Linux Commands

ls

ls command with no option list files and directories in bare format where we won't be able to view details like file types, size, modified date and time, permission and links etc.

1) To list Hidden Files	**ls -a**
2) List files and directories in long format (more detailed list)	**ls -l**
3) List Files with Human Readable Format with option -lh	**ls -lh**
4) List files and directories in reverse order	**ls -lr**
5) List files and directories in reverser last modified order (recent will come at the end)	**ls -lrt**

cat

cat command (short for "concatenate") allows us to create single or multiple files, view contain of file, concatenate files and redirect output in terminal or files.

1) Display Contents of File	**Cat <file_name>**
2) View Contents of Multiple Files in terminal	**Cat <file1> <file2>** **Or** **Cat <file1>; cat <file2>**
3) Create a File with Cat Command	**cat >file**
4) Display Line Numbers for the content in File while printing	**cat -n file**

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

more file.txt # show the document one page at a time

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

less **less** is much the same as more command except:

a) You can navigate the page up/down using the less command and not possible in more command.
b) You can search a string in less command. (use /keyword to search)
c) "more" was fairly limited, and additional development on "more" had stopped
d) it uses same functions as vi editor

head

a) **head -n n file.txt** The head command, as the name implies, print the top N number of lines of the given input. By default, it prints the first 10 lines of the specified files.

head -n 5 file.txt (or) head -5 file.txt

Prints first 5 line of file.txt

b) **head -c num file.txt** Prints the first 'num' bytes from the file specified. Newline count as a single character, so if head prints out a newline, it will count it as a byte. **num** is mandatory to be specified in command otherwise displays an error.

head -c 6 file.txt

tail

a) **tail -n n file.txt** The tail command, as the name implies, print the last N number of lines of the given input. By default, it prints the first 10 lines of the specified files.

tail -n 5 file.txt (or) tail -5 file.txt

Prints last 5 line of file.txt

b) **tail -c num file.txt** Prints the last 'num' bytes from the file specified. Newline count as a single character, so if tail prints out a newline, it will count it as a byte. **num** is mandatory to be specified in command otherwise displays an error.

tail -c 6 file.txt

c) To continuously print a file which is getting updated like log file.

tail -f filename

cd

Use the "cd" command to go to a directory/change the current directory. For example, if you are in the home folder, and you want to go to the downloads folder, then you can type in "cd downloads".

mkdir

Use the mkdir command when you need to create a folder or a directory.

mkdir /home/sample

rmdir

rmdir is use to delete empty directory, to remove non-empty directory use rm

rm

Use the **rm** command to delete files and directories. But rm cannot simply delete a directory. Use "**rm -r**" to delete a directory. In this case, it deletes both the folder and the files in it.

rm file_name
rm -r directory

touch

The touch command is used to create a file. It can be anything, from an empty txt file to an empty zip file.

touch filename.txt

To create randomly 1 to 100 files

touch filename{1..100}.txt

pwd

Command to know present working directory. When you first open the terminal, you are in the home directory of your user. To know which directory you are in, you can use the "**pwd**" command. It gives us the full path, which means the path that starts from the root.

mv mv command to move files through the command line. We can also use the mv command to rename a file.

mv old_filename new_filename

move all files from source to destination

mv /source_path/*.* /destination_path/

locate The locate command is used to locate a file in a Linux system, just like the search command in Windows. This command is useful when you don't know where a file is saved or the actual name of the file. Using the -i argument with the command helps to ignore the case

locate file_name

cp cp command used to copy files from source location to destination. It takes two arguments: The first is the location of the file to be copied, the second is where to copy.

cp /source_path/filename /destination_path/

rev To reverse print line or string

cat filename | rev

echo “linux” | rev

grep

'global search for the regular expression': The grep command is a filter that is used to search for lines matching a specified pattern and print the matching lines to standard output.

1) **Anchor Characters**: **'^' and '$'** at the beginning and end of the pattern are used to anchor the pattern to the start of the line, and to the end of the line respectively.

   Example: "^Name" matches all lines that start with the string "Name". The strings "\<" and "\>" are used to anchor the pattern to the start and end of a word respectively.

2) **Wildcard Character: '.'** Is used to match any character.

   Example: "^.$" will match all lines with any single character.

3) **Escaped Characters:** Any of the special characters can be matched as a regular character by escaping them with a '\'.

