Unit No II- Linux Commands

***Files and Directories management command:-***

***There are following different Linux files and directories management commands:-***

1. ***Mkdir [Make Directories]:-***

we need to organize information and files by creating one or more new directories, such as under home directory. System administrators also create new directories to hold programs, data, utilities, and other information.

The *mkdir* command is used to create a one or more directories. You must have write permission in the parent directory in order to create a directory .you create a number of subdirectories with one mkdir command i.e. Create a directories tree.

***Syntax: # mkdir [ option ] directories name***

**Option:**

**-m** Set the access permission new directories.

**-p** Create intervening parent directories if they don’t exist.

***Example :***

$ mkdir -m 777 sangola // Set the access permission

$ mkdir -p sangola/bcs/student // Créate parent directorios

$ mkdir bsc bcs bca ba

$ mkdir msc

Mkdir: failed to make directory “msc”; permission denied

**This can happen due to these reasons:**

a) The directory **msc** may already exist.

b) There may be an ordinary file by that name in the current directory.

c) The permission set for the current directory doesn’t permit the creation of files

and directories by the user.

1. ***Rmdir [ Remove Directories ]***

Delete the directories (not the contents of directory). Directories are deleted

from the parent directory but it must be empty.

***Syntax : $ rmdir [ option ] directories name.***

***Option:* -p** Remove directories and any intervening parent directories that

become empty as a result. Useful for removing subdirectories tree.

***Example :* $ rmdir sangola**

Here, ‘sangola’ is the directory under the current working directory. It should not be the current working directory. ‘rmdir’ command removes the directory only if the directory is empty.

A sub-directory cannot be removed with rmdir unless it is empty and one is

positioned in its parent directory.

**$ rmdir -p sangola/bcs/student**

1. ***Cd [ Changing Directories ]***

This command used for change current working directory. Cd without a directory name changes back to the home directory of user. In cd the references

directory single dot **( . )** to reference to your working directory , instead of using its

pathname. Double dot **( .. )** represents to pathname of its parent directory.

(ex - $ ls **.** & $ cat **../**repport ).

***Syntax: $ cd directory name***

***Example:***

**$ pwd**

/home/sangola

**$ cd info // change the directory.**

**$ pwd**

/home/sangola/info

Here, the cd command changes the directory ‘/home/sangola’ in to the sub-directory called ‘info’. The cd command used without arguments gets the user back to his HOME directory from wherever he is

$ **cd // return to home directory.**

/home/sangola

In this command pathname required because home directory is current directory.

1. ***Pwd [Present or print Working Directory ]***

This command display the **p**resent **w**orking **d**irectory path. You can see a current working directory.

**Syntax:-**

$ pwd [option ] <Enter>

**Options:-**

-P : The pathname printed will not contain symbolic links.

-L : The pathname printed may contain symbolic links

***Example:* $ pwd**

/home/computer/stud\_info

***File***

This command identifies the basic file type i.e. ordinary, directory, device file. The file recognizes text file, shell program code and object code. it also identifies

DOS executable, compressed files, PDF documents & even empty files. Using the \*

to signify all files.

***Example :*  $ file \***

Ban: empty

Bin: directory

Cal.out: ASCII text

Clean: c program text

Story: English text

***Listing , Displaying and Printing Commands***

***Ls [ List of File ] :***

This command used to list the files and directories from current directory. By

using proper options; you can also find the permissions of files, size of a file. This

command allows you to check ownership, differentiate file type.

***Option:-***

**Ls** Lists in alphabetical order all no hidden files in the current directory.

**–a** Lists all files in the current directory including hidden files.

**-lh** List Files and shows size with Human Readable Format

**–r** Lists in reverse alphabetical order all non hidden files in the current directory.

**–F** Lists all files by type. The character at the end of each file indicates the file

type.

Example forward slash (/) represents a directory & ‘@’ represents a linked file.

**–i** Lists file with inode number .an inode number represent the location of a file on

a Volume .two or more files with the same inode number are two different names

For the identical file.

**–l** Lists all the files in the current directory including the current directory (.) and

the parent directory (..) also lists the size, owner and permission associated with

each files In what known as long listing format.

