UNIT - II

Topics :- (i) Introduction to shell
(ii) Various shell
(iii) shell customization
(iv) vi editor
(v) Linux files and file structures
(vi) listing displaying and printing files.
(vii) Managing directories
(viii) File and directory operations.
(ix) Essential linux commands
(x) Internal and external commands
(xi) Archiving and compressing files

Listing Displaying and printing files

Q what is command in Linux?

→ Ans → A command is an instruction given to shell, to perform some specific task.

The general format of command is —

command -options arg-list

Note — command options and arguments must be seperated by white space or tab.

options must be preceded by - (minus).

we can combine options with only one minus sign.

for eg:- wc -l -w -c a.c

OR

wc -lwc a.c

command line :- A command entered in one line along with options and arguments, that line is klas command line.

Multiple commands we can write in single line but we need to seperate them using semicolon (;).

for eg - $ Date ; $ wc a.c

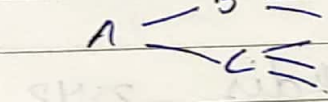
→ The Linux commands can be grouped into following categories —

(1) Directory oriented commands.

(1.1) ls

for eg:- ls abc

options — a — hidden files
l — detailed file info
r — reverse order
t → sorted according to last modification time

R → 

p → put slash after each directory
s → no. of storage blocks used by each file
x → lists contents by lines in sorted order
F → marks executable files with *

wild card character

* — Any no. of characters
? — for single character

for example :-

$ ls abc*

$ ls *x

$ ls ?abc (ending at abc)

$ ls [abc][1-5] will list abc1, abc2 abc3, abc4 abc5

(2) mkdir — to create directory

$ mkdir xyz

current - dir
↓
xyz.

$ mkdir xyz xyz/abc

current - dir
↓
xyz
↓
abc

$ mkdir -p a/b/c/d

current - dir
↓
a
↓
b
↓
c
↓
d

(3) rmdir → to remore directory

Syntax

$ rmdis [-p] <diseclory - name>

for example

$ rmdir abc this will remore abc

$ rmdis abc/ab/c this will remore c only

$ rmdir -p abc/ab/c this will remore c, ab and abc.

(4) cd → to change directory

formal :- $ cd <dir - name>

$ cd /home/abc

$ cd .. represents the parent directory

(5) pwd — represent full pathname for the current directory.

(6) find

find or searches directory tree to search files according to given search criteria.

Syntax :-

find <path-list> <selection-criteria> <action>

for example:-

$ find /home/abc -name "*.java" -print

options can be used

-name <filename> selects the file specified in file name.

-user <username> selects files owned by user name specified in <username>

-type d select directories

-mtime selects files that have been modified on exactly n minute / more than n times / less

7. du — (disk usage)

Syntax — du [-options] [<dir-name>]

$ du → reports the disk space for the current directory. (6)

a Displays counts for all files

b Size in bytes

c Displays o/p along with grand total of all arguments.

K Size in Kilo Bytes

m Size in megabytes

(8) df — (disk free)

reports available free space.

file oriented commands

(1.) **cat** :- (used to display the contents of specified file)

Syntax

```
cat [-options] <filename>
```

options can be -

- -s supress warning about non existing file
- -d list the sub-directory entry only.
- -b numbers non-blank o/p lines
- -n numbers all lines.

example

```
$ cat a.c
```

```
$ cat a.c b.c
```

Note with the help of cat command we can create new file.

$ cat > filename

" type the text
want to write
in file given in
file name.

In the last press ^d
to stop.

for example:-

$ cat > xyz.txt

Hi
How are you
come
we will see movie

^d

This will create a file named xyz.txt in the current directory.

——— * o * ———

② cp : (copy command)

Syntax $ cp file1 file2

OR

$ cp file1 dir1

for eg:-

$ cp abc.txt xyz.txt

the contents of abc.txt is copied to xyz.txt

Syntax: $ cp [-option] s-f d-f

-i° prompt before over-writing d-f.

p preserve all information (owner, groups, permissions)

R recursively copies file in all sub-directories.

