a=x ffff.out
 - (vi) chmodu+tmydir

Solutions of Lab Exercise - 5

1.

Ans: Both commands take the contents of file1 and overwrite the contents of file2 with them, and the only difference lies in the order of input and output redirections. The result is the same in both cases.

2.

Ans: Using ‘wc’ - word count command. It gives no. of lines, word, and character count for each file.

```
[root@localhost Desktop]# wc file1
 2 2 13 file1
[root@localhost Desktop]# █
```

3.

(i) **umask**: It will give the default umask value for the current system.

```
[root@localhost Desktop]# umask
0022
[root@localhost Desktop]# █
```

(ii)) **chmod u+w g-w abcd.out** :

This command will give write permission to user and remove write permission from the group of file “abcd.out”.

```
[root@localhost lab4]# ls -l abcd.out
-rw-r--r--. 1 root root 0 Dec 13 05:21 abcd.out
[root@localhost lab4]# chmod u+w,g-w abcd.out
[root@localhost lab4]# ls -l abcd.out
-rw-r--r--. 1 root root 0 Dec 13 05:21 abcd.out
[root@localhost lab4]# █
```

(iii) chmod 777 aaa.c :

This command will give all read(4), write(2) and execute(1) permissions to user, group, others of file aaa.c

```
[root@localhost lab4]# ls -l aca.c
-rw-r--r--. 1 root root 0 Dec 13 05:23 aca.c
[root@localhost lab4]# chmod 777 aca.c
[root@localhost lab4]# ls -l aca.c
-rwxrwxrwx. 1 root root 0 Dec 13 05:23 aca.c
[root@localhost lab4]# █
```

(iv) chmod ug+rw a=x ffff.out:

This will add read and write permission to user and group and a=x will assign execute permission to all user, group and others of file ffff.out.

```
[root@localhost lab4]# ls -lfaf.out
-rw-r--r--. 1 root root 0 Dec 13 05:25faf.out
[root@localhost lab4]# chmod a=xfaf.out
[root@localhost lab4]# ls -lfaf.out
---x---x---. 1 root root 0 Dec 13 05:25faf.out
[root@localhost lab4]# █
```

(v) chmod u+t mydir:

This will add a sticky bit to mydir directory. It means only user can delete or rename the files contained in mydir. Group and others cannot do the same.

```
[root@localhost lab4]# mkdir mydir
[root@localhost lab4]# chmod u+t mydir
[root@localhost lab4]# █
```

Lab Exercise - 6

Q1 Using sed command, write down the commands for the followings

What Regular Expression would you use to do the following:

- (j) List all the records having either woodhouse or wodehouse.
- (k) List all the records having either trueman or truman.
- (l) List all the records having Wilcox or wilcocks.
- (m) List the records having first name as p. followed by any numbers of characters and last name as woodhouse.
- (n) List the records which begin with employee code 2.....
- (o) Inverse your regular expression i.e. list all except the ones starting with 2.
- (p) List all records having salary between 70000 to 79999.
- (q) List records having p.g. woodhouse as name
- (r) List only directories in your current directories.

Solutions of Lab Exercise - 6

1.

(j) .

```
[root@localhost Desktop]# sed -n '/woodhouse\|wodehouse/p' file.txt
woodhouse wodehouse name is the good workwoodhouse wodehousettr
huigh woodhouse
good woodhouse
bad wodehouse
tty woodhousemmn
p.m woodhouse
p.q woodhouse
high p woodhouse
p.g. woodhouse
p.g woodhouse
p.g. woodhouse high
low p.g. woodhouse good
[root@localhost Desktop]# █
```

(k)

```
[root@localhost Desktop]# sed -n '/trueman\|truman/p' file.txt
trueman
truman
nihtruman
yes trueman
[root@localhost Desktop]# █
```

(l)

```
[root@localhost Desktop]# sed -n '/Wilcox\|wilcocks/p' file.txt
wilcocks
hwilcocks
for wilcocks
[root@localhost Desktop]#
```

(m)

```
[root@localhost Desktop]# sed -n '/^p.* woodhouse/p' file.txt
p.m woodhouse
p.q woodhouse
p.g. woodhouse
p.g woodhouse
p.g. woodhouse high
[root@localhost Desktop]#
```

(n)

```
[root@localhost Desktop]# sed -n '/^2/p' file.txt
21 emp code
202 emp code
205 emp code
[root@localhost Desktop]#
```

(o)

```
[root@localhost Desktop]# sed -n '/^2/!p' file.txt
woodhouse wodehouse name is the good workwoodhouse wodehousetttr
good
very bad
huigh woodhouse
good woodhouse
bad wodehouse
tty woodhousemmn
trueman
truman
nihtruman
yes trueman
wilcox
wilcocks
hwilcocks
for wilcocks
p.m woodhouse
p.q woodhouse
high p woodhouse
salary: 70000
salary: 71000
salary: 75000
p.g. woodhouse
p.g woodhouse
p.g. woodhouse high
low p.g. woodhouse good
[root@localhost Desktop]#
```

(p)

```
low p.g. woodhouse good
[root@localhost Desktop]# sed -n '/7..../p' file.txt
salary: 70000
salary: 71000
salary: 75000
[root@localhost Desktop]#
```

(q)

```
[root@localhost Desktop]# sed -n '/p.g. woodhouse/Ip' file.txt
p.g. woodhouse
p.g. woodhouse high
low p.g. woodhouse good
[root@localhost Desktop]#
```