   Example: "\$*" will match the lines that contain the string "$*"

4) **Character Range:** A set of characters enclosed in a '[' and ']' pair specify a range of characters to be matched.

   Example: "[aeiou]" will match all lines that contain a vowel. A hyphen can be used while specifying a range to shorten a set of consecutive characters. E.g. "[0-9]" will match all lines that contain a digit. A carat can be used at the beginning of the range to specify a negative range. E.g. "[^xyz]" will match all lines that do not contain x, y or z.

5) **Repetition Modifier:** A '*' after a character or group of characters is used to allow matching zero or more instances of the preceding pattern.

   ? The preceding item is optional and matched at most once.
   ***** The preceding item will be matched zero or more times.
   + The preceding item will be matched one or more times.
   {n} The preceding item is matched exactly n times.
   {n,} The preceding item is matched n or more times.
   {,m} The preceding item is matched at most m times.
   {n,m} The preceding item is matched at least n times, but not more than m times.

The grep command supports a number of options for additional controls on the matching:

-i: performs a case-insensitive search.
-n: displays the lines containing the pattern along with the line numbers.
-v: displays the lines not containing the specified pattern.
-c: displays the count of the matching patterns.

1) **Match all lines that start with 'hello'. E.g: "hello there"**
   grep "^hello" file1

2) **Match all lines that end with 'done'. E.g: "well done"**
   grep "done$" file1

3) **Match all lines that contain any of the letters 'a', 'b', 'c', 'd' or 'e'.**
   grep "[a-e]" file1

4) **Match all lines that do not contain a vowel**

```
grep “[^aeiou]” file1
```

5) **Match all lines that start with a digit following zero or more spaces. E.g: “ 1.” or “2.”**

```
grep “ *[0-9]” file1
```

6) **Searching in all files recursively using grep -r**

```
grep -r "ramesh" *
```

7) **Search multiple patterns**

```
grep -v -e "pattern" -e "pattern"
```

8) **Show only the matched string**

```
grep -o "is.*line" demo_file
```

9) **Show line number while displaying the output using grep -n**

```
grep -n "pattern" file_name
```

10) **Checking for the given string in multiple file patterns (demo_file, demo_file1)**

```
grep "this" demo_*
```

11) **pattern that starts with “lines” and ends with “empty” with anything in-between**

```
grep "lines.*empty" demo_file
```

12) **Checking for full words, not for sub-strings using grep -w**

```
grep -iw "is" demo_file
```

13) **Displaying lines before/after/around the match using grep -A, -B and -C**

a. **Display N lines after match**

```
grep -A <N> -i "string" FILENAME
```

b. **Display N lines before match**

```
grep -B <N> -i "string" FILENAME
```

c. **Display N lines around match**

```
grep -C 2 -i "Example" demo_text
```

14) **Searching with logical operators**

a. **Grep OR Using \|**

```
grep 'pattern1\|pattern2' filename

grep -E 'pattern1|pattern2' filename

grep -e pattern1 -e pattern2 filename
```

b. **Grep AND using -E ‘pattern1.*pattern2’**

```
grep -E 'pattern1.*pattern2' filename

grep -E 'pattern1.*pattern2|pattern2.*pattern1' filename

grep -E 'pattern1' filename | grep -E 'pattern2'
```

sed

SED command in UNIX is stands for stream editor and it can perform lots of function on file like, searching, find and replace, insertion or deletion. Though most common use of SED command in UNIX is for substitution or for find and replace. By using SED you can edit files even without opening it, which is much quicker way to find and replace something in file, than first opening that file in VI Editor and then changing it.

Using below file for example:

```
cat > file.txt
```

unix is great os. unix is opensource. unix is free os.
learn operating system.
unix linux which one you choose.
unix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

1) Replacing or substituting string

sed 's/unix/linux/' file.txt

by default, Replace/substitute only the first occurance of string (unix -> linux) in each line

output:

linux is great os. unix is opensource. unix is free os.
learn operating system.
linux linux which one you choose.
linux is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

2) Replacing the nth occurrence of a pattern in a line

sed 's/unix/linux/2' file.txt

Use the /1, /2 etc flags to replace the first, second occurrence of a pattern in a line.
Replace/substitute only the second occurance of string (unix -> linux) in each line

output:

unix is great os. linux is opensource. unix is free os.
learn operating system.
unix linux which one you choose.
unix is easy to learn.linux is a multiuser os.Learn unix .unix is a powerful.

