INDEX

UNIT-3 Filters

3.1 **Filters:**
Simple Filters: Paginating files –pr, Splitting a file vertically–cut, Pasting files-paste, Ordering a file-sort, Locate repeated and non-repeated lines - uniq, Translating characters–tr

3.2 **Filters using regular expression:**
Searching for pattern-grep, stream editior –sed

3.3 **Advanced filters:**
Simple awkfiltering, Comparison operators, Variables, Built invariables, Control flow, looping.

UNIT-4 Compressing ,decompressing and achieving files

4.1 **Compressing , decompressing and achieving files:**
Gzip,gunzip, Tar, Zip and unzip

4.2 **Environment variables:**
Environment variables, Alias, Inline command editing, Miscellaneous features, Initialization script.

4.3 **Communication commands:**
Finger, Talk, Mesg, Mailx, Pine, Write, Wall

Practical

1. Check the output of the following commands: date, ls, who, cal, ps, wc, cat, uname, pwd, mkdir, rmdir, cd, cp, rm, mv, diff, chmod, grep, sed, head, tail,cut, paste, sort, find, man
2. Write a script to find the complete path for any file.
3. Write a shell script to execute following commands
   - ⇨ Sort file abc.txt and save this sorted file inxyz.txt
   - ⇨ Give an example of : To execute commands together without affecting result of Each other.

   How to print
   "thisis a three –line
   1. Textmessage"

- ⇨ Which command display version of the UNIX?
- ⇨ How would you get online help of cat command?

4. Write a shell script to execute following commands
   - ⇨ How would u display the hidden files?
   - ⇨ How delete directory with files?
   - ⇨ How would user can do interactive copying?
   - ⇨ How would user can do interactive deletion offiles?
   - ⇨ Explain two functionality of "mv" command with example?
5. Write a shell script to execute following commands
   - ⇨ Create a file called text and store name,age and address init.
   - ⇨ Display the contents of the file text on the screen.
   - ⇨ Delete the directories mydir and new dir at one-shot.
   - ⇨ Sort a numeric file?
   - ⇨ Change the permissions for the file new text to666.
6. Write shell script that accept filename and displays last modification time if file exists, otherwise display appropriate message.
7. Write a shell script to display the login names that begin with 's'.
8. Write a shell script to remove the zero sized file from the current directory
9. Write a shell script to display the name of all the executable file from the current directory.
10. Write a shell script that will display welcome message according to time
11. Write a shell script to find number of ordinary files and directory files.
12. Write a shell script that takes a filename from the command line and checks whether the file is an ordinary file or not.
    - ⇨ If it is an ordinary file then it should display the contents of the file.
    - ⇨ If it is not an ordinary file then script should display the message: " File does not exist or is not ordinary, cannot display."
13. Write a shell script that takes a filename from the user and checks whether it is a directory file or not.
    - ⇨ If it is a directory, then the script should display the contents of the directory.
    - ⇨ If it is not a directory file then script should display the message: "File is not a directory file"
14. Write a shell script that takes a filename as an argument and checks if the file exists and is executable.

- ⇨ If the file is executable then the shell script should display the message: "File exists"
- ⇨ If the file does not exists and is not executable then the script should display the message: "File does not exist or is not executable."

15. Write a shell script that displays all subdirectories in current working directory.

16. Write a shell script that calculates the number of ordinary and directory files in your current working directory.

17. Write a shell script that accepts 2 filenames and checks if both exists; if both exist then append the content of the second file into the first file.

18. Write a shell script that takes the name of two files as arguments and performs the following:
    i. Displays the message :
    "Displaying the contents of file :(first argument)" and displays the contents page wise.
    ii. Copies the contents of the first argument to second argument.
    iii. Finally displays the message: "File copied successfully."

19. Write a shell script to display the following menu and acts accordingly:
    i. Calendar of the current month and year.
    ii. Display "Good Morning/Good Afternoon/Good Evening" according to the current login time.
    iii. User name, Users home directory.
    iv. Terminal name, Terminal type.
    v. Machine name.
    vi. No. of users who are currently logged in; List of users who are currently logged in.

20. Write a shell script that displays the following menu and acts accordingly
    1. Concatenates two strings
    2. Renames a file.
    3. Deletes a file.
    4. Copy the file to specific location

21. Write a shell script to change the suffix of all your *.txt files to .dat.

22. Write a shell script to accept a directory-name and display its contents. If input is not given then HOME directory's contents should be listed. (Make use of command line argument)

23. Write a shell script to get all files of home directory and rename them if their names start with c.
    Newname = oldname111

24. Write a shell script that takes two filename as arguments. It should check whether the contents of two files are same or not, if they are same then second file should be deleted.