(r)

```
[root@localhost Desktop]# ls -l | sed '-n '/^d/p'
drwxr-xr-x. 5 root root 4096 Dec 13 04:03 defense
drwxr-xr-x. 2 root root 4096 Dec 13 04:08 lab3
drwxr-xr-x. 3 root root 4096 Dec 13 05:27 lab4
[root@localhost Desktop]#
```

Lab Exercise -7

Question No. 1

Write a program to verify how many users are logged on to the system.

```
#vi file1.sh
```

```
#sh file1.sh
```

```
Echo "The no of users logged in are `who | wc -l`"
```

Question No. 2

Write a program that will calculate the amount of disk space your directory is using on the system.

```
Du
```

```
Dfspace
```

```
Echo "The amount of disk space used is `du`"
```

Question No. 3

Write Script to see current date, time, username, and current directory

```
Echo "The current date and time is `date`"
```

```
Echo "The current user logged in is `USERNAME` `LOGNAME`"
```

```
Echo "The current directory is `HOME`"
```

```
Echo "The present working directory is `pwd`"
```

Question No. 4

How to calculate $5.12 + 2.5$ real number calculation at \$ prompt in Shell ?

```
A=5.12
```

```
B=2.5
```

```
Echo "The value of calculation of a and b is `expr $a + $b`"
```

```
`echo $a + $b`
```

```
Echo "value is `expr 5.12 + 2.5`"
```

```
`echo 5.12 + 2.5`
```

Question No. 5

How to perform real number calculation in shell script and store result to third variable ,lets say a=5.66, b=8.67, c=a+b?

Solutions of Lab Exercise -7 :-

Q1)

```
# file7.sh
echo "The no of Users logged in are `who | wc -l`"
[root@localhost Desktop]# sh file1.sh
The no of users logged in are 2
```

Q2)

```
# diskpspace.sh
echo "The amount of disk space used is `du`"
```

```
[root@localhost Desktop]# vi diskspace.sh
[root@localhost Desktop]# sh diskspace.sh
The amount of disk space used is 8      ./Nothing
16    ./Defence/Army
4     ./Defence/Airforce/Operating/groups
4     ./Defence/Airforce/Operating/squadron
4     ./Defence/Airforce/Operating/wings
16    ./Defence/Airforce/Operating
4     ./Defence/Airforce/Reporting/secretory
4     ./Defence/Airforce/Reporting/sergeant
12    ./Defence/Airforce/Reporting
44    ./Defence/Airforce
16    ./Defence/Navy
80    ./Defence
12    ./Harsh
128   .
[root@localhost Desktop]#
```

Q3)

```
# file3.sh
echo "The current date and time is `date`"
echo "The current user logged in is $USER $LOGNAME"
echo "The current directory is $HOME"
[root@localhost Desktop]# vi file3.sh
[root@localhost Desktop]# sh file3.sh
The current date and time is Fri Dec  8 16:18:10 IST 2023
The current user logged in is root root
The current directory is /root
[root@localhost Desktop]#
```

Q4)

```
# file4.sh
a=5.12
b=2.5
result=`echo $a + $b | bc`
echo "The value of calculation $a + $b is $result"
[root@localhost ~]# sh file4.sh
The value of calculation 5.12 + 2.5 is 7.62
[root@localhost ~]#
```

Q5)

```
# file5.sh
a=5.66
b=8.67
c=`echo $a + $b`
echo "The value of $a + $b = $c"
```

```
[root@localhost ~]# sh file5.sh
The value of 5.66 + 8.67 = 5.66 + 8.67
[root@localhost ~]#
```

Lab Exercise - 8

Question No. 1

How to write script, that will print, Message "Hello World" , in Bold and Blink effect, and in different colors like red, brown etc using echo command.

Question No. 2

Write shell script to show various system configuration like

1) Currently logged user and his logname

2) Your current shell

Echo \$SHELL

3) Your home directory

Echo \$HOME

4) Your operating system type

Echo \$OSTYPE

5) Your current path setting

Echo \$PATH

6) Your current working directory

Echo \$PWD

7) Show Currently logged number of users

Echo `who | wc -l`

8) About your os and version ,release number , kernel version

/etc/Redhat-version

/etc/Redhat-release

9) Show all available shells

/etc/shells

10) Show mouse settings

11) Show computer cpu information like processor type, speed etc

Cpuinfo

12) Show memory informationmeminfo

13) Show hard disk information like size of hard-disk, cache memory, model etc

14) File system (Mounted)

/dev/mnt/cdrom

/dev/mnt/hda1

Question No. 3

If cost price and selling price of an item is input through the keyboard, write a program to determine whether the seller has made profit or incurred loss. Also determine how much profit was made or loss incurred.

Question No. 4

Any integer is input through the keyboard. Write a program to find out whether it is an even number or odd number.

If test `expr \$a % 2`

Question No. 5

Write a shell script, which receives any year from the Keyboard, and determine whether the year is a leap year or not. If no argument is supplied the current year should be assumed.

Question No. 6

Write a shell script, which receives two filenames as arguments. It should check whether the two file's contents are same or not. If they are same then second file should be deleted.

