**Linux Commands: -**

**cd command: -**

cd command in Linux is known as change directory command. It is used to change current working directory.

Syntax: - cd [directory name]

cd / - This command is used to change directory to root directory.

cd directory1/directory2/directory3 – This command is used to move to directory3 directly without writing cd command 3 times to move to directory3.

cd ~ -- This command is used to change directory to home directory.

Only cd command also will work as cd ~ command i.e. it will move us to the home directory.

cd .. – This command is used to move to the parent directory of current directory, or the directory one level up from the current directory ‘..’ represents parent directory.

cd ‘directory name’ – This command is used to move to a directory which has white spaces in its name.

**mkdir command: -**

mkdir command in Linux allows the user to create directories (also referred to as folders in some operating systems). This command can create multiple directories at once.

mkdir directory name – will create a directory in the current working directory.

mkdir -p directory1/directory2/directory3 – The p flag enables the mkdir command to create parent directories as necessary. If the directories exist, no error is specified.

**rmdir command: -**

rmdir command in Linux is used to remove empty directories from the filesystem in Linux. The rmdir command removes each and every directory specified in the command line only if these directories are empty.

rmdir directory1 – removes the directory with the name directory1 in the current working directory if they are empty.

rmdir directory1 directory2 directory3 – removes the directory1, directory2, directory3 in the current working directory if they are empty.

rmdir -p directory/directory1 – In this option each of the directory argument is treated as a path name of which all the components will be removed, if they are already empty starting from the last component.

**rm command: -**

rm stands for remove here. rm command is used to remove objects such as files, directories and so on from the file system like Linux. By default, rm does not remove directories. This command normally works silently, and you should be very careful while running rm command because once you delete the files then you are not able to recover the contents of files and directories.

rm filename.txt – removes the file filename in the current working directory.

rm filename1.txt filename2.txt – removes both filename1 and filename2 in the current working directory.

rm -i filename.txt – The -i option makes the rm command ask the user for confirmation before removing each file, you have to press y for confirm deletion, any other key leaves the file un-deleted.

rm -f filename.txt – rm prompts for confirmation removal if a file is write protected. The -f option overrides this minor protection and removes the file forcefully.

Note: -f option of rm command will not work for write – protected directories.

rm -r directory (Recursive deletion) – With -r option rm command performs tree-walk and will delete all the files and sub-directories recursively of the parent directory. At each stage it deletes everything it finds. Normally, rm wouldn’t delete the directories but when used with this option it will delete.

Deleting a file whose name starts with a hyphen symbol (-): - To remove a file whose name starts with a dash, you can specify a double dash separately before the filename.

rm -- -file.txt

**cat command: -**

cat (concatenate) command is very frequently used in Linux. It reads the data from the file and gives their content as output. It helps us to create, view, concatenate files.

cat filename – It will show the content of the given filename.

cat file1 file2 – This will show the content of the file1 and file2.

cat -n filename – To view the content of a file preceding with line numbers.

cat > file1.txt – To create a new file.

cat source > destination – Copy the contents of one file to another.

cat -s file.txt – To suppress the repeated empty lines in the output.

cat file1 >> file2 – To append the contents of one file to the end of another file.

cat -e file1.txt – To append each line of a file with $ at the end.

**vi editor: -**

The vi editor (short for visual editor) is a text editor which is available on almost all Linux systems. vi has no menus but instead uses the combination of key strokes in order to accomplish commands. You can use vi editor to edit (add, change and delete text) an existing file or to create a new file from scratch. You can also use this editor to just read a text file.

Operating modes of vi editor: -

While working with vi editor you would come across the following two modes.

Command mode: - This mode enables you to perform administrative tasks such as saving the files, executing commands, moving the cursor, cutting and pasting lines or words, and searching and replacing. In this mode whatever you type is interpreted as a command.

Insert mode: - This mode enables you to insert text into the file. Everything that’s typed in this mode is interpreted as input and finally it is put in the file.

The vi always starts in command mode.

To enter text, you must be in insert mode.

To come to insert mode just type i when you are in command mode.

To get out of the insert mode, press the Esc key which will put you back into the command mode. If you are not sure which mode you are in, press the Esc key twice, and then you will be in command mode.

Note: -vi commands are case sensitive.

vi filename – Creates a new file if it already does not exist, otherwise opens an existing file.

Getting out of vi: -

The command to quit of vi is :q. Once in the command mode, type colon and ‘q’ followed by enter. If your file has been modified in any way, the editor will warn you of this and not let you quit. To ignore this message to quit out of vi without saving is :q!. This lets you exit vi without saving any of the changes.

