CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING

(AFFILIATED TO GURU GOBIND SINGH INDRAPRASTHA

UNIVERSITY)

MASTER OF COMPUTER APPLICATIONS
(Session: 2023-25)

OPERATING SYSTEMS WITH LINUX
LAB PRACTICAL FILE

Submitted to:
Ms. Rosy Verma

Submitted by:
Rishabh Sharma
MCA 1st Sem
(E.no: 05311804423)

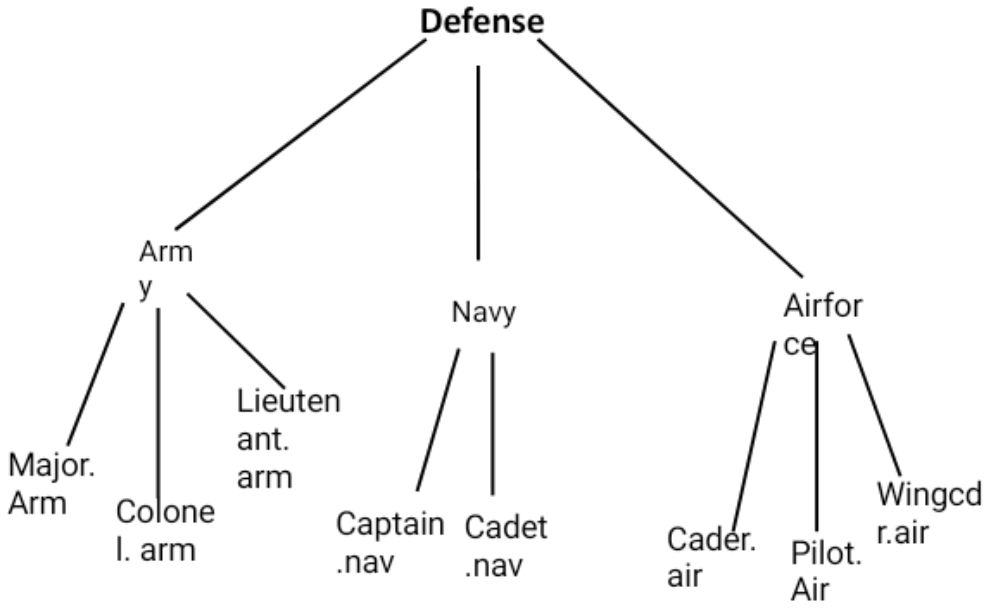
INDEX

S. NO	LAB EXERCISE	SIGN
1.	Lab Exercise 1	
2.	Lab Exercise 2	
3.	Lab Exercise 3	
4.	Lab Exercise 4	
5.	Lab Exercise 5	
6.	Lab Exercise 6	
7.	Lab Exercise 7	
8.	Lab Exercise 8	
9.	Lab Exercise 9	
10.	Lab Exercise 10	
11.	Lab Exercise 11	
12.	Lab Exercise 12	
13.	Lab Exercise 13	

Lab Exercise - 1

Q: 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:-

Q1.) :-

```
[RedHat@AzureVmRedHat airforce]$ echo $HOME
/home/RedHat
```

Q2.):-

```
[RedHat@AzureVmRedHat ~]$ mkdir defence
[RedHat@AzureVmRedHat ~]$ cd defence
[RedHat@AzureVmRedHat defence]$ mkdir army navy airforce
[RedHat@AzureVmRedHat defence]$ cd army
[RedHat@AzureVmRedHat army]$ touch major.arm colonel.arm lieutenant.arm
[RedHat@AzureVmRedHat army]$ cd ..; cd navy
[RedHat@AzureVmRedHat navy]$ touch captain.nav touch cadet.nav
[RedHat@AzureVmRedHat navy]$ cd ..; cd airforce
[RedHat@AzureVmRedHat airforce]$ touch cadet.air pilot.air wingcdr.air
```