Example $ cp abc.txt xyz
↓ ↓
file directory

3. rm (remove)

→ used to remove a file from the specified directory.

→ To remove a file user must have write permission on that directory.

Syntax
rm <-options> <filename>

-r → Deletes all

-i → prompt before deleting

-f → removes write-protected files also.

eg:-

```
$ rm abc.txt
```

→ It will delete file abc.txt from the current directory.

→ Current directory still exists.

```
$ rm -f /usr/vishu
```

→ Delete all the files and subdirectories of the specified directory /usr/vishu

→ vishu is also deleted.

4. mv — move command to rename file.

Syntax

```
$ mv <old-file-name> <new-name>
```

eg.

$ mv a.c b.c

→ Now a.c is renamed as b.c

→ Note that the user must have execute and write permission to rename a file.

5. wc (world - count)

displays No. of lines
No. of words
No. of characters

Syntax :

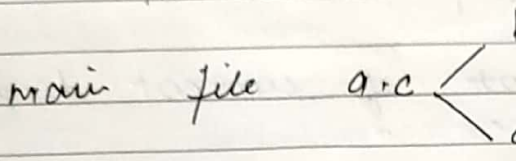
```
wc [-options] <file-name>
```

where options may be -l lines
-w words
-c characters

6) ln : (link) creates link for the given file name

ln <file-names> <additional-file names>

ln a.c b.c d.c



If you modify b.c or d.c, the a.c will automatically modify.

7) File :- displays general classification of a specified file.

Syntax

file <filename>

for eg:- It shows the contents of the specified file is

→ Ascii text.
→ c prog. text
→ data
→ seperate executable
→ empty or others

for eg:- $ file a.c

o/p ASCII text

for eg:- $ file *

* → means content of current directory

& cmp (compare)

→ used to compare two files.

Syntax:

cmp <filename1> <filename2>

$ cmp file1.txt file2.txt

$ cat file1.txt

hi
I am vishu

o/p. Character 1, Line 3

$ cat file2.txt

hi
I am vishu
How are you?

⑨ comm (common)

→ uses two sorted files as argument and reports what is common.

Syntax

```
comm [-options] <file-name1> <file-name2>
```

o/p
column1 lines unique to filename1
column 2 lines unique to filename2
column 3 lines common for both file

- options can be -

1	supresses	listing	of	column1
2	"	"	"	2
3	"	"	"	3

U - II

INTERNAL AND EXTERNAL commands

Linux commands are classified into two categories :-

(i) Internal commands
Eg :- cd, source, fg

(ii) External commands
Eg :- ls, cal-

Let us took at internal and external commands in detail -

→ **INTERNAL commands**

→ Internal commands are those commands which are built into the shell.

→ For shell's built-in commands the execution speed is very high, Because for built-in commands, no process needs to be spawned for executing it.

→ **EXTERNAL COMMAND**

→ External commands are not built into

the shell. These are executables present in seperate files.

→ when an external commands has to be executed, the new process has to be spawned and the command get executed.

for eg:- "cat" command

when we execute cat command /usr/bin/cat get executed.

How to get the list of internal commands?

→ we can get list of internal commands only if we are working with bash shell.

→ Bash shell has a command called "help" which will list out all the built in shell commands.

```
$ help
```

How to find out whether a command is internal or external?

- with the help of "type" command we can check whether the command is internal or external.

Eg :- $ type cd

o/p cd is a shell built-in command

$ type cat

o/p cat is /bin/cat

NOTE For the internal commands, the type command will clearly say its shell built-in.

But for external command, it gives path of the command from where it is executed.

Difference between Internal and External commands :-

→ The big difference between internal and external commands is their "performance".

→ Internal commands are much

more faster compare to external commands, for the simple reason that no process needs to be spawned for an internal command.