**-n** To display **UID** and **GID** of files and directories

**-S** Sort Files by File Size and displays file size in order

**–t** Lists files by the last time they were being changed; most recent files are list first.

**–u** Lists files by the last time they were accessed; most recent files are listed first.

***Example:***

**$ ls // List Files using ls with no option**

0001.pcap Desktop Downloads index.html

**$ ls –a // View Hidden Files**

bashrc Documents .gconfd install.log .nautilus

$ **ls –lh // List Files and shows size with Human Readable Format.**

-rw-------. 1 root root 1.6K Jul 31 02:17 anaconda-ks.cfg

$ **ls –F // List Files and Directories with ‘/’ Character at the end**

Desktop/ Downloads/ index.html Pictures/ Templates/ bca@ Public~

***File type Meaning***

- Normal file

d Subdirectory

l Symbolic link

b Block device file

c Character device file

***Cat [Create , Display , Concatenate & Append ]***

It is used to **create , display ,concatenate and append to the file** .it also

act on a stream you can supply input to cat not only by specifying a filename ,but also from the output of another command .it is mainly used to display the contents of a small file on terminal. This accepts more than one filename as arguments.

***Option:***

**-v** If you have nonprinting ASCII characters in your input, you can use [-v]

option to Display these characters.

**-n** Indicates numbers lines. This facility used for programmers its finds errors

Using numbers line.

***Example:*  $ cat > bsc**

Vidnyan mahavidyalaya sangola

C^

You input into a bsc file. i.e we are created bsc file then we append it.

**$ cat >> bsc**

Near ST Stand sangola

This college under solapur university solapur.

C^

**$ cat bsc**

Vidnyan mahavidyalaya sangola

Near ST Stand sangola

This college under solapur university solapur.

Display output in a bsc file using cat command.

**$ cat chap1 chap2 (concatenation of two file)**

The show content of second file is shown immediately after the first file

without any header information. i.e. cat concatenate the two files.

***More & Less:-***

The cat command only display small files on screen. the big output files are

displayed on screen that use a more & less command .*it is a standard pager in Linux system.*

***More:-***

The more command is used to read text files and display it . For example, we could do this:

$ more ac.txt <Enter>

The effect of this to let you read the file “abc.txt” . It probably will not fit in one screen, so you need to know how to "turn pages". Here are the basic commands,

* **q** --- quit more
* **spacebar** --- read next page
* **return key** --- read next line
* **b** --- go back one page

**less :-**

The less command is used to same as more command, but it has many more features. **less** does not have to read the entire input file before starting, so with large input files it starts up faster than text editors.

$ less program.sh <enter>

if( 2 ge 3) {

print "greater\n";

}else {

print "lesser\n";

}

***Example:*  $ more ebook**

You will see contents of ebook on the screen one page at a time .at the bottom of the screen you will also see the filename and percentage of the file that has been viewed

**File and Directory Operations:-**

***Find :***

Find command used to search and locate list of files and directories based on

conditions you specify for files that match the arguments. Find can be used in variety of conditions like you can find files by **permissions**, **users**, **groups**, **file**

**type**, **date**, **size** and other possible criteria. There are several possible options that

specify to search by type and also criteria; it then searches within the directories

listed & their subdirectories for files that meet these criteria. Find command divide

into five parts.

**Part I:** Basic Find Commands for Finding Files with Names

**Part II:** Find Files Based on their Permissions

**Part III:** Search Files Based On Owners and Groups

**Part IV:** Find Files and Directories Based on Date and Time

**Part V:** Find Files and Directories Based on Size

***Syntax: $ find directory \_list (path \_list selection) - option criteria***

**Find operates:**

a) First, it recursively examine all file in the directories specified in pathname.

b) It then matches each file for one or more selection criteria.

c) It takes some action on these selected files.