Q3.):-

```
[RedHat@AzureVmRedHat army]$ cat > major.arm
All the second lieutenant works under the major. It is a high post in the army.
[RedHat@AzureVmRedHat army]$ cat > colonel.arm
Colonel is responsible for all the majors under his regiment.
[RedHat@AzureVmRedHat army]$ cat > lieutenant.arm
He is responsible to look after all the cadets.
[RedHat@AzureVmRedHat army]$ cat > captain.nav
One ship is managed by one captain.
[RedHat@AzureVmRedHat army]$ cat > cadet.nav
cadets are the trimness in NAvy.
[RedHat@AzureVmRedHat army]$ cat > cadet.air
Cader are the cadets of air force
[RedHat@AzureVmRedHat army]$ cat > pilot.air
Pilots manages the highest plain.
[RedHat@AzureVmRedHat army]$ cat > wingcdr.air
Wing cdr manages all pilots under his regiment
[RedHat@AzureVmRedHat army]$
```

Q4.):-

```
[RedHat@AzureVmRedHat army]$ cat major.arm
All the second lieutenant works under the major. It is a high post in the army.
[RedHat@AzureVmRedHat army]$ cat colonel.arm
Colonel is responsible for all the majors under his regiment.
[RedHat@AzureVmRedHat army]$ cat lieutenant.arm
He is responsible to look after all the cadets.
[RedHat@AzureVmRedHat army]$ cat captain.nav
One ship is managed by one captain.
[RedHat@AzureVmRedHat army]$ cat cader.nav
cat: cader.nav: No such file or directory
[RedHat@AzureVmRedHat army]$ cat cadet.nav
cadets are the trimness in NAvy.
[RedHat@AzureVmRedHat army]$ cat pilot.air
Pilots manages the highest plain.
[RedHat@AzureVmRedHat army]$ cat wingcdr.air
Wing cdr manages all pilots under his regiment
```

Q5.):-

```
[RedHat@AzureVmRedHat ~]$ cd defence/airforce
[RedHat@AzureVmRedHat airforce]$
```

Q6.):-

```
[RedHat@AzureVmRedHat ~]$ cd defence/airforce
[RedHat@AzureVmRedHat airforce]$
[RedHat@AzureVmRedHat airforce]$ mkdir operation reporting
```

Q7.):-

```
[RedHat@AzureVmRedHat airforce]$ cd reporting
[RedHat@AzureVmRedHat reporting]$
[RedHat@AzureVmRedHat reporting]$ mkdir secretary sergeant
```

Q8.):-

```
[RedHat@AzureVmRedHat ~]$ cd defence
[RedHat@AzureVmRedHat defence]$ mkdir airforce/operation/wings airforce/operation/groups airf
orce/operation/squadron
[RedHat@AzureVmRedHat defence]$ cd airforce/operation
[RedHat@AzureVmRedHat operation]$ ls
groups  squadron  wings
[RedHat@AzureVmRedHat operation]$
```

Q9.):-

```
[RedHat@AzureVmRedHat defence]$ cat >> navy/captain.nav
This is new data to be appended from defence directory
[RedHat@AzureVmRedHat defence]$ cd navy; cat captain.nav
One ship is managed by one captain.
This is new data to be appended from defence directory
[RedHat@AzureVmRedHat navy]$
```

Q10.):-

```
[RedHat@AzureVmRedHat defence]$ cp army/major.arm airforce/wingcdr.air
[RedHat@AzureVmRedHat defence]$
```

Q11.):-

```
[RedHat@AzureVmRedHat defence]$ cd airforce; cat wingcdr.air
All the second lieutenant works under the major. It is a high post in the army.
[RedHat@AzureVmRedHat airforce]$
```

Q12.):-

```
[RedHat@AzureVmRedHat defence]$ cat army/colonel.arm airforce/cadet.air > navy/test
[RedHat@AzureVmRedHat defence]$ cd navy; cat test
Colonel is responsible for all the majors under his regiment.
Cader are the cadets of air force
```

Q13.):-

```
[RedHat@AzureVmRedHat defence]$ cp navy/test navy/groups/tmpfile
[RedHat@AzureVmRedHat defence]$ cd navy/groups; cat tmpfile
Colonel is responsible for all the majors under his regiment.
Cader are the cadets of air force
```

Q14.):-

```
[RedHat@AzureVmRedHat navy]$ ls
cadet.nav  captain.nav  groups  test
[RedHat@AzureVmRedHat navy]$ rm -i test
rm: remove regular file 'test'? y
```