3) Replacing all the occurrence of the pattern in a line

sed 's/unix/linux/g' file.txt

The substitute flag /g (global replacement) specifies the sed command to replace all the occurrences of the string in the line.

you may want to consider using gi instead of g in order to ignore character case:

sed 's/unix/linux/gi' file.txt

output:

linux is great os. linux is opensource. linux is free os.
learn operating system.
linux linux which one you choose.
linux is easy to learn.linux is a multiuser os.Learn linux .linux is a powerful.

4) Replacing from nth occurrence to all occurrences in a line

sed 's/unix/linux/3g' file.txt

The above sed command replaces the third, fourth, fifth… "unix" word with "linux" word in a line. (from 3rd occurance to all till end of line)

output:

unix is great os. unix is opensource. linux is free os.
learn operating system.
unix linux which one you choose.
unix is easy to learn.unix is a multiuser os.Learn linux .linux is a powerful.

5) Replacing string on a specific line number

sed '3 s/unix/linux/' file.txt

restricting the sed command to replace the string on a specific line number (3th line)

output:

unix is great os. unix is opensource. unix is free os.
learn operating system.
linux linux which one you choose.
unix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

6) Duplicating the replaced line with /p flag

sed 's/unix/linux/p' file.txt

The /p print flag prints the replaced line twice on the terminal. If a line does not have the search pattern and is not replaced, then the /p prints that line only once.

output:

linux is great os. unix is opensource. unix is free os.
linux is great os. unix is opensource. unix is free os.
learn operating system.
linux linux which one you choose.
linux linux which one you choose.
linux is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

linux is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

7) Printing only the replaced lines

sed -n 's/unix/linux/p' file.txt

Use the -n option along with the /p print flag to display only the replaced lines. Here the -n option suppresses the duplicate rows generated by the /p flag and prints the replaced lines only one time.

output:

linux is great os. unix is opensource. unix is free os.
linux linux which one you choose.
linux is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

8) Write a sed command to print the lines that do not contain the word "run"?

sed -n '/run/!p' file.txt

9) Replacing string on a range of lines

sed '1,3 s/unix/linux/' file.txt

Here the sed command replaces the lines with range from 1 to 3.

Output:

linux is great os. unix is opensource. unix is free os.
learn operating system.
linux linux which one you choose.
unix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

sed '2,$ s/unix/linux/' file.txt

Here $ indicates the last line in the file. So the sed command replaces the text from second line to last line in the file.

Output:

unix is great os. unix is opensource. unix is free os.
learn operating system.
linux linux which one you choose.
linux is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful

10) Deleting lines from a particular file

- To Delete a particular line, say 5 in this example

sed '5d' file.txt

- To Delete a last line

  sed '$d' file.txt

- To Delete line from range x to y

  sed '3,6d' file.txt

- To Delete from nth to last line

  sed '12,$d' file.txt

- To Delete pattern matching line

  sed '/pattern/d' file.txt

11) Write A Command to Replace the Word "apple" With "(apple)" In A File?sed s/apple/(&)/ < file.txt

12) Write A Command to Switch the Two Consecutive Words "apple" And "mango" In A File?sed 's/(apple) (mango)/2 1/' < file.txt

13) Write a command to replace the character '/' with ',' in a file?

sed 's/\//,/' < file.txt (or) sed 's|/|,|' < file.txt

14) Replace a line with line number with newline in a file

sed 'Nc <newline text>' <file_name>
where, N is the line number
Example: replace 5th line with newline in a file
Sed '5c This is the new line to be replaced' test.txt

find

find command can be used to find files and directories and perform subsequent operations on them. It supports searching by file, folder, name, creation date, modification date, owner and permissions. By using the '-exec' other commands can be executed on files or folders found.

SYNTAX: find <location_to_find> [options]

- **. current directory**
- **.. One previous directory from current**

1) **Search a file with specific name.**
   find . -name file.txt

2) **search a files in multiple directories.**
   find . /home /user -name file.txt

3) **Search a file with specific name with ignore case**
   find . -iname file.txt

4) **Search only files containing name**
   find . -type f -iname file.txt

5) **Search only directories containing name**
   find . -type d -iname file.txt

6) **Search for empty files and directories.**
   find . -empty

7) **Search for file with entered permissions (655).**
   Find . -perm 655

8) **Search text within multiple files.**
   find ./ -type f -name "*.txt" -exec grep 'search_string' {} \;