***Option:***

|  |  |
| --- | --- |
| **-name**  **filename** | *To specify the name of the file (including wildcards) to be used for*  *searching.* You can also use range as part of the wildcards. If you  want to use wildcard characters, you must specify them within  quotes. For example, "test\*" will find all files starting with test.  If you specify, "test [1-2]", you will find files that start with test  and have 1 or 2 as the last characters such as test1 and test2. |
| **-size Number** | *To specify the size of the file to be used for searching.* The file size  specified in blocks. if you want to match size of files less than the  specified size, use a(**-)** in front of the size& if you want to match  size of files greater than the specified size use (**+).**   **c**: bytes   **k**: Kilobytes   **M**: Megabytes   **G**: Gigabytes   **b**: 512-byte blocks |
| **-atime number**  **-amin n** | *To search for files that has been accessed in the specified number*  *of 24-hour periods.*  *Find files accesses in last 1 hour.* |
| **-mtime**  **number**  **-mmin n** | *To search for files that has been modified in the specified number*  *of 24-hour periods.*  *Files modified in last hour* |
| **-cmin n** | *Find files changed in last N minutes.* |
| **-type file type** | *To search for a specific type of file.* The following is a list of types  that can be used:  o b--Block special file  o c--Character special file  o d--Directory  o f--Regular file  o l--Symbolic link  o p--FIFO (a named pipe)  o s--Socket |
| **-user** | *Is the username to search for files whose owner matches the*  *specified username.* |
| **-perm** | *Permission to search for files with specified permission.* The  permission is specified as an octal number of up to 3 digits. If the  permission is not preceded by a - (hyphen), an exact match of the  permission specified is made with the file permissions. For  example, if you want to search for files with owner read  permission, use -perm -400. |
| **-newer**  **filename** | *To search for files that has time of modification later than that of*  *the specified filename.* |
| **-group group**  **name** | *To search for files that belongs to the specified group.* |
| **-inum Number** | *to search for files whose i-node number matches the specified inode*  *number* |
| ***-links Number*** | *To search for files with a specified number of links.* |
| **-ls** | *to print the current path name* along with the following  attributes:  o i-node number  o Size  o Permissions  o Number of hard links  o User name  o Group name  o Modification time |
| **exrc cmd**  **name** | Execute command **cmd** on a file. |

**Part I – Basic Find Commands for Finding Files with Names**

1. Find Files Using Name in Current Directory

Find all the files whose name is **tech.txt** in a current working directory.

**$ find . -name tech.txt**

./tech.txt

2. Find Files under Home Directory

Find all the files under **/home** directory with name **tech.txt**.

**$ find /home -name tech.txt**

/home/tech.txt

3. Find Files Using Name and Ignoring Case

Find all the files whose name is **tech.txt** and contains both capital and small

letters in **/home** directory.

**$ find /home -iname tech.txt**

./tech.txt

./Tech.txt

4. Find Directories Using Name

Find all directories whose name Tech is in **/** directory.

**$ find / -type d -name Tech**

/Tech

5. Find PHP Files Using Name

Find all **php** files whose name is **tech.php** in a current working directory.

**$ find . -type f -name tech.php**

./tech.php

6. Find all PHP Files in Directory

**$ find . -type f -name "\*.php"**

./tech.php

./login.php

./index.php

**Part II – Find Files Based on their Permissions**

7. Find Files with 777 Permissions

Find all the files whose permissions are **777**.

**$ find . -type f -perm 0777 –print**

8. Find all Files without 777 Permissions

**$ find / -type f ! -perm 777**

10. Find all Read Only Files

**$ find / -perm /u=r**

11. Find all Executable Files

**$ find / -perm /a=x**

12. Find and remove single File

To find a single file called **tech.txt** and remove it.

**$ find . -type f -name "tecmint.txt" -exec rm -f {} \;**

13. Find all Empty Files

To file all empty files under certain path.

**$ find /tmp -type f –empty**

14. Find all Empty Directories

To file all empty directories under certain path.

**$ find /tmp -type d –empty**

15. File all Hidden Files

To find all hidden files, use below command.

**$ find /tmp -type f -name ".\*"**

**Part III – Search Files Based On Owners and Groups**

16. Find Single File Based on User

To find all or single file called **tech.txt** under **/** root directory of owner root.

**$ find / -user root -name tech.txt**

17. Find all Files Based on User

To find all files that belongs to user **bcs** under **/home** directory.

**$ find /home -user bcs**

18. Find all Files Based on Group

To find all files that belongs to group **Developer** under **/home** directory.

