LINUX LAB MANUAL
COURSE CODE: 15CS47P

FOR 4th Sem CS & E

(2017-18)

By

Mrs. RENU HARSHA

HEAD OF THE DEPARTMENT (HOD)

COMPUTER SCIENCE & ENGINEERING

RJS POLYTECHNIC

BANGALORE-34.

For Any Feedback Contact

Email: Velumani2296@gmail.com

<u>List of Graded Practical Exercises</u>

PART-A

1. **Introduction-** Linux Architecture- Shell, Kernel, System calls. Linux installation- Steps for installing Linux Operating System
 Comparison between Linux and other Operating Systems, Applications of Linux Operating System.

2. **Internal & External commands in Linux.** ∑ Internal commands- echo, type, etc.
   ∑ External commands- ls, cp, mv, rm, cat, etc

   ∑ Other commands – tput clear, who, cal, date, bc, man, passwd, uname (with different options).

3. **Working with files & directories**.
   ∑ Know the categories of files.

   ∑ Directory related Commands – pwd, mkdir, rmdir, cd, ls

   ∑ Manipulating Absolute paths and Relative paths using **cd** command.

   ∑ File related Commands – cat, cp, mv, rm, comm, cmp, diff, tar, umask, wc

4. **Basic File attributes.**
   ∑ Listing seven attributes of a file : ls and its options

   ∑ File Permissions: Absolute and Relative permissions

   ∑ Manipulating File permissions using **chmod** command

   ∑ Manipulating File Ownership using **chown** command
   ∑ Manipulating Hardlink and Softlink using **ln** command

5. **Learn to use vi editor.**
   ∑ Three modes of **vi** editor.

   ∑ Input mode commands.

   ∑ Command mode commands.

∑ Ex mode commands.

6. **Simple Filters** – head, tail, cut, paste, sort, uniq, tr, pr.

7. **Expressions & search patterns** .(dot operator), *, ^, +, ?, grep, egrep, fgrep 8. **Process Management commands**.
   - ∑ Process creation, status, Identifying process, ps -f & its options,
   - ∑ Running process in background, Job control, and Process termination.
   - ∑ Changing process priority, scheduling process (Usage of sleep and wait commands)

9. **Introduction to shell programming.**
   - ∑ Introduction, Uses of shell script, Shell special characters, comments, command separator, escaping, quoting command substitution.
   - ∑ Creating shell script, Shell identifiers, Shell variables, Destroying a variable, Positional parameters & command line arguments.
   - ∑ Evaluating expressions, Text formatting with echo & tput script termination.

10. **Shell control structures**
   ∑ if, case, for, while, relational and logical operators, ∑
   Advanced filter – sed and awk.

11. **Linux system administration**
   Managing file system, Disk management utilities, mounts,
   umount, df, du, fdisk, su, useradd etc.

12. **Linux Environment**
   Introduction, Environment variables, Command prompt system variables, Profiles, files, terminal variable stty command and its options, Command history, editing Environment variable.

PART – B

13. Write a shell script to display current date, time, username and directory.
14. Write script to determine whether given file exist or not, file name is supplied as command line argument, also check for sufficient number of command line argument
15. Write shell script to show various system configuration like:
    a) Currently logged user name and his long name
    b) Current shell
    c) Your home directory
16. Write shell script to show various system configuration like:
    a) Your operating system type
    b) Your current path setting
    c) Your current working directory
    d) Show all available shells
17. Write a Shell script to accept any two file names and check their file permissions.
18. Write a Shell script to read a file name and change the existing file permissions.
19. Write a shell script to print current month calendar and to replace the current day number by ‘*’or ‘**’ respectively.
20. Write a C-program to fork a child process and execute the given Linux commands.
21. Write a C-program to fork a child process, print owner process ID and its parent process ID.
22. Write a C-program to prompt the user for the name of the environment variable, check its validity and print an appropriate message.

PART- A

COMMANDS

Internal & External commands in Linux

Internal commands

1)Echo

Command:echo
Syntax:echo “arguments”
Purpose: Displays the given text on screen
EX:echo “welcome to linux lab”

OUTPUT:

```
[sudha@rjsplinux ~]$ echo “welcome to linux lab”
Welcome to linux lab”
[sudha@rjsplinux ~]$
```

2)Type

Command:type
Syntax:type arguments/command
Purpose: It is used to know the location of the executable program
Ex:type echo

OUTPUT:

```
[sudha@rjsplinux ~]$ type echo
Echo is shell builtin
[sudha@rjsplinux ~]$ type who
Who is /usr/bin/who
```

External commands

1)LS

Command:ls
Syntax:ls
Purpose: This command displays all the files and directories.
EX:ls

OUTPUT:

```
[sudha@rjsplinux ~]$ ls
372CS14041.sh   dad  sudha   parrot   sudu1.sh   sundisplaymenu.sh
manjupathname.sh   sudha.g   3star   dot      logname.sh   pgm1.sh
sun.pineapple   end   mango   pgm7.sh   sun11   gowriganapathi
[sudha@rjsplinux ~]$
```

Options of LS

1)-x

Command:-X
Syntax:ls –x
Purpose: It displays in multi columnar output.

2)-f

Command:-f
Syntax:ls –f
Purpose: List the files in long format.

3)-a

Command:-a
Syntax:ls –a
Purpose: List all entries including hidden files.

4)-p

Command:-p
Syntax:ls –p
Purpose:puts a slash after each directory

5)-r

Command:-r
Syntax:ls –r
Purpose:sorts filename in reverse order.

6)-u

Command:-u
Syntax:ls –u
Purpose:sorts the filename by last access time.

7)-l

Command:-l
Syntax:ls –l
Purpose: Displays one filename in each line.

2)CP (Copying a file)

COMMAND:CP
PURPOSE: This command copies a file (or) a group of files.It creates an exact of a file on disk with different name.
SYNTAX:cp source_file_name destination_file_name
EX:cp xyz abc
OUTPUT:

```
[velumani@rjsplinux ~]$ cat > xyz
wp
```

```
mc
gc
^Z
[1]+  Stopped                 cat > xyz
[velumani@rjsplinux ~]$ cat abc
cat: abc: No such file or directory
[velumani@rjsplinux ~]$ cp xyz abc
[velumani@rjsplinux ~]$ cat abc
wp
mc
gc
```

CP OPTIONS

a)Interactive coping(-i):

COMMAND: This option warns the user before overwriting the text file.
SYNTAX:cp –i filename1 filename2
EX:cp –i abc xyz

OUTPUT:

```
[velumani@rjsplinux ~]$ cp -i fourth fifth
cp: overwrite `fifth'? n
```

b)Coping directory(-r)

COMMAND:cp –r
SYNTAX:cp –r dirname
PURPOSE: This option copies an entire directory string that is it copies all sub-directories (or) files.

2)more(paging output):

Command:more
Purpose: This command is used to display a file or program output one
File at a time.
Syntax1:more filename
Syntax2:more filename1 filename2 filename3

Output:

```
[sudha@rjsplinux ~]$ more xyz
a
```

```
b
c
d
[sudha@rjsplinux ~]$
```

3)MV(move command)

Command:mv

Syntax:mv oldfilename newfilename.

Purpose: This command is used to rename the file.