The command to save the contents of the editor is :w. You can combine the above command with the quit command, or use :wq and return.

The easiest way to save your changes and exit vi is with the ZZ command. When you are in the command mode, type ZZ. The ZZ command works the same way as the :wq command.

If you want to specify /state any particular name for the file, you can do so by specifying after :w. For example, if you want to save the file you were working on with different filename called filename2, you would type :w filename2 and return.

Moving within the file: -

To move around within the file without affecting you text, you must be in the command mode.

k – moves the cursor up one line.

j – moves the cursor down one line.

h – moves the cursor to the left one-character position.

l – moves the cursor to the right one-character position.

$ -- Positions the cursor at the end of the line.

0 or |--Positions the cursor at the beginning of the line.

G – moves to the last line of the file.

nG – moves to the nth line of the file.

Deleting characters: -

x – Deletes the character under the cursor location.

dw – Delete from the current cursor location to the next word.

D – Delete from the current cursor position to the end of the line.

dd – Deletes the line the cursor is on.

Most commands in vi can be prefixed by the number of times you want the action to occur. For example, 2dd deleted the two lines the cursor is on.

Searching in vi: -

For a String search, the / and ? commands are used. When you start these commands, the command just typed will be shown on the last line of the screen, where you type the particular string to look for.

These 2 commands differ in the direction where the search takes place.

1. The / command searched forwards (downwards) in the file.
2. The ? command searches backwards (upwards) in the file.

The n and N commands repeat the previous search command in the same or in the opposite direction respectively.

**grep command: -**

The grep command searches for a particular pattern of characters and displays all the lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression. grep stands for globally search for regular expression and print out.

grep -c file.txt – This print only a count of the lines that match the pattern.

grep -h file.txt – Display the matched line, but do not display the filenames.

grep -i file.txt – Ignores, case for searching.

grep -l file.txt file1.txt – Display list of filenames only.

grep -n file.txt – Display the matched lines and their line numbers.

grep -v file.txt – This prints out all the lines that do not match the pattern.

grep -w file.txt – Matches whole word only.

grep -o file.txt – Print only the matched parts of a matching line, with each such part on a separate line.

grep ^unix file.txt – The ^ regular expression pattern specifies the start of a line. This can be used in grep to match the lines which start with the given string or pattern. In the above example it searches the file.txt and displays all the lines that start with the pattern unix.

grep unix$ file.txt – The $ regular expression pattern specifies the end of a file. This can be used in grep to match the lines which end with the given string or pattern. In the above example it searched the file.txt and displays all the lines that end with the pattern unix.

grep -A 1 unix file.txt – This command will display the line in which the unix pattern was found in the file.txt along with the line after it.

grep -B 1 unix file.txt – This command will display the line in which the unix pattern was found in the file.txt along with the line before it.

grep -C 1 unix file.txt – This command will display the line in which the unix pattern was found in the file.txt along with the line before and after it.

If we give grep -C 3 it will display 3 lines before and after the pattern.

**crontab command: -**

The crontab is a list of commands that you want to run on a regular schedule. Crontab stands for cron table because it uses the job scheduler cron to execute tasks. Cron itself is named after chronos the Greek word for time. Cron is the system process which will automatically perform tasks for you according to a schedule. The schedule is also called the crontab, which is also the name of the program to edit that schedule.

Linux crontab format: -

MIN HOUR DOM MON DOW CMD

Crontab fields and allowed ranges: -

Field Description Allowed Value

MIN Minute Field 0 to 59

HOUR Hour Field 0 to 23

DOM Day of Month 1 – 31

MON Month Field 1 – 12

DOW Day of Week 0 – 6

CMD Command Any Command to be executed.

The below crontab will execute the date command and append the data to the data.txt file on 6th June 07: 35 PM on every day of the week.

35 19 6 6 \* date >> /Users/dharani-kumar/Documents/data.txt

To create a cron tab use the crontab -e command.

Here you can create the job you want to run and at what time.

To view the crontab entries use crontab -l command.

To delete the crontab entries use the crontab -r command.

We can execute scripts also using crontab. All we need to do is in the place of command in the syntax give the script location.

**ps and kill command in Linux: -**

Every command we execute in the Linux a separate process will be running in the background and it will be assigned a process id. We can use the ps command to know the list of background running/System process. If a process is running for long time we can use the kill command to kill that process.

ps -elf – gives complete information about the background process.

Normally this command will be used along with grep to know whether a particular process is running or not like MySQL, Oracle etc.