25. Write a shell script that accepts two directory names from the command line and copies all the files of one directory to another. The script should do the following:
    - If the source directory does not exist, flash a error message
    - If destination directory does not exist create it
    - Once both exist copy all the files from source directory to destination directory.

26. Write a shell script that displays the following menu
    - List home directory
    - Date
    - Print working directory
    - Users logged in

Read the proper choice. Execute corresponding command. Check for invalid choice.

27. Write a shell script that displays all hidden files in current directory.

28. Write a shell script that Combine two files in the third file horizontally and vertically.

29. Write a shell script to delete all the spaces from a given file.

30. Write a shell script to find a given date fall on a weekday or a weekend.

31. Write a shell script to search for a given word in all the files given as the Arguments on the command line.

32. Write a shell script that display last modified file in the current directory.

33. Write a script to display the permissions of the particular file.

34. Write a shell script to display the calendar in the following manner:
    i. Display the calendar of months m1 and m2 by 'CAL m1, m2' command file.
    ii. Display the calendar of the months from m1 to m2 by 'CAL m1-m2' command file.

35. Write a shell script to display the following menu for a particular file :
    i. Display all the words of a file in ascending order.
    ii. Display a file in descending order.
    iii. Toggle all the characters in the file.
    iv. Display type of the file.

36. Write a shell script to check whether the named user is currently logged in or not.

37. Write a shell script to display the following menu for a particular file:

i. Display all the words of a file in ascending order.
ii. Display a file in descending order.
iii. Display a file in reverse order.
iv. Toggle all the characters in the file
v. Display type of the file.

38. Write a shell script to find total no. Of users and finds out how many of them are currently logged in.

39. Write a shell script that displays the directory information in the following format
Filename Size Date Protection Owner

40. Write a shell script to display five largest files from the current directory.

41. Write a shell script that toggles contents of the file

42. Write a shell script that report whether your friend 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 have write permission to his terminal or if he hasn't logged in then such a message should be mailed to him about your dinner proposal.

43. Write a shell script for the performing the write and mail.

44. Write a shell script to accept any character using command line and list all the files starting with that character in the current directory.

45. Create a file called student containing roll-no, name and marks.
a. Display the contents of the file sorted by marks in descending order
b. Display the names of students in alphabetical order ignoring the case.
c. Display students according to their rollnos.
d. Sort file according to the second field and save it to file'names'.
e. Display the list of students who scored between 70 and80.ize Date Protection Owner

UNIT-1

General –Purpose Utilities and Files and directories

- **General Purpose Utilities:**

  Cal, date, echo, bc, script, who, uname, tty, man, info, passwd, logout, wc.

- **Linux Files and Directories:**

  Current Working Directory-pwd, Listing Files and Directories-ls, Matching Filenames with Patterns (wildcard characters), Simple ways to create a file- touch, cat,, echo, Showing the contents of a file-cat, more, less, head, tail

Unit –1 General –Purpose Utilities and Files and Directories Introduction

1.1 General Purpose Utilities:

Unix/Linux general purpose utility command list: man, who, cat, cd, cp, ps, ls, mv, rm, mkdir, rmdir, echo, more, date, time, kill, history, finger, pwd, cal, logout, shutdown. From that some of general-purpose utilities command we will discuss: Cal, date, echo, bc, script, who, uname, tty, man, info, passwd, logout, wc.

1. **Calendar**: cal command. Its use to display the calendar.

Syntax:

cal [options] [month] [year]

Examples:

⇨ cal

In general, if no options are given, cal displays the current month at the command line.

⇨ cal 12 2000

Displays the calendar for December of the year 2000.

⇨ cal -3 5 2008

Displays the calendar of April, May and June month of the year 2008.

```
[root@localhost ~]#
[root@localhost ~]# cal
     January 2014
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
[root@localhost ~]# cal 12 2000
      December 2000
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
[root@localhost ~]# cal -3 5 2008
      April 2008                    April 2008                    April 2008
Su Mo Tu We Th Fr Sa          Su Mo Tu We Th Fr Sa          Su Mo Tu We Th Fr Sa
      1  2  3  4  5                      1  2  3             1  2  3  4  5  6  7
 6  7  8  9 10 11 12           4  5  6  7  8  9 10           8  9 10 11 12 13 14
13 14 15 16 17 18 19          11 12 13 14 15 16 17          15 16 17 18 19 20 21
20 21 22 23 24 25 26          18 19 20 21 22 23 24          22 23 24 25 26 27 28
27 28 29 30                   25 26 27 28 29 30 31          29 30
```

- **Calendar other options:**

Options	Meaning
-1	Displays the single month as output.
-3	Displays prev/current/next month output.
-s	Displays Sunday as the first day of the week.
-m	Displays Monday as the first day of the week.
-j	Displays Julian dates (days one-based, numbered from January 1).
-y	Displays a calendar for the current year.