OUTPUT:

```
[sudha@rjsplinux ~]$ cat > CSE
OS
DBMS
C++
SE
^Z
[1]+ stopped     cat > CSE
sudha@rjsplinux ~]$ cat sudha
welcome
gowri
ganapathi
sudha
[sudha@rjsplinux ~]$ mv sudha CSE
[sudha@rjsplinux ~]$ cat CSE
welcome
gowri
ganapathi
sudha
[sudha@rjsplinux ~]$
```

4)RM(remove/delete)

Command:rm

Syntax:rm filename

Purpose: Files can be deleted or removed by using rm command.

OUTPUT:

```
[velumani@rjsplinux ~]$ cat fourth
OS
DS
```

```
PE
[velumani@rjsplinux ~]$ rm fourth
[velumani@rjsplinux ~]$ cat fourth
cat: fourth: No such file or directory
```

5)Cat

Command:cat

Syntax:cat > filename

Purpose: This command is to create, display, concatenate, append information to files

OUTPUT:

```
[sudha@rjsplinux ~]$ cat > fourth
OS
DS
^Z
[1]+ stopped  cat > fourth
[sudha@rjsplinux ~]$ ls
372CS14041.sh    dad       fourth    parrot    sudu1.sh      sun
displaymenu.sh   manju       pathname.sh       sudha.g    3star       dot
logname.sh    pgm1.sh    sun.                   pineapple    end        mango
pgm7.sh    sun11      gowriganapathilinuxfile
```

To display the contents of the file

Command:cat

Syntax:cat filename

Purpose: This command is to display the contents of an existing file

EX: cat fourth

OUTPUT:

```
[sudha@rjsplinux ~]$ cat fourth
OS
DS
```

To copy the contents of second file

Command:cat

Syntax:cat filename1 filename2

Purpose: This command is to store the contents of second file in first file

EX: cat fourth fifth

OUTPUT:

```
[velumani@rjsplinux ~]$ cat > fourth
```

```
OS
DS
^Z
[2]+  Stopped                 cat > fourth
[velumani@rjsplinux ~]$ cat > fifth
Webprogramm
mobilecomputing
^Z
[3]+  Stopped                 cat > fifth
[velumani@rjsplinux ~]$ cat fourth fifth
OS
DS
Webprogram
Mobilecomputing
```

To append data to an existing file

Command:cat

Syntax:cat >> filename

Purpose: This command is used to append data to an existing file.

EX: cat >> fourth

OUTPUT:

```
[velumani@rjsplinux ~]$ cat >> fourth
PE
^Z
[4]+  Stopped                 cat >> fourth
[velumani@rjsplinux ~]$ cat fourth
OS
DS
PE
```

Cat options

a) Command: \v

Purpose: Displaying non printable characters. If we have non printing ASCII character in our input this option is used.

b) Command:- n

Purpose: This option numbers line. Each line of the file will be numbered

Syntax: cat -n filename

EX:cat -n fourth

OUTPUT:

```
[velumani@rjsplinux ~]$ cat -n fourth
     1  OS
     2  DS
     3  PE
```

Other commands

1)tput clear

Command:tput clear
Syntax:tput clear
Purpose: This command clears the screen.
EX: tput clear

2)Who

Command:who
Syntax:who
Purpose: This command maintains an account of all users who are logged on to the system.It displays the information listing of the users.

OUTPUT:

```
[velumani@rjsplinux ~]$ who
shashank pts/0        2002-01-01 05:35 (192.168.1.3)
syed     pts/2        2002-01-01 05:36 (192.168.1.26)
velumani pts/1        2002-01-01 05:39 (192.168.1.1)
smitha   pts/3        2002-01-01 05:40 (192.168.1.6)
vikramkumar pts/4        2002-01-01 05:42 (192.168.1.56)
shwetha  pts/6        2002-01-01 05:47 (192.168.1.5)
velumani pts/5        2002-01-01 05:52 (192.168.1.1)
```

4)cal

Command:cal
Syntax:cal
Purpose: This command displays the calendar of any specific month of a complete year.
OUTPUT:

```
[sudha@rjsplinux ~]$ cal
    February 2015
Su Mo Tu We Th Fr Sa
```

```
 1   2    3  4  5  6  7
 8   9   10 11 12 13 14
15  16   17 18 19 20 21
22  23   24 25 26 27 28

[sudha@rjsplinux ~]$
```

5)Date

Command:date
Syntax:date
Purpose: This command is used to display current date and time of the system.

OUTPUT:

```
[sudha@rjsplinux ~]$ date
Thr mar 24 7:14:55 IST 2016
[sudha@rjsplinux ~]$
```

Date Options:

```
[velumani@rjsplinux ~]$ date +%D
01/01/02
[velumani@rjsplinux ~]$ date +%H
06
[velumani@rjsplinux ~]$ date +%M
42
[velumani@rjsplinux ~]$ date +%S
59
[velumani@rjsplinux ~]$ date +%T
06:43:06
[velumani@rjsplinux ~]$ date +%w
2
[velumani@rjsplinux ~]$ date +%a
Tue
[velumani@rjsplinux ~]$ date +%h
Jan
[velumani@rjsplinux ~]$ date +%r
06:43:35 AM
```

```
[velumani@rjsplinux ~]$ date +%y
02
```

6)BC(Binary calculator)

Command:bc

Syntax:bc

Purpose: This command is used in converting number system.

OUTPUT:

```
        [sudha@rjsplinux ~]$ bc
        Bc 1.06.95
        Copyright 1991-1994,1997,1998,2000,2004,2006 free software
foundation,inc.
        This is free software with ABSOLUTELY NO WARRANTY.
        For details type ‘warranty’.
        10+20
        30
        5*5;8/2
        25
        4
Integer computation
8/3
O/P:2
Floating point computation
bc
scale=2                     scale=3
8/3                         8/3
Result=2.66                 Result=2.666
To convert binary to decimal
EX:ibase=2
        1101
        13
Decimal to binary
EX:obase=2
        13
        1101
Hexadecimal to decimal      Decimal to Hexadecimal
```

```
EX:obase=16                    obase=16
    15                         10
    F                          A
```

7)MAN

Command:man
Syntax:man command_name
Purpose: This command is used for getting information for all commands.
EX:man ls

OUTPUT:

```
[sudha@rjsplinux ~]$ man ls
LS(1)        User commands           LS(1)
NAME
 Ls – list directory contents
SYNOPSIS
Ls[OPTION]...[FILE]...
```

DESCRIPTION

```
        List information about the FILE's(the current directory by default).
Sort entries alphabetically if none of –cftuvSUX nor –sort.

Mandatory arguments to long options are mandatory for short options too.
-a,--all
        Do not ignore entries starting with.
-A,--almost-all
        Do not list implied.and..
--author
        With –l,print the author of each file.
-b,--escape
        Print C-style escapes for non-graphic characters.
--block-size=SIZE
        Use SIZE-byte blocks.see SIZE format below.
-B,--ignore-backups
        Do not list implied entried ending with ~
-c with –It:sortby,andshow,ctime(time of last modification of file status
information)
   with – l:show ctime and sort by name otherwise:sort by ctime.
```

8)Passwd

Command: passwd
Syntax: passwd {options} {user_name}
Purpose: To change the current password.
EX: passwd velumani

OUTPUT:

```
[velumani@rjsplinux ~]$ passwd velumani
passwd: Only root can specify a user name.
```

9)Uname

Command: uname
Syntax: uname [options]
Purpose: Print information about the current system.
Print certain system information. If no *OPTION* is specified ,**uname** assumes the **-s** option.
EX: uname -s

OUTPUT:

```
[velumani@rjsplinux ~]$ uname -s
Linux
[velumani@rjsplinux ~]$ uname -n
rjsplinux
[velumani@rjsplinux ~]$ uname -r
2.6.35.6-45.fc14.i686
[velumani@rjsplinux ~]$ uname -m
i686
[velumani@rjsplinux ~]$ uname -p
i686
[velumani@rjsplinux ~]$ uname -i
i386
[velumani@rjsplinux ~]$ uname -o
GNU/Linux
[velumani@rjsplinux ~]$ uname --version
uname (GNU coreutils) 8.5
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
```

Working with files & directories

Directory related commands

1)Pwd(Print working directory)

Command:pwd

Purpose: This command displays the full path name of current directory that we are workingin.It has no options.