Solutions of Lab Exercise - 8

Q1)

```
# hello.sh
# bold
echo -e "\033[1m Hello World"
# blink
echo -e "\033[5m Hello World"
# colours
echo -e "\033[31m Hello World"
echo -e "\033[32m Hello World"
echo -e "\033[33m Hello World"
echo -e "\033[34m Hello World"
echo -e "\033[35m Hello World"
```

```
[root@localhost ~]# sh hello.sh
Hello World
[root@localhost ~]#
```

Q2)

```
# ques2.sh
echo "Username: $USER"
echo "Logname: $LOGNAME"
echo "Current Shell: $SHELL"
echo "Home Directory: $HOME"
echo "Operating System Type: $OSTYPE"
echo "Path: $PATH"
echo "Current directory: `pwd`"
echo "Currently Logged: `who | wc -l` user(s)"
echo "OS: `cat /etc/os-release`"
echo "Kernel version: `uname -r`"
echo "Available shells: `cat /etc/shells`"
echo "Mouse settings: `cat /etc/sysconfig/mouse`"
echo "Memory info: `cat /proc/meminfo`"
echo "Model: `cat /proc/ide/hda/driver`"
echo "Cache Size: `cat /proc/ide/hda/cache`"
echo "File System(mount): `cat /proc/mounts`"
```

```
[root@localhost /]# sh ques2.sh
Username: root
Logname: root
Current Shell: /bin/bash
Home Directory: /root
Operating System Type: linux-gnu
Path: /usr/local/sbin:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin
Current directory: /
Currently Logged: 0 user(s)
OS: NAME=Fedora
VERSION="33 (Rawhide)"
ID=fedora
VERSION_ID=33
VERSION_CODENAME=""
PLATFORM_ID="platform:f33"
PRETTY_NAME="Fedora 33 (Rawhide)"
ANSI_COLOR="0;34"
LOGO=fedora-logo-icon
CPE_NAME="cpe:/o:fedoraproject:fedora:33"
HOME_URL="https://fedoraproject.org/"
DOCUMENTATION_URL="https://docs.fedoraproject.org/en-US/fedora/rawhide/system-administrators-guide/"
SUPPORT_URL="https://fedoraproject.org/wiki/Communicating_and_getting_help"
BUG_REPORT_URL="https://bugzilla.redhat.com/"
REDHAT_BUGZILLA_PRODUCT="Fedora"
REDHAT_BUGZILLA_PRODUCT_VERSION=rawhide
REDHAT_SUPPORT_PRODUCT="Fedora"
REDHAT_SUPPORT_PRODUCT_VERSION=rawhide
PRIVACY_POLICY_URL="https://fedoraproject.org/wiki/Legal:PrivacyPolicy"
```

```
Kernel version: 4.15.0-00049-ga3b1e7a-dirty
Available shells: /bin/sh
/bin/bash
/usr/bin/sh
/usr/bin/bash
/usr/bin/tmux
/bin/tmux
/bin/csh
/bin/tcsh
/usr/bin/csh
/usr/bin/tcsh
/usr/bin/zsh
/bin/zsh
cat: /etc/sysconfig/mouse: No such file or directory
Mouse settings:
Memory info: MemTotal:           186324 kB
MemFree:            170740 kB
MemAvailable:       171636 kB
Buffers:            0 kB
Cached:             5356 kB
SwapCached:         0 kB
```

```
Active:           10160 kB
Inactive:         1296 kB
Active(anon):    6100 kB
Inactive(anon):   0 kB
Active(file):    4060 kB
Inactive(file):  1296 kB
Unevictable:     0 kB
Mlocked:          0 kB
SwapTotal:        0 kB
SwapFree:         0 kB
Dirty:            0 kB
Writeback:        0 kB
AnonPages:        6136 kB
Mapped:           4200 kB
Shmem:            0 kB
Slab:             3144 kB
SReclaimable:    544 kB
SUnreclaim:      2600 kB
KernelStack:     184 kB
PageTables:      136 kB
NFS_Unstable:    0 kB
Bounce:           0 kB
WritebackTmp:     0 kB
CommitLimit:     93160 kB
Committed_AS:    7764 kB
VmallocTotal:    67108863 kB
VmallocUsed:     0 kB
VmallocChunk:    0 kB
```

```
File System(mount): /dev/root / 9p rw,dirsync,relatime,mmap,access=client,trans=virtio 0 0
devtmpfs /dev devtmpfs rw,relatime,size=93120k,nr_inodes=23280,mode=755 0 0
none /proc proc rw,relatime 0 0
none /sys sysfs rw,relatime 0 0
devpts /dev/pts devpts rw,relatime,mode=600,ptmxmode=000 0 0
```

Q3)

```
echo "Enter Selling Price (SP): "
read sp
#!/bin/sh

# Input cost price and selling price
echo "Enter Cost Price (CP): "
read cp

echo "Enter Selling Price (SP): "
read sp

# Calculate profit or loss
if [ $sp -gt $cp ]
then
profit=`expr $sp - $cp`
echo "Profit: $profit"

elif [ $cp -gt $sp ]
then
loss=`expr $cp - $sp`
echo "Loss: $loss"

else
echo "No Profit, No Loss"
fi
```

```
[root@localhost ~]# sh ques3.sh
Enter Cost Price (CP):
20
Enter Selling Price (SP):
15
Loss: 5
[root@localhost ~]# sh ques3.sh
Enter Cost Price (CP):
20
Enter Selling Price (SP):
25
Profit: 5
[root@localhost ~]# sh ques3.sh
Enter Cost Price (CP):
20
Enter Selling Price (SP):
20
No Profit, No Loss
[root@localhost ~]#
```