9) **find files by last modification time**
   find . -mtime days
   find . -mtime 1 # 24 hours
   find . -mtime -7 # modified less than 7 days (7 days to till)
   find . -mtime 7 # modified before 7 days (all files modified before 7 days)
   find . -mtime +50 –mtime -100 # Last 50-100 Days Modified Files
   find . -mtime -7 -type f # just files last 7 days
   find . -mtime -7 -type d # just dirs. Last 7days

10) **Find Last 50 Days Accessed Files**
    find . -atime 50

11) **Find Changed Files in Last 1 Hour**
find / -mmin -60

12) **Find Accessed Files in Last 1 Hour**
find / -amin -60

13) **find and copy**
find . -type f -name "*.mp3" -exec cp {} /tmp/MusicFiles \;
 # cp *.mp3 files to /tmp/MusicFiles

14) **Find Files Without 655 Permissions**
find . -type f ! -perm 655 (or) find . -type f -not -perm 655

15) **Find Single File Based on User**
find . -user root -name file.txt

16) **Find all Files Based on Group**
find . -group developer

17) **Find Files by memory**

Find 50MB Files	**find / -size 50M**
Find Size between 50MB – 100MB	**find / -size +50M -size -100M**
Find and Delete 100MB Files	**find / -size +100M -exec rm -rf {} \;**

18) **Find files between date**
find -type f -newermt "2011-12-22" \! -newermt "2011-12-24"
(or)
find -type f -newermt "2011-12-22 00:00:00" \! -newermt "2011-12-24 13:23:00"

19) **Find files by user**
find /home -user $USER -mtime -90

CUT

The **cut** command in UNIX is a command for cutting out the sections column with using delimiter from each line of files and writing the result to standard output.

cut -d "delimiter" -f (field number) file.txt

a) To cut the data with " " (space) as delimiter and print first column data

cut -d " " -f 1 state.txt

b) To cut the data with " " (space) as delimiter and print first to fourth column data

cut -d " " -f 1-4 state.txt

awk

The **awk** command in UNIX is a command for cutting out the sections column with using field separator from each line of files and writing the result to standard output.

WHAT ARE THE OPERATIONS OF AWK?

a) Scans a file line by line
b) Splits each input line into fields
c) Compares input line/fields to pattern
d) Performs actions on matched lines

Useful For:

a) Transform data files
b) Produce formatted reports

Programming Constructs:

a) Format output lines
b) Arithmetic and string operations
c) Conditionals and loops

Consider the following text file as the input file for all cases below.

$cat > employee.txt

ajay manager account 45000
sunil clerk account 25000
varun manager sales 50000
amit manager account 47000

1) Default behavior of Awk: By default Awk prints every line of data from the space separated field of file.

awk '{print $1}' employee.txt

(or)

awk -f " " '{print $1}' employee.txt

Output:

Ajay
Sunil
Varun
Amit

2) To print the last column of a file using NF – Number of Fields/Columns
How to get the last word from a line in file.

awk '{print $NF}' employee.txt

Output:

45000
25000
50000
47000

3) print all columns except the first one column:

awk '{$1=""; print $0}' file

4) print all columns except the last one column:

awk '{$NF=""; print $0}' file

Alternative: rev file | cut -f2- | rev

5) Print the lines which matches with the given pattern

awk '/manager/ {print}' employee.txt

(or)

awk '/manager/ {print $0}' employee.txt

Output:

ajay manager account 45000
varun manager sales 50000
amit manager account 47000

NR NR command keeps a current count of the number of input records. Remember that records are usually lines. Awk command performs the pattern/action statements once for each record in a file.

NF NF command keeps a count of the number of fields within the current input record.

FS FS command contains the field separator character which is used to divide fields on the input line. The default is "white space", meaning space and tab characters. FS can be reassigned to another character (typically in BEGIN) to change the field separator.

RS RS command stores the current record separator character. Since, by default, an input line is the input record, the default record separator character is a newline.

OFS OFS command stores the output field separator, which separates the fields when Awk prints them. The default is a blank space. Whenever print has several parameters separated with commas, it will print the value of OFS in between each parameter.

ORS ORS command stores the output record separator, which separates the output lines when Awk prints them. The default is a newline character. print automatically outputs the contents of ORS at the end of whatever it is given to print.