2. Date: You can display the current date with date command, which shows the date and time to the nearest second:

Date: The date command can also be used with suitable format specifier as arguments.

You can print only the month using the format +%m

date +%m

Or month name

date +%h

Combining both the options.

date +"%h %m"

There are many other format specifiers, and useful ones are listed below :

d – The day of the month (1 to 31)

y – The last two digits of the year

H, M, and S – The hour, minute and second, respectively.

T – The time in the format hh:mm:ss

When you use multiple format specifier, you must enclose them within quotes (single or double), and use a single + symbol before it.

```
[root@localhost  ~]#
[root@localhost  ~]#
[root@localhost  ~]#  date
Tue   Jan  28   17:49:36 EST 2014
[root@localhost  ~]#
[root@localhost  ~]#
[root@localhost  ~]# date +%m
01
[root@localhost  ~]#
[root@localjost  ~]#  date +%h
Jan
[root@localhost  ~]#  date +" %m  %h"
01  Jan
[root@localhost  ~]#  date +%d
28
[root@localhost  ~]#  date +%y
14
[root@localhost  ~]#
[root@localhost  ~]#  date +" %H   :   %M    : % S "
 17 : 51 : 39
[root@localhost  ~]#  date +%T
17:51:50
[root@localjost  ~]#
```

3. Echo: display a line of text.

Syntax: echo [option(s)] [string(s)]

A string is any finite sequence of characters (i.e., letters, numerals, symbols and punctuation marks). echo This is a pen. Displaying the value of variable x. echo The number is $x. echo, by default, follows any output with a newline character. This is a non-printing (i.e., invisible) character that represents the end of one line of text and the start of the next

```
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# echo -e "\n Project1: \n\n\tplan \n\twrite\n\ttest\n"> project1
[root@localhost ~]# cat project1

Project1:

        plan
        write
        test

[root@localhost ~]# echo -E "\n Projects: \n\n\tplan \n\tcode \n\ttest\n">
\n Projects: \n\n\tplan \n\tcode \n\ttest\n
[root@localhost ~]#
[root@localhost ~]#
```

4. **bc:** The calculator

bc is a text-based calculator. When you invoke bc without arguments, the cursor keeps on blinking and nothing seems to happen. bc belongs to a family of commands (called filters) that expects input from keyboard when used without an argument. Use [ctrl-d] to quit bc

```
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# bc
bc 1.06.95
Copyright 19991-1994, 1997, 1998, 2000, 2004, 2006 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type 'warranty'.

123*2
246
412/2
206
230+400-12/2
624
(23 + 45) – (43*3) * 9 – (456/2)
-1321
exit
0
quit
[root@localhost ~]#
```

5. script: Recording your session

script command lets you record your login session in a file. You can later view the file. If you are doing some important work and wish to keep a log of all your activities, you should invoke this command immediately after you login:

script

Script started, the file is typescript

exit

Script done; the file is typescript. You can now view this file with the cat command. Script overwrites any previous typescript that may exist. If you want to append to it, or want to use a different log file, you can consider below arguments.

```
[root@localhost ~]#
[root@localhost ~]# script logfile
Script started. File is logfile
[root@localhost ~]# echo "My shell is $SHELL"
My shell is /bin/bash
[root@localhost ~]#
[root@localhost ~]# printf "My shell is $SHELL"
My shell is /bin/bash [root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# pwd
/root
[root@localhost ~]# echo $USER
Root
[root@localhost ~]# exit
exit
Script done, file is logfile
[root@localhost ~]# cat logfile
Script started on Tue 28 Jan 2014 05:59:54 PM EST
[root@localhost ~]# echo "My shell is $SHELL"
My shell is /bin/bash
[root@localhost ~]#
[root@localhost ~]# printf "My shell is $SHELL"
My shell is /bin/bash [root@localhost ~] #
[root@localhost ~]#
[root@localhost ~]# pwd
/root
[root@localhost ~]# echo $USER
Root
[root@localhost ~]# exit
exit
Script started on Tue 28 Jan 2014 06:01:13 PM EST
[root@localhost ~]#
```

6. **who:** Linux maintains an account of all users who are logged on to the system. It's often a good idea to know their user-ids so you can mail them messages.