Q4)

```
If test `expr $a % 2`  
# Check Even or Odd number  
read -p "Enter a number: " num  
if [ $(($num%2)) -eq 0 ]  
then  
    echo "Number is even."  
else  
    # Check Even or Odd number  
    read -p "Enter a number: " num  
    if [ $(($num%2)) -eq 0 ]  
    then  
        echo "Number is even."  
    else  
        echo "Number is odd."  
    fi
```

```
[root@localhost ~]# sh evenOdd.sh  
Enter a number: 24  
Number is even.  
[root@localhost ~]# sh evenOdd.sh  
Enter a number: 15  
Number is odd.  
[root@localhost ~]#
```

Q5)

```
# find leap year  
# If no argument is provided, use the current year  
if [ "$#" -eq 0 ]; then  
    year=$(date +"%Y")  
    # find leap year  
  
    # If no argument is provided, use the current year  
    if [ "$#" -eq 0 ]; then  
        year=$(date +"%Y")  
    else  
        year=$1  
    fi  
  
    # Check if it's a leap year  
    if [ $((year % 4)) -eq 0 ] && [ $((year % 100)) -ne 0 -o $((year%400)) -eq 0 ]; then  
        echo "$year is a leap year."  
    else  
        echo "$year is not a leap year."  
    fi
```

```
[root@localhost ~]# sh leapyear.sh 2000  
2000 is a leap year.  
[root@localhost ~]# sh leapyear.sh 1900  
1900 is not a leap year.  
[root@localhost ~]# sh leapyear.sh  
2023 is not a leap year.  
[root@localhost ~]#
```

Q6)

```
# Check if two filenames are provided as arguments
if [ $# -ne 2 ]; then
    echo "Usage: $0 <file> <file2>"
    exit 1
fi

file1=$1
# Check if two filenames are provided as arguments
if [ $# -ne 2 ]; then
    echo "Usage: $0 <file> <file2>"
    exit 1
fi

file1=$1
file2=$2

# Check if both files exist
if [ ! -e "$file1" -o ! -e "$file2" ]; then
    echo "Error: Both files must exist."
    exit 1
fi

# Check if the contents of the two files are the same
if cmp -s "$file1" "$file2"; then
    echo "The contents of $file1 and $file2 are the Same. Deleting $file2"
    rm "$file2"
else
    echo "The contents of $file1 and $file2 are different."
```

```
[root@localhost ~]# sh files.sh
Usage: files.sh <file> <file2>
[root@localhost ~]# sh files.sh file1 file2
The contents of file1 and file2 are different.
```

```
[root@localhost ~]# sh files.sh file1 file2
The contents of file1 and file2 are the Same. Deleting file2
[root@localhost ~]#
```

Lab Exercise - 9

1. Write a script that would first verify if file "myfile" exists. If it does not, create it, then ask the user for confirmation to erase it...
2. Write a shell script to accept a file name. Test whether file is ordinary file or directory file. If ordinary file check whether readable and then display its contents. If directory display its contents.
3. Write a shell script to print only those lines of file that do not contain word “example”, also store output in a file.
4. Write a shell script to view all environment variables and store it in a file.

Solutions of Lab Exercise - 9

1.

```
# myfile.sh

# Check if the file exists
if [ -e $file ]; then
    echo "File $file exists."
else
    echo "File $file does not exist. Creating it."
touch $file
fi

# Ask for confirmation to erase the file

read -p "Do you want to erase the $file? (yes/no):" choice

if [ "$choice" == "yes" ]; then
    rm $file
    echo "$file has been erased."
else
    echo "No action taken. $file is not erased."
fi
```

```
[root@localhost ~]# sh myfile.sh
File myfile exists.
Do you want to erase myfile? (yes/no):yes
myfile has been erased.
[root@localhost ~]# sh myfile.sh
File myfile does not exists. Creating it.
Do you want to erase myfile? (yes/no):no
No action taken. myfile is not erased.
[root@localhost ~]#
```

2.

```
# Accept a file name from the user

# Check if the file exists
if [ ! -e $filename ]; then
# Accept a file name from the user
    read -p "Enter a file name: " filename

# Check if the file exists
if [ ! -e $filename ]; then
    echo "Error: File $filename does not exists."
    exit 1
fi

# Check if it's an ordinary file or a directory
if [ -f $filename ]; then
    echo "$filename is an ordinary file."

# Check if the file is readable
if [ -r $filename ]; then
    echo "Content of $filename:"
    cat $filename
else
    echo "Error: $filename is not readable."
fi

elif [ -d $filename ]; then
    echo "$filename is a directory. Contents:"
    ls $filename

else
    echo "Error: $filename is neither an ordinary file nor a directory."
fi
```

```
[root@localhost ~]# sh checkfile.sh
Enter a file name: myfile
myfile is an ordinary file.
Content of myfile:
This is myfile content.
Hello, everyone.
[root@localhost ~]#
```

```
[root@localhost ~]# sh checkfile.sh
Enter a file name: mydir
mydir is a directory. Contents:
test1 test2 test3
[root@localhost ~]#
```

3.

```
# textfile.sh
read -p "Enter the file name: " filename

# Check if the file exists
if [ ! -e "$filename" ]; then
    echo "Error: File $filename does not exist."
exit 1
fi

# Specify the output file name
read -p "Enter Output file name: " outfile

# Print lines not containing the word "example" to the console and store in the
# output file
read -p "Enter the file name: " filename

# Check if the file exists
if [ ! -e "$filename" ]; then
    echo "Error: File $filename does not exist."
exit 1
fi

# Specify the output file name
read -p "Enter Output file name: " outfile

# Print lines not containing the word "example" to the console and store in the
# output file
grep -v "example" "$filename" | tee "$outfile"

echo "Filtered lines have been stored in '$outfile'."
```

```
[root@localhost ~]# sh textfile.sh
Enter the file name: sample.txt
Enter Output file name: noexample.txt
This line does not contain the word.
Filtered lines have been stored in 'noexample.txt'.
[root@localhost ~]#
```