Lab Exercise - 2

Refer to exercise 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:-

Q15.):-

```
[admin@localhost Defense]$ mv Airforce/Operating/Groups/Tmpfile Airforce/Reporting/Sergeant
[admin@localhost Defense]$ cd Airforce/Reporting/Sergeant
[admin@localhost Sergeant]$ ls
Tmpfile
[admin@localhost Sergeant]$ _
```

Q16.):-

```
[admin@localhost Sergeant]$ mv Tmpfile Subordinate.air
[admin@localhost Sergeant]$ ls
Subordinate.air
```

Q17.):-

```
[admin@localhost Navy]$ mkdir Navalchief
[admin@localhost Navy]$ ls
Cadet.nav  Captain.nav  Navalchief
```

Q18.):-

```
[admin@localhost Defense]$ mv Navy/Navalchief Airforce/Reporting
[admin@localhost Defense]$ cd Airforce/Reporting
[admin@localhost Reporting]$ ls
Navalchief  Secretary  Sergeant
```

Q19.):-

```
[admin@localhost Reporting]$ mv Navalchief Airchief
[admin@localhost Reporting]$ ls
Airchief  Secretary  Sergeant
```

Q20.):-

```
[admin@localhost Reporting]$ ls
Airchief  Secretary  Sergeant
```

Q21.):-

```
[admin@localhost Reporting]$ who
admin    tty1         2023-12-12 10:08 (:0)
admin    pts/0        2023-12-12 10:23 (:0.0)
```

```
[admin@localhost Reporting]$ echo $SHELL
/bin/bash
```

Q22.):-

```
[admin@localhost Reporting]$ tty
/dev/pts/0
```

Q23.):-

```
[admin@localhost Defense]$ touch Empty1 Empty2 Empty3 Empty4 Empty5
[admin@localhost Defense]$ ls
Airforce  Army  Empty1  Empty2  Empty3  Empty4  Empty5  Navy
```

Q24.):-

```
[admin@localhost ~]$ ls -o
total 36
drwxrwxr-x. 5 admin 4096 Dec 12 08:57 Defense
drwxr-xr-x. 2 admin 4096 Dec 12  2023 Desktop
drwxr-xr-x. 2 admin 4096 Dec 12  2023 Documents
drwxr-xr-x. 2 admin 4096 Dec 12  2023 Downloads
drwxr-xr-x. 2 admin 4096 Dec 12  2023 Music
drwxr-xr-x. 2 admin 4096 Dec 12  2023 Pictures
drwxr-xr-x. 2 admin 4096 Dec 12  2023 Public
drwxr-xr-x. 2 admin 4096 Dec 12  2023 Templates
drwxr-xr-x. 2 admin 4096 Dec 12  2023 Videos
```

Q25.):-

```
[admin@localhost ~]$ who > oldfile
[admin@localhost ~]$ cat oldfile >> newfile
[admin@localhost ~]$ cat oldfile
admin    tty1         2023-12-12 10:08 (:0)
admin    pts/0        2023-12-12 10:23 (:0.0)
[admin@localhost ~]$ cat newfile
admin    tty1         2023-12-12 10:08 (:0)
admin    pts/0        2023-12-12 10:23 (:0.0)
```

Q26.):-

```
[admin@localhost ~]$ ls a*
art
```

```
[admin@localhost ~]$ ls *.*
ls: cannot access *.*: No such file or directory
```

```
[admin@localhost ~]$ ls [!abc]art
part  tart
```

```
[admin@localhost ~]$ ls [a!bc]art
bash: !bc]art: event not found
```

```
[admin@localhost ~]$ ls [b-dku-z]
ls: cannot access [b-dku-z]: No such file or directory
```

Q27.):-

```
[admin@localhost ~]$ rm -r dir1
```

Q28.):-

```
[admin@localhost Defense]$ touch .hidden1
[admin@localhost Defense]$ ls
Airforce  Army  Empty1  Empty2  Empty3  Empty4  Empty5  Navy
```

Q29.):-

```
[admin@localhost Defense]$ ls -a
.  ..  Airforce  Army  Empty1  Empty2  Empty3  Empty4  Empty5  .hidden1  Navy
```

Q30.):-

```
[admin@localhost Defense]$ ls e*
ls: cannot access e*: No such file or directory
```

Q31.):-

```
[admin@localhost Defense]$ ls ?ing
ls: cannot access ?ing: No such file or directory
```