Syntax:pwd

OUTPUT:

```
[velumani@rjsplinux ~]$ pwd
/home/Velumani
```

2)Cd(Change directory)

Command:cd

Purpose:we can move around the file system by using the “cd” command.When “cd” is used with an argument,it changes the current directory to the directory specified as the argument.

OUTPUT:

```
[sudha@rjsplinux ~]$ cd first
[sudha@rjsplinux first]$
```

3)mkdir(make directory)

Command:mkdir

Purpose: This command is used to create directory (or) sub-directory’s.

Syntax:mkdir directory_name

OUTPUT:

```
[sudha@rjsplinux first]$ mkdir velumani
[sudha@rjsplinux first]$ ls
Velumani ceee  icc m1 science co
 [sudha@rjsplinux first]$
```

4)rmdir(Remove directory)

Command:rmdir

Purpose: This command is used to remove directories.

Syntax:rmdir directory_name

OUTPUT:

```
[sudha@rjsplinux first]$ rm icc
[sudha@rjsplinux first]$ ls
Velumani ceee m1 science co
```

<u>File related commands</u>

1)Comm

Command:comm.

Purpose: This command compares two sorted files line by line and displays the instance that are common when this command is executed it executed it displays 3 columnar output.

Syntax: comm filename1 filename2

Output:

```
[velumani@rjsplinux ~]$ cat > color1
blue
pink
red
white
^Z
[7]+  Stopped                cat > color1
[velumani@rjsplinux ~]$ cat > color2
black
green
orange
red
^Z
[8]+  Stopped                cat > color2
[velumani@rjsplinux ~]$ comm color1 color2
        black
blue
        green
        orange
pink
                red
white
```

Note:1st column displays unique lines in the first file (color1: blue,pink,white)
2nd column displays unique lines in the second file (color2: black,green,orange)
3rd column displays common to both the files.

2)Comparing two files

COMMAND:cmp

PURPOSE: This command is used to compare two files to know whether they are identical (or) not.

SYNTAX:cmp filename1 filename2

OUTPUT:

```
[velumani@rjsplinux ~]$ cmp color1 color2
color1 color2 differ: byte 3, line 1
```

3)Diff

Command:diff

Purpose: This command compares two text file and bring the lines that are different.It uses an index that all the line that differs in two files.

Syntax:diff filename1 filename2

Output:

```
[velumani@rjsplinux ~]$ cat f5
I like sunday
I like to use mobile
I like to continue my studies
[velumani@rjsplinux ~]$ cat f6
I like sunday
I like to use Tablet
I like to continue my education
I like to continue my studies
[velumani@rjsplinux ~]$ diff f5 f6
2c2,3
< I like to use mobile
---
> I like to use Tablet
> I like to continue my education
```

4)tar

Command:tar

Purpose: The archival program this command is used for creating a disk achieve which contains a group of files or an entire directory structure.

Following list shows the key option used for tar command.

-c:-creates an archive.
-x:-extract files from archive.
-t:-display files in archive.
Syntax:tar –c filename

Output:

```
[sudha@rjsplinux ~]$ tar –c os
Os00006640001057000105700000000031074256445040106340ustar One
two
Three four five
```

5)Umask (User mask)

COMMAND: umask

PURPOSE: The user file-creation mode mask (umask) is use to determine the file permission for newly created files. It can be used to control the default file permission for new files.

SYNTAX:umask [-p] [-s] [mod]

OUTPUT:

```
[sudha @rjsplinux first]$ umask –p 777
[sudha @rjsplinux first]$ ls –l
Total 16
drwxrwxr-x 2 sudha sudha 4096 Mar 03 7:30 abhi
drwxrwxr-x 2 sudha sudha 4096 Mar 03 7:31 ceee
drwxrwxr-x 2 sudha sudha 4096 Mar 04 M1
drwxrwxrwx 2 sudha sudha 4096 Mar 04 sci
```

6)WC(Word count)

COMMAND:wc

PURPOSE: This command counts number of statements (or) lines,words and character which is written in the file.

SYNTAX:wc filename

EX:wc velumani

OUTPUT:

```
[velumani@rjsplinux ~]$ cat > velumani
I like to play cricket
Raina is my favourite player
India won the 2011 world cup
^Z
[11]+  Stopped                 cat > velumani
[velumani@rjsplinux ~]$ wc velumani
 3 16 81 velumani
```

Manipulating file permission using chmod command

1)chmod(Change mode)

COMMAND:chmod

PURPOSE: This command is used to set the permission of one (or) more files for all 3 categories of users that is user group and other.It can be run only by the users and super user.

Relative permission

SYNTAX:chmod categoryoperation permission filename

OUTPUT:

```
[velumani@rjsplinux ~]$ cat abc
wp
mc
gc
[velumani@rjsplinux ~]$ ls -l abc
-rw-r--r-- 1 velumani Velumani 9 Jan  1 06:11 abc
[velumani@rjsplinux ~]$ chmod ugo+x abc
[velumani@rjsplinux ~]$ ls -l abc
-rwxr-xr-x 1 velumani Velumani 9 Jan  1 06:11 abc
[velumani@rjsplinux ~]$ chmod u-x abc
[velumani@rjsplinux ~]$ ls -l abc
-rw-r-xr-x 1 velumani Velumani 9 Jan  1 06:11 abc
```

Absolute permission

SYNTAX:chmod expression filename

OUTPUT:

```
[velumani@rjsplinux ~]$ cat > sci
hello
^Z
[12]+  Stopped                 cat > sci
[velumani@rjsplinux ~]$ ls -l sci
-rw-r--r-- 1 velumani Velumani 6 Jan  1 07:35 sci
[velumani@rjsplinux ~]$ chmod 777 sci
[velumani@rjsplinux ~]$ ls -l sci
-rwxrwxrwx 1 velumani Velumani 6 Jan  1 07:35 sci
```

2) Chown (change ownership)

Command: chown

Purpose: To change the ownership of a file.

Syntax: chown username filename

OUTPUT:

```
[velumani@rjsplinux ~]$ su
Password:
[root@rjsplinux Velumani]# chown srikanth f1
[root@rjsplinux Velumani]# ls -l f1
-rw-r--r-- 1 srikanth Velumani 22 Jan  1  2002 f1
```

Manipulating Hardlink and Softlink using ln command

Hardlink

Ln (Link)

COMMAND: ln

PURPOSE: This command is used to have multiple names for a file.