4.

```
# env.sh

# specify the output filename
read -p "Enter the output filename: " output_file

# view all environment variables and store in the output file
# env.sh

# specify the output filename
read -p "Enter the output filename: " output_file

# view all environment variables and store in the output file
# env.sh

# specify the output filename
read -p "Enter the output filename: " output_file

# env.sh
# env.sh
# specify the output filename
read -p "Enter the output filename: " output_file

# view all environment variables and store in the output file
env | tee $output_file

echo "Environment variables have been stored in $output_file"
[root@localhost ~]# sh env.sh
Enter the output filename: env_output
HISTCONTROL=ignoredups
HOSTNAME=
HISTSIZE=1000
GUESTFISH_OUTPUT=\e[0m
PWD=/root
LOGNAME=root
TZ=UTC-05:30
GUESTFISH_RESTORE=\e[0m
HOME=/root
LANG=en_US.UTF-8

LS_COLORS=rs=0;di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=01;37;41:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:t=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.ha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;1:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:
```

```
.pbm=01;35:* .pgm=01;35:* .ppm=01;35:* .tga=01;35:* .xbm=01;35:* .xpm=01;35:* .tif=01
35:* .tiff=01;35:* .png=01;35:* .svg=01;35:* .svgz=01;35:* .mng=01;35:* .pcx=01;35:*
.ov=01;35:* .mpg=01;35:* .mpeg=01;35:* .m2v=01;35:* .mkv=01;35:* .webm=01;35:* .webp=0
;35:* .ogm=01;35:* .mp4=01;35:* .m4v=01;35:* .mp4v=01;35:* .vob=01;35:* .qt=01;35:* .n
v=01;35:* .wmv=01;35:* .ASF=01;35:* .rm=01;35:* .rmvb=01;35:* .flc=01;35:* .avi=01;35
*.fli=01;35:* .flv=01;35:* .gl=01;35:* .dl=01;35:* .xcf=01;35:* .xwd=01;35:* .yuv=01;
5:* .cgm=01;35:* .emf=01;35:* .ogv=01;35:* .ogx=01;35:* .aac=01;36:* .au=01;36:* .flac
01;36:* .m4a=01;36:* .mid=01;36:* .midi=01;36:* .mka=01;36:* .mp3=01;36:* .mpc=01;36
*.ogg=01;36:* .ra=01;36:* .wav=01;36:* .oga=01;36:* .opus=01;36:* .spx=01;36:* .xspf=0
;36:
GUESTFISH_PS1=\[\e[1;32m\]><fs>\[\e[0;31m\]
TERM=linux
LESSOPEN=||/usr/bin/lesspipe.sh %
USER=root
SHLVL=2
GUESTFISH_INIT=\e[1;34m
CVS_RSH=ssh
S_COLORS=auto
PATH=/usr/local/sbin:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin
MAIL=/var/spool/mail/root
OLDPWD=/
_=~/bin/env
Environment variables have been stored in env_output
[root@localhost ~]#
```

Lab Exercise - 10

1. Write a shell script named count which takes a file name as its parameter and prints number of blank lines in it. Also display the lines containing “is” as a word.
2. While executing a shell script either the LOGNAME or the UID is supplied at the command prompt. Write a shell script to find out at how many terminals has this user logged in.
3. Given a file of numbers (one per line), write a script that will find the lowest and highest number.
4. Write a shell script, which deletes all lines containing the word “unix” in the files, supplied as argument to this shell script.

Solutions of Lab Exercise - 10

1.

```
# Check if a file name is provided as an argument
if [ $# -eq 0 ]; then
    echo "Usage: $0 <filename>"
    exit 1
fi

filename="$1"

# Check if the file exists
if [ ! -e $filename ]; then
    echo "Error: File $filename does not exist."
    exit 1
fi

# Count the number of blank lines
blank_lines=`grep -c '^$' $filename`

echo "Number of blank lines in $filename: $blank_lines"

# Display lines containing 'is' as a word
echo "Lines containing 'is' as a word in $filename:"
grep -w "is" "$filename"
```

```
[root@localhost ~]# grep -c "^$" sample.txt
2
[root@localhost ~]# sh count.sh
Usage: count.sh <filename>
[root@localhost ~]# sh count.sh sample.txt
Number of blank lines in sample.txt: 2
Lines containing 'is' as a word in sample.txt:
This is an example line.
[root@localhost ~]#
```

2.

```
#!/bin/bash

# Check if either LOGNAME or UID is supplied
if [ -z "$LOGNAME" ] && [ -z "$UID" ]; then
    echo "Error: Neither LOGNAME nor UID is supplied."
    exit 1
fi

# Determine the user
user=""
if [ -n "$LOGNAME" ]; then
    user="$LOGNAME"
elif [ -n "$UID" ]; then
    user=`id -un "$UID"`
fi

# Count the number of terminals the user is logged into
terminals_count=$(who | grep -c "^$user")

echo "User $user is logged into $terminals_count terminal(s)."
```

```
[root@localhost ~]# sh users.sh root
User  is logged into 1 terminal(s).
[root@localhost ~]#
```