If a script contain more external commands, then performance of that script is affected by it. But actually we always not have a choice / to choose an internal command over an external command.

However a careful look at our scripting practices, we can find some places where we can avoid external commands.

for eg:-

Q. Add two numbers -

we can use :-

```
z = `expr $x + $y`
```

OR

we can use :-

```
let z = x + y
```

NOTE :- 'let' is a shell built-in command (i.e internal command)

'expr' is an external command.

* o * —— * o *

"File compression and archiving"

→ It is useful to store a group of files in one file for.....

→ Easy backup

→ for transfer to another computer

→ for transfer to another directory on same computer

→ It is also useful to compress large files, because compressed files takes up less disk space and downloaded faster via the internet.

→ It is important to understand the distinction between an archive file and a compressed file.

ARCHIVE FILE

→ collection of files and directories stored in one file.

→ An archive file is a

→ An archive file is not compressed file.

→ It uses the same amount of disk space as all the individual files and directories combined.

compressed file

A compressed file is a collection of file and directories that are stored in one file, and stored in a way that uses less disk space, than all the individual files and directories combined.

If disk space is a concern compress rarely used files or place such files in a single archive file and compress it.

[9] USING FILE ROLLER :=

→ Linux includes a graphical utility called "file roller"

→ File roller can compress, decompress and archive files.

→ It has a simple interface and extensive help documentation.

→ To start file roller select-

→ Archive Manager

from the application (main menu on the pannel) then select-

→ System tools

sub-menu.

→ File roller is also integrated into the desktop environment and Nautilus.

→ If we are using a file manager we can double click the file to unarchive or decompress to start file Roller.

→ The file roller browses window appears with the decompressed / unarchive file in a folder for us to extract or browse.

[b] Decompressing and unarching with file Roller →

To unarchive and/or decompress a file click the "OPEN" button on the main toolbar.

→ A file menu pops-up, allowing us to choose the archive we want to manipulate.

→ for example :- If we have a file called fav.tar.gz located in home directory highlight the file and click OK.

→ File appears in the main file Roller browser window as a folder which we can Navigate by double clicking the folder icon.

→ File Roller preserves all the directories and sub-directories structure which is convenient if we are looking for a particular file in the archive. We can extract individual files or entire archive by clicking "Extract" button, choosing the directory in which we have to save the unarchived files and

click OK.

[c] Creating an archive with file Roller :-

→ File Roller allows you to create archive file from our files and directories.

→ To create a new archive, click NEW on the toolbar.

→ A file browser pops up, allowing you to specify an archive name and the compression technique.

For eg :-

→ we may choose a Tar compressed with gzip (.tar.gz) format- from the drop down menu and type the name of the archive file we want to create. click OK and now a new archive is ready to be filled with files and directories

→ To add files to our new archive click add, which open a browser window that we can navigate to find the file or

directory to add to the archive, click Add, when we are finished and click archive ⇒ close to close the archive.

[d] compress files at the shell prompt.

→ Linux provides the bzip2, gzip and zip tools for compression from a shell prompt.

→ the bzip2 compression tool is recommended because it provides the most compression and is found on most unix-like operating system.

→ The gzip compression tool can also be found on most unix like O.S.

→ To transfer files between Linux and other O.S such as ms windows use zip because it is compatible with the compression utilities available for windows.

→ Command for compress a file.

Light Green
Date
Page

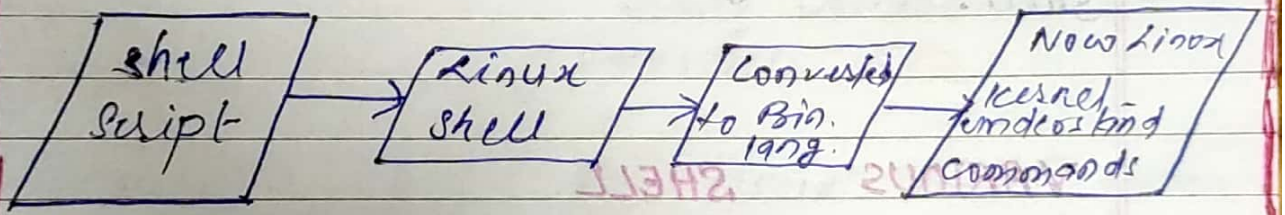
UNIT - II

INTRODUCTION TO SHELL :=

→ The shell is a linux system mech for the communication between the users and the system.