SYNTAX: ln filename targetlink

OUTPUT:

```
[velumani@rjsplinux ~]$ echo 'This is an example for hardlink' > hl
[velumani@rjsplinux ~]$ ln hl hl1
[velumani@rjsplinux ~]$ ls -li hl1 hl
262461 -rw-r--r-- 2 velumani Velumani 32 Jan  1 07:05 hl
262461 -rw-r--r-- 2 velumani Velumani 32 Jan  1 07:05 hl1
[velumani@rjsplinux ~]$ cat hl
This is an example for hardlink
[velumani@rjsplinux ~]$ cat hl1
This is an example for hardlink
```

Softlink/Symbolic link/system link

Ln (Link)

COMMAND: ln

PURPOSE: This command is used as a special file that contains a reference to a another file.

SYNTAX: ln filename linkname

OUTPUT:

```
[velumani@rjsplinux ~]$ ln -s file1 link1
[velumani@rjsplinux ~]$ ls -l file1 link1
-rw-r--r-- 1 velumani Velumani 23 Jan  1 06:04 file1
lrwxrwxrwx 1 velumani Velumani  5 Jan  1 07:19 link1 -> file
[velumani@rjsplinux ~]$ cat file1
apple
egg
fish
grapes
[velumani@rjsplinux ~]$ cat link1
apple
egg
fish
grapes
```

Learn to use vi-editor

vi insert mode

Once you issue a vi *insert*, *append*, or *open* command, you will be in *vi insert mode*. If you're working with a modern vi or vim implementation, your vi editor is typically configured to show the current mode of operation, so when you go into *insert mode*, you'll see a text string like this on the last line of your vi editor window:
-- INSERT --
At this point you can (a) type text into your file and (b) use the arrow keys to navigate around your file just as you would do with any other text editor. (There may be some complications with older Unix systems, like HP-UX systems, but this statement is generally true.)
A very important concept to know is that when you're in *vi insert mode*, but you want to switch back to *vi command mode*, you easily move back to command mode by pressing the [Esc] key. This command is so important, I'll show it again:
[Esc]
This command is very common, and I often see expert vi users press the [Esc] key several times in a row. They usually do this

(a) to be sure they hit the key and they're really back in command mode, and
(b) to hear the beep from the computer, which happens when you press the [Esc] key when you're already in vi command mode.
This seems to serve as a form of feedback which assures them that they're in command mode.

Command Mode: When vi starts up, it is in Command Mode. This mode is where vi interprets any characters we type as commands and thus does not display them in the xterm window. This mode allows us to move through a file, and to delete, copy, or paste a piece of text. To enter into Command Mode from any other mode, it suffices to press the [Esc] key. If we press [Esc] when we are already in Command Mode, then vi will beep or flash the screen.

Input Mode: In Input Mode, vi accepts keystrokes as text and displays the text as it is entered from the keyboard. vi must be in Input Mode before we can insert text into a file. To enter into Input Mode, we need to put vi into Command Mode and type the key [i].

Line Mode: Line Mode is invoked by typing a colon [:] or a slash [/] while vi is in Command Mode. The cursor will jump to the last line of the screen and vi will wait for a command.

Invoking vi

Command	Function
vi filename	Edit filename starting at line 1
vi +n filename	Edit filename starting at line n
vi + filename	Edit filename starting at the last line

Entering Text

Command	Function
cmd[a]	To insert text just after the cursor
cmd[I]	To add text to the beginning of the current line
cmd[A]	To add text to the end of the current line
cmd[O]	To insert a line just above the current line
cmd[o]	To insert a line just below the current line

Deleting and Cutting Blocks of Text

Command	Function
cmd[x]	To delete a character at the cursor
cmd[X]	To delete a character preceding the cursor
cmd[d][w]	To delete or cut a word at the cursor
cmd[d][b]	To delete or cut a word preceding the cursor
cmd[d][$] or cmd[D]	To cut from the current character to the end of the line
cmd[d][^]	To cut from the current character to the beginning of the line
cmd[d][)]	To delete or cut a sentence at the cursor
cmd[d][(]	To delete or cut a sentence preceding the cursor
cmd[d][}]	To delete or cut a paragraph at the cursor
cmd[d][{]	To delete or cut a paragraph preceding the cursor
cmd number[D] or cmd number [dd]	To cut several lines, where number is the number of lines that you want to yank

Copying and Pasting Text

Preceding, left of the cursor	At, right of the cursor	Function
cmd[y][w]	cmd[y][b]	Yank word
cmd[y][$] or cmd[Y]	cmd[y][0]	Yank whole line
cmd[p]	cmd[P]	Put contents into file

Saving the File and Quitting vi

Command	Function
cmd[:][w]	To just save the file
cmd[:][q]	To quit the file after you have saved it
cmd[:][w][q]	To save and quit vi
cmd[:][w] newfile	To retain the original version of the file and save the changes to another file called newfile
cmd [:][q][!]	To quit vi without saving changes

Simple filters-head, tail, cut, paste, sort, uniq, tr, pr

1)Head

COMMAND:Head

PURPOSE: Outputs the first 10 lines of specified file.This command displays the top of the file(10 lines from first)when used without an option.

SYNTAX:head filename

OUTPUT:

```
[velumani@rjsplinux ~]$ head alpha
a
b
c
d
e
f
g
h
i
j
[velumani@rjsplinux ~]$ head -3 alpha
a
b
c
```

2)Tail

COMMAND: Tail

PURPOSE: Outputs the last 10 lines of the file.This command display last 10 lines when used without option.

SYNTAX:tail filename

OUTPUT:

```
[velumani@rjsplinux ~]$ tail alpha
q
r
s
t
u
v
w
x
y
```

```
z
[velumani@rjsplinux ~]$ tail -3 alpha
x
y
z
```

3)Cut(Spliting a file vertically)

COMMAND: Cut

PURPOSE: We can extract both columns and fields from the file columns are separated in the –c option.

SYNTAX:cut [option] filename

OUTPUT:

```
[velumani@rjsplinux ~]$ cat > emp
Empid|Empname|Des|Salary
11|arya|Manager|25000
12|bharath|CEO|55000
13|ramakrishna|vicepresident|60000
14|dany|engineer|30000
[8]+  Stopped                 cat > emp
[velumani@rjsplinux ~]$ cut -d '|' -f2 emp
Empname
arya
bharath
ramakrishna
dany
[velumani@rjsplinux ~]$ cut -d '|' -f3 emp
Des
Manager
CEO
vicepresident
engineer
[velumani@rjsplinux ~]$ cut -d '|' -f2-4 emp
Empname|Des|Salary
arya|Manager|25000
bharath|CEO|55000
ramakrishna|vicepresident|60000
dany|engineer|30000
[velumani@rjsplinux ~]$ cut -c 1-10 emp
Empid|Empn
11|arya|Ma
```

```
12|bharath
13|ramakri
14|dany|en
[velumani@rjsplinux ~]$ cut -c 1-5 emp
Empid
11|ar
12|bh
13|ra
14|da
```

4)Paste

COMMAND: Paste

PURPOSE: What we cut with cut command can be pasted back with the paste command vertically rather than horizontally.

SYNTAX: paste –d cutlist1 cutlist2

OUTPUT:

```
[velumani@rjsplinux ~]$ cut -d '|' -f2 emp > list1
[velumani@rjsplinux ~]$ cut -d '|' -f3 emp > list2
[velumani@rjsplinux ~]$ cat list1
Empname
arya
bharath
ramakrishna
dany
[velumani@rjsplinux ~]$ cat list2
Des
Manager
CEO
vicepresident
engineer
[velumani@rjsplinux ~]$ paste list1 list2
Empname Des
arya    Manager
bharath CEO
ramakrishna     vicepresident
dany    engineer
```

5)Sort: Ordering a file

COMMAND: Sort

PURPOSE: Arranging the contents of a file in order. It identifies the fields and it can sort a specified field.