Q32.):-

```
[admin@localhost Navy]$ ls [abc]*
ls: cannot access [abc]*: No such file or directory
```

Q33.):-

```
[admin@localhost Navy]$ ls ^[abc]*
ls: cannot access ^[abc]*: No such file or directory
```

Q34.):

```
[admin@localhost Army]$ ls [a-z][aeiou]?
ls: cannot access [a-z][aeiou]?: No such file or directory
```

Q35.):-

```
[admin@localhost Navy]$ ls -l
total 8
-rw-rw-r--. 1 admin admin  37 Dec 12 06:14 Cadet.nav
-rw-rw-r--. 1 admin admin 107 Dec 12 06:44 Captain.nav
```

Q36.):-

```
[admin@localhost Army]$ ls [a-z][0-9]*.[co]
ls: cannot access [a-z][0-9]*.[co]: No such file or directory
```

Q37.):-

```
[admin@localhost Defense]$ mv Army/*.[co] Airforce/Reporting/Secretary/Olddir
[admin@localhost Defense]$ cd Airforce/Reporting/Secretary/Olddir
[admin@localhost Olddir]$ ls
lol.c  xd.o
```

Q38.):-

```
[admin@localhost Secretary]$ rm -r Olddir
```

Q39.):-

```
[admin@localhost ~]$ echo "A line is any group of characters not containing a newline." >> Empty1
[admin@localhost ~]$ echo "A word is a group of characters not containing a space, tab or newline." >> Empty1
[admin@localhost ~]$ echo "A character is the smallest unit of information, and includes a space, tab and newline." >> Empty1
[admin@localhost ~]$ 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 newline.
A character is the smallest unit of information, and includes a space, tab and newline.
```

Q40.):-

```
[admin@localhost ~]$ wc -l Empty1
3 Empty1
[admin@localhost ~]$ wc -w Empty1
40 Empty1
[admin@localhost ~]$ wc -c Empty1
220 Empty1
```

Q41.):-

```
[admin@localhost ~]$ cat Empty1 > infile
[admin@localhost ~]$ 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 newline.
A character is the smallest unit of information, and includes a space, tab and newline.
```

Q42.):-

```
[admin@localhost ~]$ cmp Empty1 infile
[admin@localhost ~]$ _
```

Q43.):-

```
[admin@localhost ~]$ echo "wc offers three options to make a specific count" >> infile
[admin@localhost ~]$ 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 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
```

Q44.):-

```
[admin@localhost ~]$ cmp Empty1 infile
cmp: EOF on Empty1
```

Q45.):-

```
[admin@localhost ~]$ 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
```

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:-

I.

A.): Output-

```
[RedHat@AzureVmRedHat lab3]$ touch pacetalk pacemon
[RedHat@AzureVmRedHat lab3]$ ls -d [p]* | cat > file1
[RedHat@AzureVmRedHat lab3]$ cat file1
pacemon
pacetalk
```

B.): Output-

```
[RedHat@AzureVmRedHat lab3]$ ls p*|more
pacemon
pacetalk
```

C.): Output-

```
[RedHat@AzureVmRedHat lab3]$ cat >> a.txt
one
two
three
[RedHat@AzureVmRedHat lab3]$ cat >> b.txt
four
five
six
[RedHat@AzureVmRedHat lab3]$ cat >> c.txt
seven
eight
nine
[RedHat@AzureVmRedHat lab3]$ cat a.txt b.txt c.txt | sort
eight
five
four
nine
one
seven
six
three
two
```

D.): Output-

```
[RedHat@AzureVmRedHat lab3]$ ls -l | tail -20 | cat > profile
[RedHat@AzureVmRedHat lab3]$ cat profile
total 16
-rw-rw-r--. 1 RedHat RedHat 14 Jan 15 11:45 a.txt
-rw-rw-r--. 1 RedHat RedHat 14 Jan 15 11:45 b.txt
-rw-rw-r--. 1 RedHat RedHat 17 Jan 15 11:46 c.txt
-rw-rw-r--. 1 RedHat RedHat 17 Jan 15 11:43 file1
-rw-rw-r--. 1 RedHat RedHat  0 Jan 15 11:43 pacemon
-rw-rw-r--. 1 RedHat RedHat  0 Jan 15 11:43 pacetalk
-rw-rw-r--. 1 RedHat RedHat  0 Jan 15 11:47 profile
```