6) Another use of NR built-in variables (Display Line From 3 to 6)

awk 'NR==2, NR==3 {print NR,$0}' employee.txt

Output:

2 sunil clerk account 25000
3 varun manager sales 50000

7) To find the length of the longest line present in the file:

awk '{ if (length($0) > max) max = length($0) } END { print max }' employee.txt

8) To count the lines in a file using awk

awk 'END { print NR }' employee.txt

9) Printing lines with more than 10 characters:

awk 'length($0) > 10' employee.txt

10) To find/check for any string in any column:

awk '{ if($3 == "B6") print $0;}' geeksforgeeks.txt

11) To print any non-empty line if present

awk 'NF > 0' geeksforgeeks.txt

12) To print all columns from the nth to the last

awk '{for(i=3;i<=NF;++i)print $i}'

13) To print variables in awk

awk -v a="$var1" -v b="$var2" 'BEGIN {print a,b}'

date

date command is used to display the system date and time. date command is also used to set date and time of the system. By default the date command displays the date in the time zone on which unix/linux operating system is configured. You must be the super-user (root) to change the date and time.

date

Output:

Tue Oct 10 22:55:01 PDT 2017

1) Using –date option for displaying past dates:

	COMMAND	OUTPUT
a) Date and time of 2 years ago.	**date --date="2 year ago"**	Sat Oct 10 23:42:15 PDT 2015
b) Date and time of 5 seconds ago.	**date --date="5 sec ago"**	Tue Oct 10 23:45:02 PDT 2017
c) Date and time of previous day.	**date --date="yesterday"**	Mon Oct 9 23:48:00 PDT 2017
d) Date and time of 2 months ago.	**date --date="2 month ago"**	Thu Aug 10 23:54:51 PDT 2017
e) Date and time of 10 days ago.	**date --date="10 day ago"**	Sat Sep 30 23:56:55 PDT 2017

2) Using –date option for displaying future date:

	COMMAND	OUTPUT
a) Date and time of upcoming particular week day.	**date --date="next tue"**	Tue Oct 17 00:00:00 PDT 2017
b) Date and time after two days.	**date --date="2 day"**	Fri Oct 13 00:05:52 PDT 2017
c) Date and time of next day.	**date --date="tomorrow"**	Thu Oct 12 00:08:47 PDT 2017
d) Date and time after 1 year on the current day.	**date --date="1 year"**	Thu Oct 11 00:11:38 PDT 2018

3) -s or –set Option: To set the system date and time -s or –set option is used.

date --set="date to be set"

date --set="Tue Nov 13 15:23:34 PDT 2018"

date

Output:

Tue Nov 13 15:23:34 PDT 2018

4) List of Format specifiers used with date command:

%D	Display date as mm/dd/yy.
%d	Display the day of the month (01 to 31).
%a	Displays the abbreviated name for weekdays (Sun to Sat).
%A	Displays full weekdays (Sunday to Saturday).
%h	Displays abbreviated month name (Jan to Dec).
%b	Displays abbreviated month name (Jan to Dec).
%B	Displays full month name(January to December).
%m	Displays the month of year (01 to 12).
%y	Displays last two digits of the year(00 to 99).
%Y	Display four-digit year.
%T	Display the time in 24 hour format as HH:MM:SS.
%H	Display the hour.
%M	Display the minute.
%S	Display the seconds.
%D	Display date as mm/dd/yy.
%d	Display the day of the month (01 to 31).

COMMAND	OUTPUT
date "+%D"	10/11/17
date "+%D %T"	10/11/17 16:13:27
date "+%Y-%m-%d"	2017-10-11
date "+%Y/%m/%d"	2017/10/11
date "+%A %B %d %T %y"	Thursday October 07:54:29 17

Special Variables

These are special shell variables which are set internally by the shell and which are available to the user:

VARIABLE	DESCRIPTION
$0	The filename of the current script.
$n	These variables correspond to the arguments with which a script was invoked. Here n is a positive decimal number corresponding to the position of an argument (the first argument is $1, the second argument is $2, and so on). User flower braces for double or more digits like ${10}, ${100}
$$	The process ID of the current shell. For shell scripts, this is the process ID under which they are executing.
$#	The number of arguments supplied to a script.
$@	All the arguments are individually double quoted. If a script receives two arguments, $@ is equivalent to $1 $2.
$*	All the arguments are double quoted. If a script receives two arguments, $* is equivalent to $1 $2.
$?	The exit status of the last command executed.
$!	The process ID of the last background command.
$_	The last argument of the previous command.