SYNTAX:sort filename

OUTPUT:

```
[velumani@rjsplinux ~]$ cat color1
blue
pink
red
white
[velumani@rjsplinux ~]$ sort color1
blue
pink
red
white
```

Sort Options: (-r) Reverse order

COMMAND: Sort

PURPOSE: Arranging the contents of a file in reverse order.

SYNTAX:sort [options] filename

OUTPUT:

```
[velumani@rjsplinux ~]$ cat color1
blue
pink
red
white
[velumani@rjsplinux ~]$ sort -r color1
white
red
pink
blue
```

Sorting on keys(-k)

COMMAND: Sort

PURPOSE: Arranging the contents of a specific column in an order.

SYNTAX:sort [options] filename

OUTPUT:

```
[velumani@rjsplinux ~]$ cat > rjsp1
CS    11    DE
CP    12    BS
ME     06    SOM
EC    01    CEEE
^Z
[5]+  Stopped              cat > rjsp1
[velumani@rjsplinux ~]$ sort -k2 rjsp1
EC    01    CEEE
ME     06    SOM
CS    11    DE
CP    12    BS
[velumani@rjsplinux ~]$ sort -k3 rjsp1
CP    12    BS
EC    01    CEEE
CS    11    DE
ME     06    SOM
```

6)Uniq

COMMAND: Uniq

PURPOSE: When we concatenate a merge files that duplicate the entries. The uniq compare adjacent lines in the sorted files and when used in different options displays the single occurrence.

SYNTAX: uniq [options] filename

OPTIONS	**DESCRIPTION**
-C	Count occurrence of each line.
-d	Prints only duplicate lines.
-D	Prints all duplicate lines.
-i	Ignore case when computing.
-u	Prints only unique lines.

OUTPUT:

```
[velumani@rjsplinux ~]$ cat computer
computer is an electronic device.
computer is an electronic device.
COMPUTER IS AN ELECTRONIC DEVICE.
computer is easy to use.
computer is a dumb machine.
computer is a dumb machine.

[velumani@rjsplinux ~]$ uniq -c computer
2 computer is an electronic device.
1 COMPUTER IS AN ELECTRONIC DEVICE.
1 computer is easy to use.
2 computer is a dumb machine.

[velumani@rjsplinux ~]$ uniq -u computer
COMPUTER IS AN ELECTRONIC DEVICE.
computer is easy to use.

[velumani@rjsplinux ~]$ uniq -d computer
computer is an electronic device.
computer is a dumb machine.

[velumani@rjsplinux ~]$ uniq -D computer
computer is an electronic device.
computer is an electronic device.
computer is a dumb machine.
computer is a dumb machine.

[velumani@rjsplinux ~]$ uniq -i computer
computer is an electronic device.
computer is easy to use.
computer is a dumb machine.
```

7)Tr(Translating character)

COMMAND: Tr

PURPOSE: This command is used to translate a character. Tr command take input from standard input.It does not take filename an arguments.

SYNTAX:tr option expression1 expression2 symbols

OUTPUT:

```
[sudha@rjsplinux ~]$ cat > symbols
| pipe
*astric
^ caret
^Z
```

```
[1]+ stopped          cat > symbols
[sudha@rjsplinux ~]$ tr '*' '~' < symbols
| pipe
~ astric
^ caret
[sudha@rjsplinux ~]$
```

8)Pr

COMMAND: Pr

PURPOSE: This command format inputs to print the pr command prepares file for printing by adding header,footers & formatted text.

SYNTAX:pr filename

OUTPUT:

```
[sudha@rjsplinux ~]$ pr xyz
2002-01-09  6:43          xyz          page1
A
B
C
D
E
F
G
H
I
J
```

Pr options

Character	**Description**
1)-t	To suppress the header and footer.
2)-dp	Double spaces input.
3)-n	Number lines.
4)-h	Header of our option.
5)-on	Offset lines by n spaces and increases left Margin of page sets the page length.
6)-l	sets the page length. **EX:**pr –l 54 velu

Expressions and search patterns. (dot operator),*,^,+,?,grep,egrep,fgrep

1)GREP (Global Regular Expression Print)

COMMAND:grep

PURPOSE:grep scans its input for its pattern and displays the selected patterns, the line number or the filename where the pattern occurs.

SYNTAX:grep option pattern filename.

OUTPUT:

```
        [velumani@rjsplinux ~]$ cat > students
        1       cs      arun
        2       cs      bhavya
        3       cp      varun
        4       me      ajith
        5       CS      vikram
        ^Z
[1]+  Stopped                 cat > students
[velumani@rjsplinux ~]$ grep 'cs' students
1       cs      arun
2       cs      bhavya
```

GREP OPTIONS

a) ignoring case(-i)

COMMAND:grep

PURPOSE:grep option –i(ignore),which ignores case for the pattern matching.

SYNTAX:grep option pattern filename.

OUTPUT:

```
[velumani@rjsplinux ~]$ grep -i 'cs' students
1       cs      arun
2       cs      bhavya
5       CS      vikram
```

b) deleting the line(-n)

COMMAND:grep

PURPOSE:this option is used to display the line number containing the pattern along with the lines.

SYNTAX:grep option pattern filename.

OUTPUT:

```
[velumani@rjsplinux ~]$ grep -n 'me' students
4:4     me      ajith
```

c) delecting lines(-v)

COMMAND:grep

PURPOSE:this option selects all the lines except lies containing the pattern

SYNTAX:grep option pattern filename.

OUTPUT:

```
[velumani@rjsplinux ~]$ grep -v 'me' students
1	cs	arun
2	cs	bhavya
3	cp	varun
5	CS	vikram
```

d) displaying filename(-l)

COMMAND:grep

PURPOSE:This option displays only the names of files containing the pattern.

SYNTAX:grep option pattern filename.

OUTPUT:

```
[velumani@rjsplinux ~]$ grep -l 'cs' students
Students
```

e) counting lines containing patterns (-c)

COMMAND:grep

PURPOSE:This option counts the number of lines containing the patterns.

SYNTAX:grep option pattern filename.

OUTPUT:

```
[velumani@rjsplinux ~]$ grep -c 'me' students
1
```

2)EGREP(Extended Global Regular Expressions Print)

COMMAND:egrep

PURPOSE:This command extends grep pattern matching capability.it offers all the option of grep but it moves useful features.Its the facilts to specify more than one pattern for search.each pattern is separated from the other by a pipe(|)symbol.

SYNTAX:egrep ‘pattern’ | ‘pattern2’ filename..