**$ find /home -group developer**

19. Find Particular Files of User

To find all **.txt** files of user **root** under **/home** directory.

**$ find /home -user root -iname "\*.txt"**

**Part IV – Find Files and Directories Based on Date and Time**

20. Find Last 50 Days Modified Files

To find all the files which are modified **50** days back.

**$ find / -mtime 50**

21. Find Last 50 Days Accessed Files

To find all the files which are accessed **50** days back.

**Part V – Find Files and Directories Based on Size**

22. Find 50MB Files

To find all **50MB** files, use.

**$ find / -size 50M**

23. Find Size between 50MB – 100MB

To find all the files which are greater than **50MB** and less than **100MB**.

**$ find / -size +50M -size -100M**

24. Find and Delete 100MB Files

To find all **100MB** files and delete them using one single command.

**$ find / -size +100M -exec rm -rf {} \;**

25. Find Specific Files and Delete

Find all **.mp3** files with more than **10MB** and delete them using one single

command.

**$ find / -type f -name \*.mp3 -size +10M -exec rm {} \;**

***Example:* $ find . –name ‘[A-Z]\*’ –print**

**$ find / -name a.sh –print //Search a a.sh file and print it.**

/home/program/a.sh

/home/script/a.sh

/home/stud/bcs/a.sh **$ find / -atime 50**

***Cp [ Copying Files & Directories ]***

The cp command copies a file or a group of files. It creates an exact image of

the file on disk with a different name. To copy one or more files into directory. This

command requires at least two arguments in the command line.

***Syntax: $ cp source filename destination filename***

***Option:***

**-r** Copies subdirectories from one directory to another. The copied Directory

includes all its own directories.

**-i** it will ask for confirmation before overwriting a file

**-n** Don’t overwrite an existing file

**-t** Copy all source argument into directory.

**-p** Preserve attributes of file or directory while copying

**-s** Create Soft link to a file or directory (instead of copying)

**-v** To see copy progress.

***Example:* $ cp page book**

If the destination file book doesn’t exist, it will first be created before copying text place. Otherwise it will simply over write without any warning of the system. Cp often used with short hand notation **( . ) dot** to signify the current directory as the destination .for instance to copy the file dept from to your current directory.

**$ cp /home/ram/dept ./dt //destination dt is a file**

**$ cp linux java web mis bcs // bcs is directory.**

This command linux , java , web , mis copies a file into bca directory.

You can copy multiple file using ( \* ) metacharacter if in the current directory having common string then you copies all files to the first letter is A.

**$ cp A\* stream**

**$ cp -r prog new // new must not exist***.*

If the file results exists, the command

**$ cp -i test results**

Overwrite results? (y/n)

If you choose y (yes), the file will is overwritten. If you choose n (no), the file results

aren’t changed.

***Mv [ Rename or Move Files & Directories ]***

The mv command renames file (move file) a file one directory to another

directory. It renames a file or directory. It moves a group of files to a different

directory. It keeps the inode number same even after moving it to a different name.

If you move the file to a different filesystems, the inode number will be different.

***Syntax : $ mv source filename destination filename***

***Option:***

**-I** Interactive Before overwriting an existing file, prompt the user for

confirmation. If this option is not specified, mv will silently overwrite files.

**-u** Move only the files that don’t exist in the destination directory

**-v** To see moving progress.

***$ mv information student***

The first argument ‘information ‘ is current filename for a renaming. You want

rename to a file then you move in a new file ‘ student ‘ that is second argument. You

can move more than one file into a one file.

***Example : $ mv add sub mul result***

***$ mv \*.c program***

***rm [ Remove Files & Directories ]***

This used for removing one or more files & directories.

***Syntax: $ rm [option ] filenames***

***Option :***

**-d** Remove files.

**-v** To see removing progress. Explain what being done

**-i** it will ask for confirmation before removing a file Yes (remove the file) or

No(don’t remove the file).

**[-r]**

**Or**

**[-R]** If file is a directory, remove the entire directory & all its contents,

including Subdirectories.