3.

```
if [ $# -eq 0 ];
then
echo "Filename not entered"
exit 1
fi

# sort the contents of the file and save to the same file
if [ $# -eq 0 ];
then
echo "Filename not entered"
exit 1
fi

# sort the contents of the file and save to the same file
sort "$1" -o "$1"

# Display minimum and maximum values
echo "Minimum: `head -1 $1`"
echo "Maximum: `tail -1 $1`"
```

```
[root@localhost ~]# sh ques3.sh numbers.txt
Minimum: 12
Maximum: 75
[root@localhost ~]# cat numbers.txt
12
20
34
43
56
66
75
[root@localhost ~]#
```

4.

```
#!/bin/bash

# Check if filenames are provided as arguments
if [ "$#" -eq 0 ]; then
    echo "Usage: $0 <file1> [<file2> ...]"
    exit 1
fi

# Loop through each filename provided as an argument
for file in "$@"; do
    # Check if the file exists
    if [ ! -e "$file" ]; then
        echo "Error: File '$file' does not exist."
    else
        # Use sed to delete lines containing the word "unix"
        sed -i '/unix/d' "$file"
        echo "Lines containing 'unix' deleted from $file."
    fi
done
```

```
[root@localhost ~]# sh file4.sh
Usage: file4.sh <file1> [<file2> ...]
[root@localhost ~]# sh file4.sh unix.txt
Lines containing 'unix' deleted from unix.txt.
```

```
[root@localhost ~]# cat > unix.txt
This is a line containing unix.
Another line without the word.
This line also mentions unix.
A line with unix unix unix.
```

```
[root@localhost ~]# cat unix.txt
Another line without the word.
[root@localhost ~]#
```

Lab Exercise - 11

- (1) Ramesh's basic salary is input through the keyboard. His dearness allowance is 40% of basic salary, and house rent allowance is 20% of basic salary. WAP to calculate his gross salary.
- (2) The distance between two cities (in km) is input through the keyboard. WAP to convert and print this distance in meters, feet, inches and centimeters.
- (3) The length and breadth of a rectangle and radius of a circle are input through the keyboard. WAP to calculate the area and perimeter of the rectangle, and the area and circumference of the circle.
- (4) If a five digit number is input through the keyboard, WAP to calculate the sum of its digits. (Use the modulus operator ‘%’)
- (5) Write a shell script, which will automatically get executed, on logging in. This shell script should display the present working directory and report whether your friend whose logname is aa10 has currently logged in or not. If he has logged in then the shell script should send a message to his terminal suggesting a dinner tonight. If you do not have write permission to his terminal or if he hasn't logged in then such a message should be mailed to him with a request to send confirmation about your dinner proposal.

Solutions of Lab Exercise - 11

(1)

```
# salary.sh

read -p "Enter basic salary: " sal

# Use backquotes for command substitution
# salary.sh

read -p "Enter basic salary: " sal

# Use backquotes for command substitution
da=`expr $sal \* 40 / 100`
hr=`expr $sal \* 20 / 100`

echo "Dearness allowance is: $da"
echo "House rent allowance is: $hr"

# Calculate gross salary
gs=`expr $sal + $hr + $da` 

echo "Your gross salary is: $gs"
```

```
[root@localhost ~]# sh salary.sh
Enter basic salary: 10000
Dearness allowance is: 4000
House rent allowance is: 2000
Your gross salary is: 16000
[root@localhost ~]#
```

(2)

```
# Convert distance to meters, feet, inches, and centimeters using expr
#!/bin/bash
# Input distance in kilometers
echo "Enter distance between two cities in kilometers: "
read distance_km

# Conversion factors
meter_factor=1000
feet_factor=3280.84
inch_factor=39370.1
centimeter_factor=100000

# Convert distance to meters, feet, inches, and centimeters using expr
distance_meter=`echo "$distance_km * $meter_factor" | bc`
distance_feet=`echo "$distance_km * $feet_factor" | bc`
distance_inch=`echo "$distance_km * $inch_factor" | bc`
distance_centimeter=`echo "$distance_km * $centimeter_factor" | bc`

# Display the converted distances
echo "Distance in meters: $distance_meter m"
echo "Distance in feet: $distance_feet ft"
echo "Distance in inches: $distance_inch in"
echo "Distance in centimeters: $distance_centimeter cm"
```

```
[root@localhost ~]# sh distance.sh
Enter distance between two cities in kilometers:
10
Distance in meters: 10000 m
Distance in feet: 32808.40 ft
Distance in inches: 393701.0 in
Distance in centimeters: 1000000 cm
[root@localhost ~]#
```

(3)

```
#!/bin/bash
# Input length and breadth of the rectangle
read -p "Enter length of the rectangle: " length

read -p "Enter breadth of the rectangle: " breadth

# Input radius of the circle
read -p "Enter radius of the circle: " radius

# Calculate area and perimeter of the rectangle
rectangle_area=`echo "$length * $breadth" | bc`
rectangle_perimeter=`echo "2 * ($length + $breadth)" | bc`

# Calculate area and circumference of the circle
circle_area=`echo "3.1415 * $radius * $radius" | bc`
circle_circumference=`echo "2 * 3.1415 * $radius" | bc`
```

```
# Display the results
echo "Rectangle Area: $rectangle_area"
echo "Rectangle Perimeter: $rectangle_perimeter"
echo "Circle Area: $circle_area"
echo "Circle Circumference: $circle_circumference"
```

```
[root@localhost ~]# sh file3.sh
Enter length of the rectangle: 5.0
Enter breadth of the rectangle: 2.0
Enter radius of the circle: 5.0
Rectangle Area: 10.0
Rectangle Perimeter: 14.0
Circle Area: 78.5375
Circle Circumference: 31.4150
[root@localhost ~]#
```