OUTPUT:

```
[velumani@rjsplinux ~]$ egrep 'c|s' students
1	cs	arun
2	cs	bhavya
3	cp	varun
```

OPTIONS:

a) **To match one or more occurrence (+)**
COMMAND:egrep
PURPOSE:This character matches one or more occurrence of the previous character.
SYNTAX: command pattern character filename
OUTPUT:

```
[velumani@rjsplinux ~]$ egrep cs+ students
1       cs      arun
2       cs      bhavya
```

b) **To match zero or one occurrence(?)**
COMMAND:egrep
PURPOSE:This character matches zero or one occurrence of the previous character.
SYNTAX: command pattern character filename
OUTPUT:

```
[velumani@rjsplinux ~]$ egrep z? students
1       cs      arun
2       cs      bhavya
3       cp      varun
4       me      ajith
5       CS      vikram
```

2)FGREP(Fixed GREP)

COMMAND:fgrep
PURPOSE:This command is used to extract only fixed string without the use of any regular expression.
SYNTAX:fgrep 'fixed string' filename..
OUTPUT:

```
[velumani@rjsplinux ~]$ cat health
Apple is a good fruit.
Apple is grown in kashmir
I like mangoes
An Apple a day keeps the doctor away
[velumani@rjsplinux ~]$ fgrep "Apple" health
Apple is a good fruit.
Apple is grown in kashmir
An Apple a day keeps the doctor away
```

Following list describes various character for pattern matching

CHARACTER	DESCRIPTION
1)(Asterisk)	Its refers to immediately preceeding character. **SYNTAX:**Grep g* filename
2) .(Dot)	A dot matched a single character. **SYNTAX:**Grep 1.filename **EX:**Grep 1. students
3) ^(Caret)	A ^(Caret symbol is used for matching at the beginning of the line. **SYNTAX:**Grep ^2 filename
4) $(Dollar)	A $(Dollar) symbol is used for matching at the end of the line. **SYNTAX:**Grep $ filename.

Example for 1):

```
[velumani@rjsplinux ~]$ grep j* dot
id      name    age
1       manju   21
2       venu    20
3       jaga    19
4       hari    19
5       siva    18
```

Example for 2):

```
[velumani@rjsplinux ~]$ grep 1. students
1       cs      arun
```

Example for 3):

```
[velumani@rjsplinux ~]$ grep ^2 dot
2       venu    20
```

Example for 4):

```
[velumani@rjsplinux ~]$ grep 9$ dot
3       jaga    19
4       hari    19
```

Process management commands

Process status(ps)

COMMAND:ps

PURPOSE: This command displays some process attributes and the processes associated with the user at the terminal.

SYNTAX:ps

OUTPUT:

```
[velumani@rjsplinux ~]$ ps
  PID TTY          TIME CMD
 2096 pts/6    00:00:00 bash
 2442 pts/6    00:00:00 ps
```

Identifying Process

COMMAND: ls

PURPOSE: This command is used to identify all currently running process.

SYNTAX: ls /proc

OUTPUT:

```
[velumani@rjsplinux ~]$ ls /proc
1     1308  1739  1850  29   49   992          kallsyms      self
10    131   1745  1852  3    5    acpi         kcore         slabinfo
1005  132   1762  1854  30   50   asound       keys          softirqs
1018  14    1768  1892  31   51   buddyinfo    key-users     stat
1019  1428  1770  1893  315  528  bus          kmsg          swaps
1026  1429  1773  19    32   529  cgroups      kpagecount    sys
1034  1430  1775  1908  322  6    cmdline      kpageflags    sysrq-trigger
1042  1431  1777  1910  33   7    cpuinfo      latency_stats  sysvipc
1069  1434  1789  1911  331  712  crypto       loadavg       timer_list
1079  1476  1792  1934  332  761  devices      locks         timer_stats
1080  15    18    1938  333  762  diskstats    mdstat        tty
1087  1544  1807  1941  34   763  dma          meminfo       uptime
11    16    1819  2     35   764  dri          misc          version
1100  1658  1820  20    384  765  driver       modules       vmallocinfo
1142  1663  1825  21    394  766  execdomains  mounts        vmstat
1161  1667  1829  22    4    8    fb           mtrr          zoneinfo
1162  1686  1833  23    405  809  filesystems  net
12    1694  1834  24    41   9    fs           pagetypeinfo
1222  17    1836  25    42   914  interrupts   partitions
1257  1707  1839  26    44   947  iomem        sched_debug
13    1718  1841  27    45   966  ioports      schedstat
130   1730  1843  28    48   980  irq          scsi
```

Process options

1)-f(Full listing)

COMMAND:ps

PURPOSE: This option is used to get all detailed listing which also shows the parent of every process.

SYNTAX:ps –f

OUTPUT:

```
[velumani@rjsplinux ~]$ ps -f
UID        PID  PPID  C STIME TTY          TIME CMD
velumani  1911  1910  0 05:36 pts/0    00:00:00 -bash
velumani  1938  1911  0 05:37 pts/0    00:00:00 less
velumani  1942  1911  1 05:44 pts/0    00:00:00 ps -f
```

Where

UID: User ID

PID: Process ID

C:Indicates the amount of CPU time consumed by the process.

STIME: Shows the time of process started.

TTY(Terminal): In which the process is executing.

TIME: Shows the total CPU time used by the process.

CMD: Display the command.

2)-u(User)

COMMAND: ps

PURPOSE: Displays the process of a user.

SYNTAX: ps –u username

OUTPUT:

```
[velumani@rjsplinux ~]$ ps -u velumani
PID TTY          TIME CMD
1911 pts/0    00:00:00 bash
1938 pts/0    00:00:00 less
1946 pts/0    00:00:00 ps
```

3)-a(All users)

COMMAND: ps

PURPOSE: This options list the process of all users but does not displays the system process.

SYNTAX: ps –a

OUTPUT:

```
[velumani@rjsplinux ~]$ ps -a
PID TTY          TIME CMD
2841 pts/5    00:00:00 vim
2974 pts/1    00:00:00 vim
3345 pts/0    00:00:00 vim
 3428 pts/7    00:00:00 bash
 3429 pts/7    00:00:00 pk-command-not-
 3435 pts/7    00:00:00 bash
 3543 pts/13   00:00:00 bash
 3544 pts/13   00:00:00 pk-command-not-
 3545 pts/8    00:00:00 vim
 3650 pts/0    00:00:00 vim
 3846 pts/5    00:00:00 vim
 3848 pts/5    00:00:00 vim
 3849 pts/5    00:00:00 vim
 3934 pts/12   00:00:00 vim
 3937 pts/3    00:00:00 vim
 3938 pts/11   00:00:00 vim
 3965 pts/10   00:00:00 ps
```

4)-e / -A (System processes)

COMMAND:ps

PURPOSE: To display the number of system processes keep running all the times

SYNTAX:ps -e or -A

OUTPUT:

```
[velumani@rjsplinux ~]$ ps -A
  PID TTY          TIME CMD
    1 ?        00:00:01 init
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 ksoftirqd/0
    4 ?        00:00:00 migration/0
    5 ?        00:00:00 watchdog/0
    6 ?        00:00:00 migration/1
    7 ?        00:00:00 ksoftirqd/1
    8 ?        00:00:00 watchdog/1
    9 ?        00:00:00 events/0
   10 ?        00:00:00 events/1
   11 ?        00:00:00 cpuset
```

```
    12 ?        00:00:00 khelper
    13 ?        00:00:00 netns
```

Running process in background

1)&:(No logging out)
COMMAND: &
PURPOSE: This operator is used to run a process in the background.
SYNTAX: command filename operator
OUTPUT:

```
[velumani@rjsplinux ~]$ sort employee &
[5] 1956
[4]   Done                  sort employee
```

2)Nohup(No hang up:Logout out safely)
COMMAND: nohup
PURPOSE: This command when prefix to a command it permits execution of a process even after user has logout.
SYNTAX: nohup command filename operator
OUTPUT:

```
[velumani@rjsplinux ~]$ nohup sort f1 &
[2] 1988
[velumani@rjsplinux ~]$ nohup: ignoring input and appending output to `nohup.out'
[2]-  Done                  nohup sort f1
```

Process termination

Kill: Premature termination of a process.
COMMAND: Kill
PURPOSE: The kill command terminates a process. It uses one (or) more pid's as its arguments.
SYNTAX: kill pid
EX: kill 1766

Changing process priority

Nice: Job execution with low priority
COMMAND: nice
PURPOSE: This command is used to reduce the priority of jobs.
SYNTAX: nice command filename
OUTPUT:

```
[velumani@rjsplinux ~]$ nice wc f5
3 14 65 f5
```

Scheduling process (Usage of sleep and wait commands)

1)AT:ONE-TIME EXECUTION

COMMAND: The AT command takes time as its argument, the job is to be executed and displays the AT > prompt.