Crontab (Job Scheduler)

The crontab is a list of commands that you want to run on a regular schedule, and also the name of the command used to manage that list. 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 set schedule. The schedule is called the crontab, which is also the name of the program used to edit that schedule.

Linux Crontab Format: * * * * * command to execute

MIN HOUR DOM MON DOW CMD

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.

To view the Crontab entries: **crontab -l**
To edit Crontab Entries: **crontab -e**
To edit crontab entries of other Linux users: **crontab -u username -e**

1) **To schedule a job for every minute using Cron**

```
* * * * * command/script
```

2) **How to Execute a Linux Cron Jobs Every Second Using Crontab.**

You cannot schedule an every-second cronjob. Because in cron the minimum unit you can specify is minute. In a typical scenario, there is no reason for most of us to run any job every second in the system.

3) **To schedule a background Cron job for every 10 minutes.**

```
*/10 * * * * /home/maverick/check-disk-space
```

4) **Schedule a Job for More Than One Instance (e.g. Twice a Day)**

executes the specified script at 11:00 and 16:00 on every day

```
00 11,16 * * * /home/ramesh/bin/incremental-backup
```

5) **Schedule a Job for Specific Range of Time (e.g. Only on Weekdays)**

This example checks the status of the database everyday (including weekends) during the working hours 9 a.m – 6 p.m

```
00 09-18 * * * /home/ramesh/bin/check-db-status
```

6) **Cron job to run on the last day of the month**

```
55 23 28-31 * * [[ "$(date --date=tomorrow +\%d)" == "01" ]] && myscript.sh
0 23 28-31 * * [ $(date -d +1day +%d) -eq 1 ] && myscript.sh
```

7) Cron special keywords and its meaning

Keyword	Equivalent
@yearly	0 0 1 1 *
@daily	0 0 * * *
@hourly	0 * * * *
@reboot	Run at startup.

a) To schedule a job for first minute of every year using @yearly

@yearly /home/maverick/bin/annual-maintenance

b) To schedule a Cron job beginning of every month using @monthly

@monthly /home/maverick/bin/tape-backup

c) To schedule a background job every day using @daily

@daily /home/maverick/bin/cleanup-logs "day started"

d) To execute a linux command after every reboot using @reboot

@reboot CMD

Network Commands

Default PORTS:

Service name / Protocols	PORT
1) File Transfer Protocol (**FTP**)	20, 21
2) Trivial File Transfer Protocol (**TFTP**)	69
3) Secure Shell (**SSH**)	22
4) Telnet	23
5) HyperText Transfer Protocol (**HTTP**)	80
6) HTTP with Secure Sockets Layer (**HTTPS/SSL**)	443
7) Simple Mail Transfer Protocol (**SMTP**)	25
8) Domain Name System (**DNS**)	53
9) Dynamic Host Configuration Protocol (**DHCP**)	67, 68
10) Post Office Protocol (**POP3**)	110
11) Internet Message Access Protocol (**IMAP4**)	143
12) Lightweight Directory Access Protocol (**LDAP**)	389
13) Remote Desktop Protocol (**RDP**)	3389

SSH

The ssh command provides a secure encrypted connection between two hosts over an insecure network. This connection can also be used for terminal access, file transfers, and for tunneling other applications.

1) Normal ssh command

ssh username@hostname

username – username of the other server that we are connection
hostname – ip-address / the domain name to the server which we want to connect

Note: after running the above command new to enter the password through console and you get successfully logged-in to another server.

2) To make password less connection use ssh-keygen

Ssh-keygen is a tool for creating new authentication key pairs for SSH. Such key pairs are used for automating logins, single sign-on, and for authenticating hosts.

Let **server1** - 192.168.0.1 and **server2** - 192.168.0.2

STEP 1: Create Authentication SSH-Keygen Keys on 192.168.0.1

ssh-keygen -t rsa

It will ask for inputs like pass phrase at that time just press enter

STEP 2: Create .ssh Directory on – 192.168.0.2

Do normal ssh to server 2 and create .ssh directory

(or)

Create .ssh directory directly using ssh command from server1

ssh server1@192.168.0.1 mkdir -p .ssh

STEP 3: Upload Generated Public in server1 to server2 – from 192.168.0.1 to 192.168.0.2

cat .ssh/key.pub | ssh user@192.168.0.2'cat >> ~/.ssh/authorized_keys'

STEP 4: Set Permissions of authorized_keys on server1 and server2

ssh user@192.168.0.2 "chmod 700 .ssh; chmod 640 .ssh/authorized_keys"