→ It is a command interpretor that interprets the user commands and conveys them to the kernel which executes them.

→ Shell accepts your instruction or commands in English and translate it in computers binary language.



→ There are many shell variables in Linux, including Bash (Bourne again shell), Bourne shell, c shell (tcsh) Korn shell.

→ Most popular shell is bash.

→ Front of this bash shell is $ symbol.

→ Linux shells are very popular and powerful because i/p and o/p can

be redirected using < and >.

→ Linux is quite feasible to have multiple shell installed.

→ Users can pick what they prefer.

→ On linux the standard shell that is always installed as /bin/sh is called bash.

→ We can check the version of bash with following commands.

$ /bin/bash --version

It will give information about shell.

VARIOUS SHELL

The following table offers a brief summary of some of the common shell available -

SHELL NAME	HISTORY
sh (Bourne)	The original shell from every version of unix.

csh, tcsh, zsh	The c shell and its derivates originally created by Bill Joy of Barkeley UNIX frame. The c shell is the most popular type of shell often bash and Korn shell.
ksh, pdksh	The korn shell and its public domain cousin is written by David Korn. This is default shell on many commertial UNIX version.
bash	The Linux shell from the GNU project- or Bourne Again shell has the advantage that the source code is freely available and even if it is not- currently running on our system, it has probably been ported to it. bash has many similarities to the korn shell.

Light Green
Date electronics 03
Page

vi editor

Linux provide various type of editors like ex, sed, ed, vi, vim, xvi, nvi, elvis etc.

With the help of these editors we can create and edit our files.

files like data files and program files and text files.

The famous one is vi editor. The full form of vi editor is visual full screen editor. It was created by Bill joy at the university of California at Beskeley.

STARTING VI

→ This editor can be involked by typing vi at the doller ($) prompt.

→ If we specify a file name as an argument to vi then it will edit the specified file, if file exist.

→ vi <file name>

→ A status line at the bottom of the screen i.e 25th line shows the file name, current line, character position.

→ vi + <line number> <file name>

eg:- vi +10 a.sh.

It will edit a.sh and places the cursor at 10th line of the file.

Modes of VI editor

The vi editor works in three modes.

INSERT MODE

→ text is entered in this mode.
→ Any key pressed in this mode is treated as text.
→ When can enter in this mode by pressing any keys.

{i, I, a, A, o, O, r, R, s, S

& i, a, o, r, s }

COMMAND MODE

→ This is default mode. when we start vi editor the system is in command mode.

→ All the commands on vi editor should be used in this mode.

→ We can enter into this mode from insert mode by pressing Esc key (escape key) and from execution mode by pressing enter (↵) key.

Ex — Mode

→ The ex - mode command like saving files, find, replace etc can be entered at the last line of the screen in this mode.

→ from command mode to Ex-mode press colon (:).

commands for working with vi

1) INSERT COMMANDS

i - insert before cursor

I - At the beg. of line

a - Append after cursor

A - Append at the end of the current line.

o - Insert Blank line below the current line

O - Insert a black line above the current line

(2) Delete commands

x - Deletes a character at cursor position

<n>x - n no of characters

X - Deletes a character before cur-position.

[n]X - Deletes n no. of characters

dw - cursor position to the end of the word

dW - ignores any punctuation

db - cursor position to beg. of the word.

dB - ignores punctuations

dd - delete current line

<n>dd - specified no. of lines from current line.

d↵ - current line and the following line.

Linux Files and File structure

* Already done in unit-first *.

Listing Displaying and Printing Files