**\*** (\*) wildcard (asterisk) can represent all file names. It use for delete all files

***Example :* $ rm dev // dev file removed.**

**$ rm -i mydata bcs**

Remove mydata? N

Remove bcs? Y

You can use a [-i] option, you are prompted separately for each file & ask with each

file. If you can enter Y the file will be removed & enter the N file will not be

removed. The user can decide to remove the file.

**$ rm \* // Delete all files**

To delete all files from #HOME/personl-files, enter:

**$ rm $HOME/personl-files/\***

***Ln [Linking Files ]***

This command used linking for files. Create links for files allowing them to

be accessed by different names. Links may be “Hard link” or “Soft link as symbolic

link”. A hard link gives two names to a one file & a soft link creates s a second file

which acts as a shortcut to the first file.

***Syntax: $ ln [option] source filename destination file / directory name***

The first from links source name or destination name. Where destination name is usually either a new filename or by default a file in the current directory. With same name as source name. if a destination name is an existing file it is

overwritten. If a destination name is an existing directory a link named source

name is created in that directory.

The second from creates links in destination directory each link having

the same name as the file specified.

***Hard Link:***

Every file has a single hard link that gives the file its name. When we create a

hard link, we create an additional directory entry for a file***.***

***Hard links have two important limitations:***

 A hard link cannot reference a file outside its own file system. This means a link

cannot reference a file that is not on the same disk partition as the link itself.

 A hard link cannot reference a directory.

***Example: $* ln class student**

The option **[-i]** option to ls shows that they have the same inode no. meaning that are actually one & the same file.

**$ ls -li class student**

29518 -rwxr-xr-x 2 comp metal 302 step 4 09:45 class

29518 -rwxr-xr-x 2 comp metal 302 step 4 09:45 student

You can link multiple files ( i.e create a link for each ), but then destination filename must be a directory.

**$ ln mca ?? mca\_dir**

If mca ?? Matches 20 files, there will be 20 linked filenames in mca\_dir.

***Soft Link:***

Linux OS recognizes the data part of this special file as a reference to another file

path. The data in the original file can be accessed through the special file, which is

called as Soft Link.

Symbolic links were created to overcome the limitations of hard links. Symbolic

links work by creating a special type of file that contains a text pointer to the

referenced file or directory A file pointed to by a symbolic link. For example, if you

write something to the symbolic link, the referenced file is also written to when you

delete a symbolic link, only the link is deleted, not the file itself. If the file is deleted

before the symbolic link, the link will continue to exist but will point to nothing.

*The ln command create symbolic link using [-s] option.*

**$ ln -s notes notes.sym**

**$ ln -li notes notes.sym**

2951 -rw-r--r-- 1 comp group 302 step 4 09:45 notes

2955 lrwxr-xr-x 1 comp group 67 step 4 09:57 notes.sym  notes

You can identify symbolic link by the character (l) seen before the permission field.

The pointer notation point’s notes suggest that notes.sym contains the pathname for

the file name notes. But actually is not present in the notes.sym. if we remove notes

file then the loose the file containing the data. In that case notes.sym would print to

nonexistent file & become a dangling symbolic link.

***Differences between soft and hard links***

***Soft Links:***

 Soft Links can be created across file systems.

 Soft link has a different inode number than the original file.

 On deleting the original file, soft link cannot be accessed.

 Soft link needs extra memory to store the original file name as its data.

 Source file need not exist for soft link creation.

 Can be created on a file or on a directory.

 Access to the file is slower due to the overhead to access file.

***Hard Links:***

 Hard links can be created only within the file system.

 Hard links have the same inode number as the original file.

 On deleting the original file, hard linked file can still be accessed.

 Hard links do not need any extra data memory to save since it uses links

 Source file should exist.

 Can be created only on files, not on directories.

 Access to the file is faster compared to soft link.

**Printing the files:-**

***Lpr [To Print File]***

To send files to the printer connected to your system for printing. A lists of files may be used as arguments. if you want to print several files at once then you specify more than one file on the command line.

***Option:*** [ -p ] : this option to specify a printer.

[ -t ] : this option use this title

[ -# ] : this use for prints a numbers of copies.

[ -m ] : mail message after completion.