First column Shows usernames (or user-ids)

Second column The second column shows device names of their respective terminals

The third column, Fourth column & Fifth column,Shows the date and time of logging in.

Sixth column Shows machine name from where the user logged in

who –Hu

-H for header option –u for more detailed list.

```
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# who
root     :0       2014-01-28  14:51 (:0)
root     pts/0    2014-01-28  14:52 (:0)
[root@localhost ~]#
[root@localhost ~]# who –Hu
Name             Line     Time                    IDLE          PID COMMENT
root             :0       2014-01-28  14:51       ?             1408 (:0)
root             pts/0    2014-01-28  14:52       old           1793 (:0)
root@localhost ~]#
[root@localhost ~]# who am i
root     pts/0            2014-01-28  14:52 (:0)
[root@localhost ~]#
[root@localhost ~]#
```

7. **uname:** Knowing your machine's characteristics

The uname command displays certain features of the operating system running on your machine.To simply display the name of operating system

uname

Use –r option to know the version of your operating system

uname -r

If your machine is connected to a network, it must have a name (called hostname).

uname –n

The same output is obtained with hostname command:

```
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# uname
Linux
[root@localhost ~]#
[root@localhost ~]# uname -r
3.6.10-4. fc18. *86_64
[root@localhost ~]#
[root@localhost ~]# uname -n
Localhost.localdomain
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
```

8. **tty:** Know your Terminal

Since Unix/Linux treats even terminals as files, it's reasonable to expect a command that tells you the filename of the terminal you are using. It's the tty (teletype) command. This command is simple and need no arguments.

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


[root@localhost ~]#
```

9. **man:** man command in Linux is used to display the user manual of any command that we can run on the terminal. It provides a detailed view of the command which includes NAME, SYNOPSIS, DESCRIPTION, OPTIONS, EXIT STATUS, RETURN VALUES, ERRORS, FILES, VERSIONS, EXAMPLES, AUTHORS and SEE ALSO.

Every manual is divided into the following sections:

- Executable programs or shell commands

- System calls (functions provided by the kernel)
- Library calls (functions within program libraries
- Games
- Special files (usually found in /dev)
- File formats and conventions eg /etc/passwd
- Miscellaneous (including macro packages and conventions), e.g. groff(7)
- System administration commands (usually only for root)
- Kernel routines [Non standard]

Syntax :

$man [OPTION]... [COMMAND NAME]...

10. info: info command reads documentation in the info format. It will give detailed information for a command when compared with the man page. The pages are made using the texinfo tools because of which it can link with other pages, create menus and easy navigation.

Syntax:

info [OPTION]... [MENU-ITEM...]

Options:

-a, –all: It use all matching manuals.

-k, –apropos=STRING: It look up STRING in all indices of all manuals.

-d, –directory=DIR: It add DIR to INFOPATH.

-f, –file=MANUAL: It specify Info manual to visit.

-h, –help: It display this help and exit.

-n, –node=NODENAME: It specify nodes in first visited Info file.

-o, –output=FILE: It output selected nodes to FILE.

-O, –show-options, –usage: It go to command-line options node.

-v, –variable VAR=VALUE: It assign VALUE to Info variable VAR.

–version: It display version information and exit.

-w, –where, –location: It print physical location of Info file.

Examples:

-a : It use all matching manuals and display them for a particular command.

info -a cvs

11. passwd: passwd command in Linux is used to change the user account passwords. The root user reserves the privilege to change the password for any user on the system, while a normal user can only change the account password for his or her own account.

Syntax:

passwd [options] [username]

12. logout: session-logout.

logout command allows you to programmatically logout from your session. causes the session manager to take the requested action immediately.

Syntax:

session-logout[options]

13. wc: Counting Lines, Words & Characters

wc is a word-counting program that counts lines, word and characters. It takes one or more filename as arguments and displays a four-columnar output.

Consider a regular text file sample with the following contents:

I am the wc command

I count lines, words, and characters

With options I can also make a selective count