STEP 5: Login from 192.168.0.1 to 192.168.0.2 Server without Password

ssh username@192.168.0.2

To run command directly on remote SERVER

ssh username@IP_address 'commands to run on remote'
ssh username@192.168.0.2 'ls -lrt'
ssh username@192.168.0.2 'df -h'

ping

Ping a remote host by sending packets. Used to check whether the remote server is up and running.

ping hastname
ping IP_addres

curl

curl is a tool to transfer data from or to a server, using one of the supported protocols (HTTP, HTTPS, FTP, FTPS, SCP, SFTP, TFTP, DICT, TELNET, LDAP or FILE). The command is designed to work without user interaction. It offers proxy support, user authentication, FTP uploading, HTTP posting, SSL connections, cookies, file transfer resume, Metalink, and many other features, listed below

1) View curl Version

The -V or --version options will not only return the version, but also the supported protocols and features in your current version.

curl –version

2) Download a File

If you want to download a file, you can use curl with the -O or -o options. The former will save the file in the current working directory with the same name as in the remote location, whereas the latter allows you to specify a different filename and/or location.

curl -O http://yourdomain.com/yourfile.tar.gz # Save as yourfile.tar.gz
curl -o newfile.tar.gz http://yourdomain.com/yourfile.tar.gz # Save as newfile.tar.gz

3) Resume an Interrupted Download

If a download was interrupted for some reason (for example, using Ctrl + c), you can resume it very easily. The use of -C – (dash C, space dash) tells curl to resume the download beginning where it left off.

curl -C - -O http://yourdomain.com/yourfile.tar.gz

4) Download Multiple Files

With the following command you will download info.html and about.html from http://yoursite.com and http://mysite.com, respectively, in one go.

curl -O http://yoursite.com/info.html -O http://mysite.com/about.html

5) Download URLs From a File

If you combine curl with xargs, you can download files from a list of URLs in a file.

xargs -n 1 curl -O < listurls.txt

6) Use a Proxy with or without Authentication

If you are behind a proxy server listening on port 8080 at proxy.yourdomain.com, do.

curl -x proxy.yourdomain.com:8080 -U user:password -O http://your.com/yourfile.tar.gz

7) Query HTTP Headers

HTTP headers allow the remote web server to send additional information about itself along with the actual request. This provides the client with details on how the request is being handled.

To query the HTTP headers from a website, do:

curl -I www.tecmint.com

8) Make a POST request with Parameters

The following command will send the firstName and lastName parameters, along with their corresponding values, to https://yourdomain.com/info.php.

curl --data "firstName=John&lastName=Doe" https://yourdomain.com/info.php

9) Download Files from an FTP Server with or without Authentication

If a remote FTP server is expecting connections at ftp://yourftpserver, the following command will download yourfile.tar.gz in the current working directory.

curl -u username:password -O ftp://yourftpserver/yourfile.tar.gz

10) Upload Files to an FTP server with or without Authentication

To upload a local file named mylocalfile.tar.gz to ftp://yourftpserver using curl, do:

curl -u username:password -T mylocalfile.tar.gz ftp://yourftpserver

11) Specify User Agent

The user agent is part of the information that is sent along with an HTTP request. This indicates which browser the client used to make the request. Let's see what our current curl version uses as default, and let's change it later to "I am a new web browser":

curl -I http://localhost --user-agent "I am a new web browser"

12) Store Website Cookies

Want to see which cookies are downloaded to your computer when you browse to https://www.cnn.com? Use the following command to save them to cnncookies.txt. You can then use cat command to view the file.

curl --cookie-jar cnncookies.txt https://www.cnn.com/index.html -O

netstat

1) **List all tcp ports -** netstat -at
2) **To show both listening and open ports** - netstat -a
3) **List all udp ports** – netstat -au
4) **List only listening ports** – netstat -l
5) **List only listening TCP ports –** netstat -lt
6) **List the statistics/details for all ports -** netstat **-s**
7) **Check port open or not** - netstat -tulpn | grep LISTEN | grep '<port_number>'
8) **List all connections to a port:** netstat -a | grep '<port_number>'

Memory related Commands

free

Free command used to check the used and available space of physical memory and swap memory (ram memory) in KB. The free command displays:

- Total amount of free and used physical memory
- Total amount of swap memory in the system
- Buffers and caches used by the kernel

free

```
             total       used       free     shared    buffers     cached
Mem:       1021628     912548     109080          0     120368     655548
-/+ buffers/cache:     136632     884996
Swap:      4194296          0    4194296
```