If we want to kill a process, then we will use the kill process\_id command. The process id of the process is given by the ps -ef command.

If you are not able to kill a process using kill command, then use kill -9 command which can kill that process. -9 means forcefully we are killing that process.

**screen command in Linux: -**

screen command in Linux provides the ability to launch and use multiple shell sessions from a single session. When a process is started with screen, the process can be detached from session and then can reattach the session at a later point of time. When the session is detached, the process that was originally started from the screen is still running and managed by the screen itself. The process can then re-attach the session at a later point of time, and the terminals are still there, the way it is left.

screen – It will start a new window within the terminal with the default name given by the terminal.

screen -S name – This command is used to create a screen with a name attached to it.

To detach a screen, we need to give Ctrl + a d in the screen.

screen -ls – This command gives the list of all screens.

screen -r Process id of the screen -- To reattach a screen.

Once we are in the screen use the Ctrl + a k to kill the screen.

You can also use the below command to terminate the screen.

screen -X -S process\_id\_of\_the\_screen quit.

**cp command: -**

cp stands for copy. This command is used to copy file or group of files or directory. It creates an exact image of a file on a disk with different file name. cp command requires at least two filenames in its arguments.

Syntax: -

cp source destination

cp source directory

cp source1 source2 source3 sourcen Directory

First and second syntax is used to copy source file to destination file or directory.

Third syntax is used to copy multiple source files to directory.

cp command works on three principal modes of operation and these operations depend upon number and type of arguments passed in cp command.

1. Two filenames: -If the command contains two filenames, then it copy the content of first file to the second file. If the second file does not exist, then first it creates one and then content is copied to it. But if the second file exists then it simply overwrites without any warning. So be careful while you choose the destination file name.
2. One or more arguments: - If the command has one or more arguments, specifying filenames and following those arguments an argument specifying directory name then this command copies each source file to the destination directory with the same name, created if not existed but if already existed then it will be overwritten.

Note: -For this case last argument muse be a directory name. For the above command to work the destination directory must exists because cp command won’t create it.

1. Two directory names: -If the command contains two directory names, cp copies all files of the source directory to the destination directory, creating any files or directories needed. This mode of operation requires an additional operation, typically R to indicate the recursive copying of the directories. In the above command cp behavior depend upon whether destination directory exist or not. If the destination directory does not exist, cp creates it and copies the content of source directory recursively as it is. But if the destination directory exists then copy of the source directory becomes sub directory of the destination directory.

cp -i source.txt destination.txt – i stands for interactive copying. With this option system warns the user before overwriting the destination file. cp prompts for response if you press y then it overwrites the file and with any other option leave it uncopied.

cp -f source.txt destination.txt – If the system is unable to open the destination file for writing operation because use don’t have writing permission for this file then by using -f option with the cp command, destination file is deleted first and then copying of content from source to destination file.

Copying using \* wildcard: -Suppose we have many text documents in a directory and wants to copy it to another directory, it takes lot of time to copy 1 by 1 but by using \* wildcard it becomes simple.

cp \*.txt directory

**mv command: -**

mv stands for move. mv is used to move one or more files or directories from one place to another in file system in Linux. It has two distinct functions.

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

No additional space is consumed on the disk during renaming. This command normally works silently means no prompt for confirmation.

mv source destination – If the destination file does not exist, it will be created. In the above command mv simply replaces the source filename in the directory with the destination filename. If the destination file exists it will be overwritten, and the source file will be deleted.

The -i option makes the command ask the user for confirmation before moving a file that would overwrite an existing file, you have to press y for confirmation.

**ssh command: -**

ssh stands for secure shell. It is a protocol used to securely connect to a remote host/server.

ssh is secure in the sense that it transfers the data in encrypted form between the host and the client. It transfers the input from the client to the host and relays back the output. ssh runs at TCP/IP port 22.

ssh command consists of 3 different parts.

1. ssh command instructs the system to establish an encrypted secure connection with the host machine.
2. User name represents the user account that is being accessed on the host.
3. Host refers to the machine which can be a computer or a router that is being accessed. It can be an IP address or domain name.

Note: After logging into the host computer, commands will work as if they were directly written on the host terminal.

Syntax: - ssh username@destination

**scp command: -**

scp command is used to copy files between servers in a secure way.

Basic Syntax of scp: -

scp source\_file\_name username@destination\_host:destination\_folder

Provide the detail information of the scp process using -v parameter: -

Basic scp command without parameter will copy the files in the background. Users will see nothing unless the process is done or some error appears. You can use -v parameter to print debug information into the screen. It can help you debugging connection, authentication and configuration problems.