```
[root@localhost ~]#
[root@localhost ~]# cat sample
I am the wc command
I count lines, words and characters
With options I can also make a selective count
[root@localhost ~]#
[root@localhost ~]# wc sample
  3   20  103 sample
[root@localhost ~]#
[root@localhost ~]# wc – l sample
3 sample
[root@localhost ~]# wc – w sample
20 sample
[root@localhost ~]# wc – c sample
103 sample
[root@localhost ~]#
```

1.2 Linux Files and Directories:

Current Working Directory: The current working directory is the directory in which the user is currently working in. Each time you interact with your command prompt, you are working within a directory. By default, when you log into your Linux system, your current working directory is set to your home directory. To change the working directory use the cd command. For example, to change the current working directory to /tmp you would type:

cd /tmp

a) pwd: The pwd command stands for print working directory. It is one of the most basic and frequently used commands in Linux. When invoked the command prints the complete path of the current working directory.

pwd is a shell builtin in most modern shells such as bash and zsh. Its behavior is slightly different than the standalone /bin/pwd executable. You can use the type command to display all locations containing pwd:

type -a pwd

output:

pwd is a shell builtin

pwd is /bin/pwd

As you can see from the output below, the shell builtin has priority over the standalone executable and it is used whenever you type pwd. If you want to use the standalone pwd binary type the full path to the file /bin/pwd

b) ls: Listing Files and Directories

ls is one of the basic commands that any Linux user should know. The ls command lists files and directories within the file system, and shows detailed information about them. It is a part of the GNU core utilities package which is installed on all Linux distributions. This article will show you how to use the ls command through practical examples and detailed explanations of the most common ls options.

Syntax:

ls [OPTIONS] [FILES]

When used with no options and arguments, ls displays a list of the names of all files in the current working directory :

$ ls

The files are listed in alphabetical order in as many columns as can fit across your terminal:

Output:

cache db empty games lib local lock log mail opt run spool tmp

c) **Wildcards and Pattern Matching**: Wildcards can be used in two different ways. They can be used to specify a single location or file by using a wildcard to represent a character or characters, or they can be used to reference multiple files with a single command.

These wildcards are interpreted by the shell and the results are returned to the command you run. There are three main wildcards in Linux:

- ⇨ An asterisk (*) – matches one or more occurrences of any character, including no character.
- ⇨ Question mark (?) – represents or matches a single occurrence of any character.
- ⇨ Bracketed characters ([ ]) – matches any occurrence of character enclosed in the square brackets. It is possible to use different types of characters (alphanumeric characters): numbers, letters, other special characters etc.

There are several wildcards that can be used in the bash shell. The most common wildcards are * and - . The wildcard * can represent 0, 1 or more of any string of regular characters. The wildcard - represents exactly 1 of any character.

In this use of a wildcard, a single location is being specified.

S cd /home/je*/notes

Since there is only one directory in /home that starts with je, the wildcard is translated to fill in the rest of the directory name and the cd command succeeds.

In this example, use the same wildcard to specify multiple files which match a pattern.

S ls chapter*

chapter1.txt chapter3.txt chapter5.txt

chapter2.txt chapter4.txt

There are multiple matches, but the ls command is capable of handling multiple files, so the command again succeeds.

d) **Simple ways to create a file:** To create a file there are three commands. Touch, cat and echo. Below is given description and their examples of each command.

- **touch** command with any extension to create file, this command will create an empty file touch.txt in your current directory as an example below.

  $ sudo touch touch.txt

  $ sudo touch touch.docx

- **cat** command to create file, this command will create an empty file cat.txt in your current directory as an example below, but you must add text in the file.

$ cat > cat.txt

Add the text below.

This file has been created with cat command

To save the file hit Ctrl + d, and to see the file type command below.

- **echo** command to create file, this command will create a file echo.txt in your current directory as an example below, but you should add text in the line command.

  $ echo "This file has been created with echo command" > echo.txt

e) Showing the contents of a file-cat, more, less, head, tail:

- **cat:** can be used to join multiple files together and print the result on screen (it will not show page by page)

  Example:

  cat 01.txt

  to display the contents of file 01.txt

  cat 01.txt 02.txt

  to display the contents of both files

  cat file1.txt file2.txt > file3.txt

  Reads file1.txt and file2.txt and combines those files to make

  cat note5 >> notes – attach note5 to notes

  cat >> file1 – add additional data in file1

- **more:** to view a text file one page at a time, press spacebar to go to the next page

  more filename : show the document one page at a time

  more -num filename : show the document page few lines as specified bu (-num)

  example : more -10 filename will show 10 lines for every page

- **less:** is much the same as more command except:
  i. You can navigate the page up/down using the less command and not possible in more command.
  ii. You can search a string in less command. (use /keywordto search)
  iii. "more" was fairly limited, and additional development on "more" had stopped
  iv. it uses same functions as vi editor

  Example: less filename

- **head:** displays the first ten lines of a file, unless otherwise stated.

  Examples:

head myfile.txt – Would display the first ten lines of myfile.txt.

head -15 myfile.txt – Would display the first fifteen lines of myfile.txt.

- **tail:** display the last part of the file

  Examples: tail filename

  tail -n filename : display the last n lines of the file