E.): Output-

```
[RedHat@AzureVmRedHat lab3]$ cat > FILE1
one
two
three
four
five
[RedHat@AzureVmRedHat lab3]$ cat > FILE2
six
seven
eight
nine
ten
[RedHat@AzureVmRedHat lab3]$ head -n 3 FILE1 | cat >> FILE2
[RedHat@AzureVmRedHat lab3]$ cat FILE2
six
seven
eight
nine
ten
one
two
three
[RedHat@AzureVmRedHat lab3]$ head -n 3 FILE2 | cat >> FILE1
[RedHat@AzureVmRedHat lab3]$ cat FILE1
one
two
three
four
five
six
seven
eight
```

F.): Output-

```
[RedHat@AzureVmRedHat lab3]$ cat > FILE1
I am studing OS
[RedHat@AzureVmRedHat lab3]$ cat > FILE2
I am learning Java
[RedHat@AzureVmRedHat lab3]$ cat FILE1 FILE2 > FILE3
[RedHat@AzureVmRedHat lab3]$ cat FILE3
I am studing OS
I am learning Java
```

Solutions:-

II.

A.): Output-

```
localhost:~/filter# grep -e "woodhouse" -e "wodehouse" filtercmd.lst
parry wodehouse
parry g wodehouse
parry woodhouse
```

B.): Output-

```
localhost:~/filter# grep -e "truman" -e "trueman" filtercmd.lst
alan trueman 80000
alan truman
```

C.): Output-

```
localhost:~/filter# grep -e "wilcox" -e "wilcocks" filtercmd.lst
silver wilcox 70000
sliver wilcocks 76567
```

D.): Output-

```
localhost:~/filter# grep "[p].*woodhouse" filtercmd.lst
parry woodhouse
```

E.): Output-

```
localhost:~/filter# grep "^2" filtercmd.lst
2 sliver
2 damon
```

F.): Output-

```
localhost:~/filter# grep -v "^2" filtercmd.lst
parry wodehouse
parry g wodehouse
parry woodhouse
silver wilcox 70000
sliver wilcocks 76567
alan trueman 80000
alan truman
```

G.): Output-

```
localhost:~/filter# grep "7....$" filtercmd.lst
silver wilcox 70000
sliver wilcocks 76567
```

H.): Output-

```
localhost:~/filter# grep -w "p.g.woodhouse" filtercmd.lst
```

I.): Output-

```
localhost:~/filter# ls -l|grep '^d'
drwxr-xr-x    2 root     root            37 Dec 14 10:46 abc
drwxr-xr-x    2 root     root            37 Dec 14 10:46 def
```

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:-

1.

(i):

```
[RedHat@AzureVmRedHat test]$ ls foo*
foo1  foo2  foo5
```

(ii):

```
[RedHat@AzureVmRedHat test]$ ls quit.?
quit.c  quit.h  quit.o
```

(iii):

```
[RedHat@AzureVmRedHat test]$ ls tutorial.*
tutorial.pdf  tutorial.ps
```

(IV):

```
[RedHat@AzureVmRedHat test]$ ls .*.swp
.test2.swp  .test3.swp  .test.swp
```

2.

Code:- $ ls [0-9]*[!a-z]