(5)

```
#!/bin/bash

# Input length and breadth of the rectangle
read -p "Enter length of the rectangle: " length

read -p "Enter breadth of the rectangle: " breadth

# Input radius of the circle
read -p "Enter radius of the circle: " radius

# Calculate area and perimeter of the rectangle
rectangle_area=`echo "$length * $breadth" | bc`
rectangle_perimeter=`echo "2 * ($length + $breadth)" | bc`

# Calculate area and circumference of the circle
circle_area=`echo "3.1415 * $radius * $radius" | bc`
circle_circumference=`echo "2 * 3.1415 * $radius" | bc`

# Display the results
echo "Rectangle Area: $rectangle_area"
echo "Rectangle Perimeter: $rectangle_perimeter"
echo "Circle Area: $circle_area"
echo "Circle Circumference: $circle_circumference"
```

```
[root@localhost ~]# sh file3.sh
Enter length of the rectangle: 5.0
Enter breadth of the rectangle: 2.0
Enter radius of the circle: 5.0
Rectangle Area: 10.0
Rectangle Perimeter: 14.0
Circle Area: 78.5375
Circle Circumference: 31.4150
[root@localhost ~]#
```

Lab Exercise – 12

1. Write a script that will echo the third parameter, but only if it is present.
2. Write a script that would recognize if a word entered from the keyboard started with an upper or lower case character or a digit. Use the "case" statement.

Echo -e “Enter a word or no \c”

Read ans

Case \$ans in

[a-z] *) echo “lower case”;;

[A-Z] *) echo “upper”;;

[0-9] * echo “no”;;

*) echo “none”;;

Esac

Exit 0

The script would then output the word, followed by "upper case", "lower case", "digit", or "not upper, lower, or digit".

3. Write a program that will take an undetermined list of parameters, and reverse them.

If [\$# -eq 0]

Then

Echo “there were no arguments”

Exit

Fi

Args=” “

While [! \$# -eq 0]

Do

Args=”\$1 \$args”

Shift

Done

Echo “Reversed arguments are \$args”

4. Write a script that will accept any number of parameters. The program should display whether an odd or an even number of parameters was given.

Rem=`expr \$# % 2`

If [\$rem -eq 0]

Then

Echo “there were even no of arguments”

Else

Echo “there were odd no of arguments”

Fi

Exit 0

Cp \$1 \$2

5. Write a script asking the user to input some numbers. The script should stop asking for numbers when the number 0 is entered. The output should look like:

- i. user: logon_name
- ii. Lowest number entered:
- iii. Highest number entered:
- iv. Difference between the two:
- v. Product of the two:

Solutions of Lab Exercise – 12

1.

```
# Check if the third parameter is present
if [ $# -ge 3 ]; then
    echo "Third parameter: $3"
else
    echo "Third parameter is not present."
fi

[root@localhost ~]# sh third.sh arg1 arg2 arg3
Third parameter: arg3
[root@localhost ~]# sh third.sh arg1 arg2
Third parameter is not present.
[root@localhost ~]#
```

2.

```
#!/bin/bash

echo -e "Enter a word or number: \c"
read ans

case $ans in
[a-z]* ) echo "$ans is in lower case.";;
[A-Z]* ) echo "$ans is in upper case.";;
[0-9]* ) echo "$ans is a digit.";;
* ) echo "$ans is not upper case, lower case, or a digit.";;
esac

exit 0
```

```
[root@localhost ~]# sh word.sh
Enter a word or number: HELLO
HELLO is in upper case.
[root@localhost ~]# sh word.sh
Enter a word or number: hello
hello is in lower case.
[root@localhost ~]# sh word.sh
Enter a word or number: 1234abc
1234abc is a digit.
```

3.

```
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "There were no arguments."
    exit 1
fi

args=""

while [ $# -gt 0 ]; do
args="$1 $args"
shift
done

echo "Reversed arguments are $args"
```

```
[root@localhost ~]# sh reverse.sh
There were no arguments.
[root@localhost ~]# sh reverse.sh 1 2 3
Reversed arguments are 3 2 1
[root@localhost ~]# sh reverse.sh b c d a
Reversed arguments are a d c b
[root@localhost ~]#
```

4.

```
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "There were no arguments."
    exit 1
fi
rem=$(( $# % 2 ))

if [ $rem -eq 0 ]; then
    echo "There were even number of arguments."
else
    echo "There were odd number of arguments."
fi
exit 0
```

```
[root@localhost ~]# sh odd_even_arg.sh
There were no arguments.
[root@localhost ~]# sh odd_even_arg.sh arg1 arg2
There were even number of arguments.
[root@localhost ~]# sh odd_even_arg.sh arg1 arg2 arg3
There were odd number of arguments.
[root@localhost ~]#
```