The input has to be supplied from the old at 14:08
AT > filename
[Ctrl+D]
SYNTAX:AT time
EX:

```
[velumani@rjsplinux ~]$ at noon
    at> f2
    at> <EOT>
    job 71 at Tue Jan  1 12:00:00 2002
```

2)Batch: Execute in batch queue

COMMAND: Batch
PURPOSE: The batch command also schedules jobs for later execution.
SYNTAX: Batch < filename
OUTPUT:

```
[velumani@rjsplinux ~]$ batch < f5
job 147 at Tue Jan  1 06:20:00 2002
```

Linux system administration

Managing file system

In this file system they are two types they are as follows

1)Mount: Mounting file system

COMMAND: Mount

PURPOSE: This command is used to mount file system.

SYNTAX:mount

OUTPUT:

```
[velumani@rjsplinux ~]$ mount
/dev/mapper/vg_rjsplinux-lv_root on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
/dev/sda1 on /boot type ext4 (rw)
/dev/mapper/vg_rjsplinux-lv_home on /home type ext4 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
```

2)Umount: Unmounting file system

COMMAND:Umount

PURPOSE:Umount file system.we can umount remote resourse by using umount command.

SYNTAX:Umount.

OUTPUT:

```
[velumani@rjsplinux ~]$ umount
Usage: umount -h | -V
      umount -a [-d] [-f] [-r] [-n] [-v] [-t vfstypes] [-O opts]
      umount [-d] [-f] [-r] [-n] [-v] special | node...
```

Disk management utilities

In order to check how much disk space is available in each file system in a computer and to make sure that enough space save our file.

UNIX OFFERS SOME DISK MANAGEMENT UTILITIES

1)Df (Disk Free): Reporting Free Space

COMMAND: df

PURPOSE: This command reports the amount of free space available for each file system.

SYNTAX:df

OUTPUT:

```
[velumani@rjsplinux ~]$ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/vg_rjsplinux-lv_root
                      51606140   5335476  43649224  11% /
tmpfs                   250124        88    250036   1% /dev/shm
/dev/sda1               495844     29024    441220   7% /boot
/dev/mapper/vg_rjsplinux-lv_home
                     187232148    328648 177392588   1% /home
```

2)Du: Disk Usage

COMMAND:du

PURPOSE: This command is used to check the information of disk usage of files and directories on a machine.

SYNTAX:du

OUTPUT:

```
[velumani@rjsplinux ~]$ du
4       ./.gnome2_private
68      ./.gconfd
4       ./.gvfs
4       ./.mozilla/extensions/{ec8030f7-c20a-464f-9b0e-13a3a9e97384}
8       ./.mozilla/extensions
4       ./.mozilla/firefox
4       ./.mozilla/plugins
[2]+  Stopped                 du
```

3)fdisk (Fixed Disk or Format Disk): Creating partitions

COMMAND:fdisk

PURPOSE: With the help of fdisk command you can view, create, resize, delete, change, copy and move partitions on a hard drive using its own user-friendly text-based menu driven interface.

SYNTAX:fdisk.

OUTPUT:

```
[velumani@rjsplinux ~]$ fdisk
Usage:
 fdisk [options] <disk>    change partition table
 fdisk [options] -l <disk> list partition table(s)
 fdisk -s <partition>      give partition size(s) in blocks

Options:
 -b <size>             sector size (512, 1024, 2048 or 4096)
 -c[=<mode>]           compatible mode: 'dos' or 'nondos' (default)
 -h                    print this help text
 -u[=<unit>]           display units: 'cylinders' or 'sectors' (default)
 -v                    print program version
 -C <number>           specify the number of cylinders
 -H <number>           specify the number of heads
 -S <number>           specify the number of sectors per track
```

4)su (Super User)

COMMAND:su

PURPOSE:We can log-on into a unix machine as a user and then issue the ‘su’ command to super user(root).

SYNTAX:su

OUTPUT:

```
[velumani@rjsplinux ~]$ su
Password:
[root@rjsplinux Velumani]#
```

5)Useradd(Adding a user)

COMMAND: useradd

PURPOSE: This command adds a new user to the Linux System Administrator with some specific properties, limitations or comments.

SYNTAX: useradd directory username.

OUTPUT:

```
[velumani@rjsplinux ~]$ useradd
```

To List all usernames: awk -F: '{print $1}' /etc/passwd

6)userdel (Deleting a user)

COMMAND: userdel

PURPOSE: The userdel command deletes a user accountand all associated files. userdel is a low-level utility for removing users.

SYNTAX:userdel directory username

7)groupadd: Adding a group

COMMAND:groupadd

PURPOSE: This command is used to place a user in a new group.An entry for the group has to be created first in /etc/ group.

SYNTAX:groupadd groupname

EX: groupadd CS-2014-batch

8)groupdel: Deleting a group

COMMAND:groupdel

PURPOSE: If a group is no longer required we can delete it.

SYNTAX:groupdel group name

EX:groupdel CS-2014-batch

9)groupmod(modifier)

COMMAND:groupmod

PURPOSE:It is used to modify a groupname.

SYNTAX:groupmod –n new_group_nameold_group_name

Ex:groupmod –n RCB CSK

10)Tar (the archival program)

COMMAND: tar

PURPOSE: This command is used to back-up individual files and directory.we can mount the directory which we want the date to perform back-up tar and wait for a restore the directory.

Advanced filter (Sed & Awk)

1)Sed (The Stream editor)

COMMAND: sed

PURPOSE: This command is used to perform basic text transformations on an input
stream.(A file or input from a pipeline)

SYNTAX: sed [Options] ‘address action’ filename

Where

d: It deletes the specified line.
p: It prints the specified line.(Duplicate line)
q: Quits after the line number specified in the instruction.
$p: Prints the last line of a file.(Duplicate line)

OUTPUT:

```
[velumani@rjsplinux ~]$ cat fruits
kiwi
grapes
apple
mango
orange
[velumani@rjsplinux ~]$ sed '1d' fruits
grapes
apple
mango
orange
[velumani@rjsplinux ~]$ sed '2p' fruits
kiwi
grapes
grapes
apple
mango
orange
[velumani@rjsplinux ~]$ sed '4p' fruits
kiwi
grapes
apple
mango
mango
orange
```

```
[velumani@rjsplinux ~]$ sed '4p' fruits
kiwi
grapes
apple
mango
mango
orange
[velumani@rjsplinux ~]$ sed '1p' fruits
kiwi
kiwi
grapes
apple
mango
orange
[velumani@rjsplinux ~]$ sed '$p' fruits
kiwi
grapes
apple
mango
orange
orange
[velumani@rjsplinux ~]$ sed 'p' fruits
kiwi
kiwi
grapes
grapes
apple
apple
mango
mango
orange
orange
```

2) Awk (Aho, Weinberger, and Kernighan)

COMMAND: awk

PURPOSE: This command is used for processing or analyzing text files which is organized by rows and columns

SYNTAX: awk options 'selection-criteria {actions}' filename

Where

Selection-criteria:It filter input and selects lines for the actions.(form of addressing)

{}:Components is enclosed within curly braces.