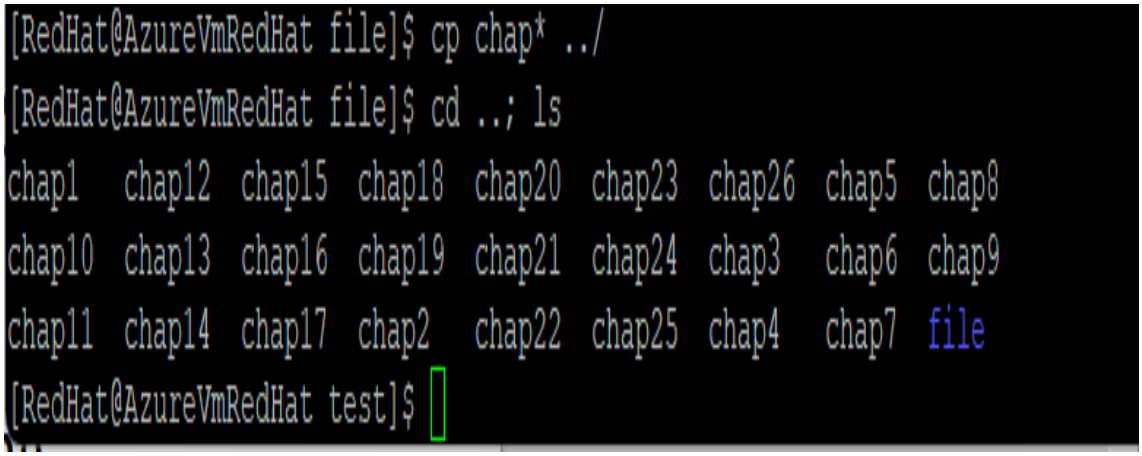
Output:

```
[RedHat@AzureVmRedHat test]$ ls [0-9]*[!a-z]
3test6  7test1
```

3.

Code:- $ cp chap* ../

Output:



4. grep john < file1 > result.

5. The first command will forward the result into file1 and second command will display the number is lines in file1.

LAB Exercise - 5

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

2. You are given a file myfile. Without opening this file how would you make a fair estimate about its contents?

3 . What will be the effect of following commands?
(i) umask
(ii)chmod u+w g-w abcd.out
(iii) chmod 777 aaa.c
(iv) chmod ug+rw a=x ffff.out
(v) chmod u+t mydir

Solutions:-

Q1):-

cat<file1>file2 :- The first part of redirection < file1 indicates that the input is to be taken from the file file1 and the second part of redirection > file2 establishes that output is to be rooted to the file file2.

cat>file2<file1:- The first part of redirection > file2 indicates that the output is to be routed to the file file2 and the second part of redirection < file1 establishes that input is taken from the file file1

Q2):-

This can be done using the “wc” command.

Output:

```
system@radeon:~$ wc empty1
  3  40 219 empty1
system@radeon:~$
```

Q3):-

(i) When a new file is created, its permissions are determined by a value called the umask. The value umask is 022. Its value is subtracted from the value 777 (for directories) or 666 (for files). Thus, a umask value of 022 yields new files with mode 644, and a directory with mode 755.

(ii) It means user have writing permission but writing permission was revoked from the group.

(iii) Everyone has all rights to read, write and execute.

(iv) Everyone has the execute permission, and only user and group have the permission to read and write too.

(v) Only user can delete or rename the file “mydir”, and group and others

cannot do the same.

Output:

(i)

```
system@radeon:~$ umask
0022
system@radeon:~$
```

(ii)

```
system@radeon:~$ chmod u+w abcd.out
system@radeon:~$ chmod g-w abcd.out
system@radeon:~$ ls -l abcd.out
-rw-r--r-- 1 system system 0 Feb  6 12:07 abcd.out
```

(iii)

```
system@radeon:~$ chmod u+w abcd.out
system@radeon:~$ chmod g-w abcd.out
system@radeon:~$ ls -l abcd.out
-rw-r--r-- 1 system system 0 Feb  6 12:07 abcd.out
```

(iv)

```
system@radeon:~$ chmod ug+rw fff.out
system@radeon:~$ chmod a=x fff.out
system@radeon:~$ ls -l fff.out
---x--x--x 1 system system 0 Feb  6 12:27 fff.out
```

(v)

```
system@radeon:~$ chmod u+t mydir
system@radeon:~$ ls -l mydir
-rw-r--r-- 1 system system 0 Feb  6 12:27 mydir
system@radeon:~$
```

LAB Exercise - 6

1. Using sed command, write down the commands for the followings
   What Regular Expression would you use to do the following:

(i) List all the records having either woodhouse or wodehouse.
(ii) List all the records having either trueman or truman.
(iii) List all the records having Wilcox or wilcocks.
(iv) List the records having first name as p. followed by any numbers of characters and last name as woodhouse.
(v) List the records which begin with employee code 2…..
(vi) Inverse your regular expression i.e. list all except the ones starting with2.
(vii) List all records having salary between 70000 to 79999.
(viii) List records having p.g. woodhouse as name
(ix) List only directories in your current directories.