5.

```
# Prompt the user for input
echo -e "Enter numbers (enter 0 to stop):"
# Initialize variables
read -p "Enter a number: " num
min=$num
max=$num
product=1
# Continue reading numbers until 0 is entered
while [ $num -ne 0 ]; do
    if [ $num -lt $min ]; then
        min=$num
    fi

    if [ $num -gt $max ]; then
        max=$num
    fi

    read -p "Enter a number: " num
done
# Display the results
echo -e "\nUser: $LOGNAME"
echo "Lowest number entered: $min"
echo "Highest number entered: $max"
echo "Difference between the two: $((max - min))"
echo "Product of the two: $((max * min))"
```

```
User: root
Lowest number entered: 10
Highest number entered: 25
Difference between the two: 15
Product of the two: 250
```

Lab Exercise -13

1. A shell script receives even number of filenames. Suppose four filenames are supplied then the first file should get copied into second file, the third file should get copied into fourth file, and so on. If odd number of filenames are supplied then no copying should take place and an error message should be displayed.
2. Write a shell script, which displays a list of all files in the current directory to which you have read, write and execute permissions.
3. Write a script that will take a person's name as a parameter to the program name. The script should greet that person, as

Good Day *name_entered*, How are you today?

If test \$# -eq 0

Then

Echo "no name on command line"

Exit 1

Fi

Echo "Good Day \$1, How are you today"

Exit 0

4. Write a series of scripts that will count the number of parameters on the command line, first using the `for` statement, then the `while` and finally the `until` statement. (Three scripts).

Num=0

For I in \$*

Do

Num=`expr \$num + 1`

Done

Echo "There were \$num arguments"

Exit 0

Num=0

While test \$# -gt 0

Do

Num=`expr \$num + 1`

shift

Done

Echo "There were \$num arguments"

Exit 0

Num=0

Until test \$# -eq 0

Do

Num=`expr \$num + 1`

shift

Done

Echo "There were \$num arguments"

Exit 0

5. Write a script that will return the number of parameters on the command line.

6. Write a script that will prompt the user for a name. That same name will then be displayed in the screen.

7. Write a script to find out biggest number from given three numbers. Numbers are supplied as command line argument. Print error if sufficient arguments are not supplied.

Solutions of Lab Exercise -13

1.

```
#!/bin/bash

# Check if the number of arguments is even
if [ `expr $# % 2` -eq 0 ]; then
    # Iterate over pairs of filenames and copy the first into the second
    while [ $# -gt 0 ]; do
        cp "$1" "$2"
        echo "Copied $1 to $2"
        shift 2
    done
else
    echo "Error: Odd number of filenames provided. No copying will take place."
    echo "Enter only even number of filenames."
fi
```

```
[root@localhost ~]# sh copy_files.sh file1
Error: Odd number of filenames provided. No copying will take place.
Enter only even number of filenames.
[root@localhost ~]# sh copy_files.sh file1 file2 file3 file4
Copied file1 to file2
Copied file3 to file4
```

```
[root@localhost ~]# cat file1 file2 file3 file4
This is file1.
This is file1.
This is file3.
This is file3.
```

2.

```
#!/bin/bash

echo "Files with read, write, and execute permissions:"
echo "-----"

for file in *; do
    if [ -f "$file" ] && [ -r "$file" ] && [ -w "$file" ] && [ -x "$file" ];
        then
    echo "$file"
        fi
done
```

```
[root@localhost ~]# sh permissions.sh
Files with read, write, and execute permissions:
-----
bench.py
copy_files.sh
file1
file2
file3
file4
hello.c
numbers.sh
permissions.sh
[root@localhost ~]#
```

3.

```
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "No name on the command line."
    exit 1
fi

echo "Good Day $1, How are you today?"
exit 0
```

```
[root@localhost ~]# sh greet_person.sh John
Good Day John, How are you today?
[root@localhost ~]#
```

4.

```
# for.sh
num=0
for arg in "$@"; do
    num=`expr $num + 1`
done

echo "There were $num arguments"
exit 0
```

```
[root@localhost ~]# sh for.sh arg1 arg2 arg3
There were 3 arguments
[root@localhost ~]#
```

```
# while.sh
num=0
while [ $# -gt 0 ]; do
    num=`expr $num + 1`
    shift
done
```

```
echo "There were $num arguments"
exit 0
```

```
[root@localhost ~]# sh while.sh file1 file2 file3 file4
There were 4 arguments
[root@localhost ~]#
```

```
# until.sh
num=0
until [ $# -eq 0 ]; do
    num=`expr $num + 1`
    shift
done

echo "There were $num arguments"
exit 0
```

```
[root@localhost ~]# sh until.sh 1 2 3 4 5
There were 5 arguments
[root@localhost ~]#
```

5.

```
# count_param.sh

echo "Number of parameters on the command line: $#"
```

```
[root@localhost ~]# sh count_param.sh arg1 arg2 arg3 arg4
Number of parameters on the command line: 4
[root@localhost ~]#
```

6.

```
echo -e "Enter name of user: \c"
read name
echo $name
[root@localhost ~]# sh user.sh
Enter name of user: root
root
[root@localhost ~]#
```

7.

```
# find_biggest.sh

if [ $# -lt 3 ]; then
    echo "Error: Insufficient arguments. Please provide three numbers."
    exit 1
fi
```

```
num1=$1
num2=$2
num3=$3

max=$num1

if [ $num2 -gt $max ]; then
    max=$num2
fi

if [ $num3 -gt $max ]; then
    max=$num3
fi

echo "The biggest number is: $max"
exit 0
```

```
[root@localhost ~]# sh find_biggest.sh
Error: Insufficient arguments. Please provide three numbers.
[root@localhost ~]# sh find_biggest.sh 1 25 13
The biggest number is: 25
[root@localhost ~]#
```