OUTPUT:

```
[velumani@rjsplinux ~]$ cat rjsp
        Harish  :     CS
        Jaga    :     CS
        Ram     :     ME
        Siva    :     CS
        Venu    :     CS
        Srini   :     EE
[velumani@rjsplinux ~]$ cat rjsp | awk 'begin {fs="Harish"} -f fs {print $1}'
        Harish
        Jaga
        Ram
        Siva
        Venu
        Srini
[velumani@rjsplinux ~]$ cat rjsp | awk 'begin {fs="Harish"} -f fs {print $1,$3}'
        Harish CS
        Jaga CS
        Ram ME
        Siva CS
        Venu CS
        Srini EE
```

PART- B

SHELL SCRIPTS & C PROGRAMS

1. Write a shell script to display current date, time, username and directory.

```
#!/bin/bash
echo “Hello,$LOGNAME”
echo "Current date is `date`"
echo "Username is `who i am`"
echo "Current directory `pwd`"
```

OUTPUT:

```
[velumani@rjsplinux ~]$ bash pgm1
Current date is Tue Jan  1 05:43:37 IST 2002
Username is velumani pts/2        2002-01-01 05:39 (192.168.1.1)
Current direcotry /home/Velumani
```

2. Write shell script to show various system configuration like:

- **Currently logged user name and his log name**
- **Current shell**
- **Your home directory**

```
#!/bin/bash
echo "user name:$USER"
echo "Logname:$LOGNAME"
echo "Home directory:$HOME"
echo "current shell:$SHELL"
```

OUTPUT:

```
[velumani@rjsplinux ~]$ bash pgm3
user name:velumani
Logname:velumani
Home directory:/home/Velumani
current shell:/bin/bash
```

3.Write shell script to show various system configuration like:

- **Your operating system type**
- **Your current path setting**
- **Your current working directory**
- **Show all available shells**

```
#!/bin/bash
echo "our OS type:$OSTYPE"
echo "Path:$PATH"
echo "Current working directory:$PWD"
```

OUTPUT:

```
[velumani@rjsplinux ~]$ bash pgm4
our OS type:linux-gnu
Path:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/Velumani/b
in
Current working directory:/home/Velumani
```

4.Write a Shell script to accept any two file names and check their file permissions.

```
#!/bin/bash
echo "Enter two filenames"
read file5
read file6
echo "File permissions are:"
ls -l $file5
ls -l $file6
```

OUTPUT:

```
[velumani@rjsplinux ~]$ bash pgm6
Enter two filenames
rjsp
velu
File permissions are:
-rw-rw-r--. 1 student student 62 Feb 23 00:48 rjsp
-rwxrwxr-x. 1 student student 16 Mar  5 22:45 velu
```

5.Write a Shell script to read a file name and change the existing file permissions.

```
#!/bin/bash
echo "Enter the filename:"
read file6
ls -l $file6
chmod ugo+x $file6
echo "After change:"
ls -l $file6
```

OUTPUT:

```
[velumani@rjsplinux ~]$ bash pgm6
Enter the filename:
file6
-rw-rw-r--. 1 velumani Velumani 22 Jan  1 07:15 file6
After change:
-rwxrwxr-x. 1  velumani Velumani 22 Jan  1 07:15 file6
```

6.Write a C-program to fork a child process and execute the given Linux commands.

```
#include<stdio.h>
#include<stdlib.h>
main()
{
system("ls");
system("cal");
system("logname");
system("pwd");
}
```

OUTPUT:

```
[velumani@rjsplinux ~]$ gcc pgm7.c
[velumani@rjsplinux ~]$ ./a.out
1st       details      f2      file7        pgm4      recipe     smitha
2nd     Documents      file11          filen        pgm5      renu      student
abc     Downloads      file2           health  pgm6      rjsp       Templates
amc      example       file33          Music  pgm7.c   rjsp.sym        velu
a.out    example1       file5          pgm1   Pictures  shanth            velu1
Desktop  f1            file6           pgm3   Public    shopping        Videos
```

```
      January 2002
Su       Mo   Tu   We   Th   Fr   Sa
          1   2    3    4     5
 6       7    8    9    10   11   12
13       14   15   16   17   18   19
20       21   22   23   24   25    26
27       28   29   30   31

velumani
/home/Velumani
```

7.Write a C-program to fork a child process, print owner process ID and its parent process ID.

```
#include<stdio.h>
#include<sys/types.h>
main()
{
printf("\n\t PID=%d\n\t Child PID=%d\n\n Parent
PID=%d\n\n\n",getpid(),getppid(),fork());
}
```

OUTPUT:

```
[velumani@rjsplinux ~]$ gcc pgm8.c
[velumani@rjsplinux ~]$ ./a.out

        PID=2237
        Child PID=1898

  Parent PID=2238

PID=2238
        Child PID=2237

  Parent PID=0
```

8.Write a C-program to prompt the user for the name of the environment variable, check its validity and print an appropriate message.

```
#include<stdio.h>
int main(int argc,char **argv,char **env)
{
while(*env)
printf("%s\n",*env++);
return 0;
}
```

OUTPUT:

```
[velumani@rjsplinux ~]$ gcc pgm9.c
[velumani@rjsplinux ~]$ ./a.out
REMOTEHOST=192.168.1.1
HOSTNAME=rjsplinux
SHELL=/bin/bash
TERM=ansi
HISTSIZE=1000
USER=velumani
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;05;37;41:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow
=34;4
2:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.taz=01;31:*.lzh=01;31:
*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;3
1:
*.gz=01;31:*.lz=01;31:*.xz=01;31:*.bz2=01;31:*.tbz=01;31:*.tbz2=01;31:*.bz=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:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.jp
g
=01;35:*.jpeg=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
=0
1;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=
01;35:*
```

```
.m2v=01;35:*.mkv=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35
:*.vob=01
;35:*.qt=01;35:*.nuv=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:*.x
wd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.axv=01;35:*.anx=01;35:*.ogv
=01;35
:*.ogx=01;35:*.aac=01;36:*.au=01;36:*.flac=01;36:*.mid=01;36:*.midi=01;36:*.m
ka=
01;36:*.mp3=01;36:*.mpc=01;36:*.ogg=01;36:*.ra=01;36:*.wav=01;36:*.axa=01;
36:*.o
ga=01;36:*.spx=01;36:*.xspf=01;36:
MAIL=/var/spool/mail/velumani
PATH=/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/Velumani
/bin
PWD=/home/Velumani
LANG=en_US.UTF-8
KDE_IS_PRELINKED=1
KDEDIRS=/usr
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
HISTCONTROL=ignoredups
SHLVL=1
HOME=/home/Velumani
LOGNAME=velumani
LESSOPEN=|/usr/bin/lesspipe.sh %s
G_BROKEN_FILENAMES=1
_=./a.out
```