Solutions:-

Q1):-

(i) sed -n '/wo[od][de]house/p' employee.txt

(ii) sed -n -e '/truman/p' -e '/trueman/p' employee.txt

(iii) sed -n -e '/wilcox/p' -e '/wilcocks/p' employee.txt

(iv) sed -n '/p\..*woodhouse/p' employee.txt

(v) sed -n '/^[2]/p' employee.txt

(vi) sed -n '/^[^2]/p' employee.txt

(vii) sed -n '/7[0-9][0-9][0-9][0-9]$/p' employee.txt

(viii) sed -n '/p\..*woodhouse/p' employee.txt

OR

sed -n '/p.g. woodhouse/p' employee.txt

(ix) ls -l|sed -n '/^d/p'

Output:

(I)

```
system@radeon:~$ sed -n '/wo[od][de]house/p' employee.txt
1. parry woodhouse 65000
3. demi wodehouse 76000
7. p.g. woodhouse 75000
```

(ii)

```
system@radeon:~$ sed -n -e '/truman/p' -e '/trueman/p' employee.txt
4. harry truman 58000
5. george trueman 78000
```

(iii)

```
system@radeon:~$ sed -n -e '/wilcox/p' -e '/wilcocks/p' employee.txt
2. ron wilcox 72000
6. fred wilcocks 79000
```

(iv)

```
system@radeon:~$ sed -n '/p\..*woodhouse/p' employee.txt
7. p.g. woodhouse 75000
```

(v)

```
system@radeon:~$ sed -n '/^[2]/p' employee.txt
2. ron wilcox 72000
```

(vi)

```
system@radeon:~$ sed -n '/^[^2]/p' employee.txt
Table format: emp_ID, emp_name, emp_salary
1. parry woodhouse 65000
3. demi wodehouse 76000
4. harry truman 58000
5. george trueman 78000
6. fred wilcocks 79000
7. p.g. woodhouse 75000
```

(vii)

```
system@radeon:~$ sed -n '/7[0-9][0-9][0-9][0-9]$/p' employee.txt
2. ron wilcox 72000
3. demi wodehouse 76000
5. george trueman 78000
6. fred wilcocks 79000
7. p.g. woodhouse 75000
```

(viii)

```
system@radeon:~$ sed -n '/p\..*woodhouse/p' employee.txt
7. p.g. woodhouse 75000
```

(ix)

```
system@radeon:~$ ls -l|sed -n '/^d/p'
drwxr-xr-x 1 system system 4096 Jan 18 11:50 airforce.air
drwxr-xr-x 1 system system 4096 Jan 18 11:51 army
drwxr-xr-x 1 system system 4096 Jan 18 11:50 defence
drwxr-xr-x 1 system system 4096 Jan 18 11:51 navy
drwxr-xr-x 1 system system 4096 Feb  1 11:13 olddir
```

LAB EXERCISE - 7

Question No. 1
Write a program to verify how many users are logged on to the system.

Question No. 2
Write a program that will calculate the amount of disk space your directory is using on the system.

Question No. 3
Write Script to see current date, time, username, and current directory

Question No. 4
How to calculate 5.12 + 2.5 real number calculation at $ prompt in Shell ?

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 Lab-7

Ans-1)
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
```

Ans-2)
diskspace.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]#
```

Ans-3)
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]#
```

Ans-4)

```
# 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 ~]#
```

Ans-5)

```
# 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 a 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
3) Your home directory
4) Your operating system type
5) Your current path setting
6) Your current working directory
7) Show Currently logged number of users
8) About your os and version ,release number , kernel version
9) Show all available shells
10) Show mouse settings
11) Show computer cpu information like processor type, speed etc
12) Show memory information meminfo
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 the 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 the same or not. If they are the same then the second file should be deleted.