***Example:* $ lpr mydata *// print file mydata***

**$ lpr –p hp 1050 add.sh *//prints on hp1050 printer***

**$ lpr –t “write a shell program “ add.sh *// uses this title***

**$ lpr -#5 add.sh *// prints 5 copies****.*

**$ lpr –m bsc.ps *// mail message after completion***

**$ lpr mydata bcs**

The user prints both files mydata & bca.

***Lpq [Display Printer Queue]***

Check the print spool queue for status of print jobs. For each job ,display

username ,rank in the queue ,file names ,job number and total file size in bytes .

***Syntax: $ lpq [ option ] [ +interval ]***

***Example : $* lpq**

Printer: lp@blob

Queue: 2 printable jobs

Server: pid 29998 active

Unspooler: pid 29999 active

Status: waiting for subserver to exit at 09:43:20.699

Rank Owner/ID Class Job Files Size Time

1 elly@blob+997 A 997 (STDIN) 129 09:42:54

2 elly@blob+22 A 22 /etc/profile 917 09:43:20

***Lprm [ Remove jobs from Printer Queue]***

This is use for cancel print jobs. Jobs ID can be obtained from lpq ;if no job is

specified ,cancel the current job on the default printer.

**Syntax :  *$ lprm [ option ] [ job id ]***

***Option :*** [ - ] : remove all job available to the user .

[-P ] : printer specify printer queue. if no printer the default printer used.

***Example :* $ lprm 31 *// Removes job number 31***

**$ lprm - *// removes all jobs owned by user***

Different Other Command

Wc Command [ Count a word, character and line ]

The wc filter counts the number of lines, words and characters in the specified file or files. It is capable of accepting input directly from the keyboard. By entering wc without any arguments, it waits for the user to type in the input. By default, wc prints all the three counts lines, words, bytes. It comes with three options, which allow user to obtain the number of lines, words or characters individually or in any desired combination.

Syntax: $ wc [ option ] filename

Option: [ -l ] : Count lines in the file.

[ -w ] : Count words in the file.

[ -c ] : Count characters in the file.

A line is any group of characters not containing a new line. A word is a group of

characters not containing a space, tab or new line. A character is the smallest unit of information and includes a space and new lines.

Example: $ wc bcs

Linux other command

Wc Command [ Count a word, character and line ]

The wc filter counts the number of lines, words and characters in the specified file or files. It is capable of accepting input directly from the keyboard. By entering wc without any arguments, it waits for the user to type in the input. By default, wc prints all the three counts lines, words, bytes. It comes with three options, which allow user to obtain the number of lines, words or characters individually or in any desired combination.

Syntax: $ wc [ option ] filename

Option: [ -l ] : Count lines in the file

[ -w ] : Count words in the file

[ -c ] : Count characters in the file.

A line is any group of characters not containing a new line. A word is a group of characters not containing a space, tab or new line. A character is the smallest unit of information and includes a space and new lines.

Example: $ wc bcs

1 3 24 bcs

Line word character filename

$ wc -l bcs // count numbers of lines.

bcs

$ wc –w bcs // count numbers of words.

bcs

$ wc -c bcs // count numbers of characters.

24 bcs

Cmp Command : [ Comparing Two Files ]

It is the simplest file comparison tools. The cmp (compare) command compares two files, character (byte) by character, and provides the location of the differences to the screen.If two files are identical the cmp command displays no message. But when two files are different then he displays the position in the file where the first difference occurs.

Syntax: $ cmp [ filename 1 ] [ filename 2 ]

Example: If you want to compare the new file with the old file:

$ cmp new\_prog1 old\_prog1

Diff Command: [ Converting one file to other ]

It can be used to display files difference. It tells you which lines in one files have to be changed to make the two files identical.

Syntax: $ diff First filename second filename

Example: $ diff bcs bsc

Cal Command

To see the calendar of any specific month or a complete year. The facility is totally accurate and takes into account the leap year adjustments that took place in the year.

Syntax: $ cal [ Month ] [ year ]

Example:$ cal 1 2018

It displays calendar 1 month of 2018

Date command

It can display the current date. It shows the date and time to nearest second.

Syntax: $ date

example: $ date

Monday September 16 11.30.15 IST 2011