Solutions Lab-8

Ans-1)

```
# 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
 Hello World
 Hello World
 Hello World
 Hello World
 Hello World
 Hello World
[root@localhost ~]#
```

Ans-2)

```
# 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-ad
ministrators-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
```

Ans-3)

```
echo "Ente 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 ~]#
```

Ans-4)

```
# 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 ~]#
```

Ans-5)

```
# 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 ~]#
```

Ans-6)

```
# 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

Q1. 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.

Q2. 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.

Q3. Write a shell script to print only those lines of file that do not contain word “example”, also store output in a file.

Q4. Write a shell script to view all environment variables and store it in a file.

Solutions Lab-9

Ans-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 ~]#
```

Ans-2)

```
# Accept a file name from the user
read -p "Enter a file name: " filename
# 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 ~]#
```

Ans-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: " outputfile

# 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: " outputfile

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

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

```
[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 ~]#
```

Ans-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:c
=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;3
:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=0
;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.e
r=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;3
:*.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 %s
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

Q1. Write a shell script named count which takes a file name as its parameter and prints a number of blank lines in it. Also display the lines containing “is” as a word.

Q2. 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.

Q3. Given a file of numbers (one per line), write a script that will find the lowest and highest number.

Q4. Write a shell script, which deletes all lines containing the word “unix” in the files, supplied as arguments to this shell script.

Solutions Lab-10

Ans-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 ~]#
```

Ans-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 ~]#
```

Ans-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 ~]#
```

Ans-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

Q1) 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.

Q2) 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.

Q3) 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.

Q4) If a five digit number is input through the keyboard, WAP to calculate the sum of its digits. (Use the modulus operator '%')

Q5) 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 Lab-11

Ans-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 allowence is: 4000
House rent allowence is: 2000
Your gross salary is: 16000
[root@localhost ~]#
```

Ans-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 ~]#
```

Ans-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 ~]#
```

Ans-4)

```
# 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

# Input a number
read -p "Enter a number: " num
```

```
# Initialise sum to zero
sum=0

# Loop to extract & sum the digits
while [ $num -gt 0 ]; do
    digit=`expr $num % 10`
    sum=`expr $sum + $digit`
    num=`expr $num / 10`
done

echo "Sum of digits: $sum"
```

```
[root@localhost ~]# sh sum.sh
Enter a number: 12345
Sum of digits: 15
[root@localhost ~]#
```

Ans-5)

```
#!/bin/bash

# Get the present working directory
current_directory=$(pwd)

# Check if user aa10 is currently logged in
if who | grep -q "aa10"; then
    # User aa10 is logged in
    echo "Your friend (aa10) is currently logged in."

    # Send a message to the terminal
```

```
    echo "Hello aa10! How about dinner tonight?" > /dev/pts/`who | grep "aa10" |
 awk '{print $2}'`
else
    # User aa10 is not logged in
    echo "Your friend (aa10) is not currently logged in."

    # Send a mail to user aa10
    echo "Hello aa10! How about dinner tonight?" | mail -s "Dinner Proposal"
aa10@localhost
  fi
```

```
[root@localhost ~]# sh invite.sh
Your friend (aa10) is not currently logged in.
[root@localhost ~]# /usr/sbin/sendmail: No such file or directory
"/root/dead.letter" 9/258
. . . message not sent.
```

LAB EXERCISE - 12

Q1) Write a script that will echo the third parameter, but only if it is present.

Q2) 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.

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

Q3) Write a program that will take an undetermined list of parameters, and reverse them.

Q4) 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.

Q5) 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: login_name
ii) Lowest number entered:
iii) Highest number entered:
iv) Difference between the two:
v) Product of the two:

Solutions Lab-12

Ans-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 ~]#
```

Ans-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.
```

Ans-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 ~]#
```

Ans-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 ~]#
```

Ans-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

Q1) A shell script receives an even number of filenames. Suppose four filenames are supplied then the first file should get copied into the second file, the third file should get copied into the fourth file, and so on. If an odd number of filenames are supplied then no copying should take place and an error message should be displayed.

Q2) Write a shell script, which displays a list of all files in the current directory to which you have read, write and execute permissions.

Q3) 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?

Q4) 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).

Q5) Write a script that will return the number of parameters on the command line.

Q6) Write a script that will prompt the user for a name. That same name will then be displayed on the screen.

Q7) Write a script to find out the biggest number from the given three numbers. Numbers are supplied as command line arguments. Print error if sufficient arguments are not supplied.

Solutions Lab-13

Ans-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.
```

Ans-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 ~]#
```

Ans-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 ~]#
```

Ans-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 ~]#
```

Ans-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 ~]#
```

Ans-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 ~]#
```

Ans-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 ~]#
```