To see the size of the memory in (MB) Megabytes use option as -m.

free -m

```
             total       used       free     shared    buffers     cached
Mem:           997        891        106          0        117        640
-/+ buffers/cache:        133        864
Swap:         4095          0       4095
```

top

The top command also gives you a real-time update on how much of your swap space is being used.
One of the things top is very good for is discovering Process ID (PID) numbers of services that might have gotten out of hand. With those PIDs, you can then set about to troubleshoot (or kill) the offending tasks.

vmstat

Another very handy tool to have at your disposal is vmstat. This particular command is a one-trick pony that reports virtual memory statistics. The vmstat command will report stats on: Processes, Memory, Paging, Block IO, Traps, Disks and CPU

/proc/meminfo

The next way to check memory usage is to read the /proc/meminfo file. Know that the /proc file system does not contain real files. They are rather virtual files that contain dynamic information about the kernel and the system.

du

"du" (Disk Usage) is a standard Unix/Linux command, used to check the information of disk usage of files and directories on a machine.

1) Using "-h" option with "du" command provides results in "Human Readable Format". Means you can see sizes in Bytes, Kilobytes, Megabytes, Gigabytes etc.
   du -h
2) The "-c" flag provides a grand total usage disk space at the last line.
   du -ch

df

A quick way to get a summary of the available and used disk space on your Linux system is to type in the df command in a terminal window. The command df stands for "disk filesystem". With the -h option (df -h) it shows the disk space in "human readable" form, which in this case means, it gives you the units along with the numbers.

The output of the df command is a table with four columns. The first column contains the file system path, which can be a reference to a hard disk or another storage device, or a file system connected to the network. The second column shows the capacity of that file system. The third column shows the available space, and the last column shows the path on which that file system is mounted. The mount point is the place in the directory tree where you can find and access that file system.

df -h #Print memory details in human readable format

df - -total -h #prints total memory of system at the end

mount

To mount a file system, you should first create a directory and mount it as shown below.

```
# mkdir /u01
# mount /dev/sdb1 /u01
```

1) **Mount the device of devicename devicename, of type type, at filesystem location destination_directory.**
   # mount -t type devicename destination_directory
2) **Display all current mounts**
   # Mount
3) **List all current mounts of type tmpfs**
   # mount -l -t tmpfs
4) **List all mounts**
   # mount -a

unmount

umount /mnt/usb

PROCESS RELATED COMMANDS

ps

ps command is used to display information about the processes that are running in the system.

ps -ef **#** To view current running processes.

ps -efH **#** To view current running processes in a tree structure. H option stands for process hierarchy.

kill

Use kill command to terminate a process. First get the process id using ps -ef command, then use kill -9 to kill the running Linux process as shown below. You can also use killall, pkill, xkill to terminate a unix process.

Before kill first get the pid (process id to kill)

ps -ef | grep process_name

OUTPUT:

user 7243 7222 9 22:43 pts/2 00:00:00 process

To kill the above process

kill -9 7243

kill (kill -s TERM or kill -15) vs kill -9

Kill -9 : It doesn't give the process a chance to cleanly not graceful close:

1) shut down socket connections
2) clean up temp files
3) inform its children that it is going away
4) reset its terminal characteristics

and so on and so on and so on.

Kill: will generate a SIGTERM signal asking a process to kill itself gracefully i.e , free memory or take care of other child processes. Killing a process using kill will not have any side effects like unreleased memory because it was gracefully killed and also wait 1 or 2 seconds to close the process.

bg

bg sends the specified jobs to the background. A background job is executed simultaneously with fish, and does not have access to the keyboard. If no job is specified, the last job to be used is put in the background. The PID of the desired process is usually found by using process expansion.

To put the job with job/process/group id 0 in the background.

bg %0

To run a command / script in background put & at the end

./script.sh &

sleep 10 &

To list all jobs

jobs

Fg

Bring a background process to the foreground

fg pid

nohup Most of the time your login into remote server via ssh. If you start a shell script or command andyou exit (abort remote connection), the process / command will get killed. Sometime job or command takes a long time. If you are not sure when the job will finish, then it is better to leavejob running in background. But, if you log out of the system, the job will be stopped and terminatedby your shell. What do you do to keep job running in the background when process gets SIGHUP? The answer is simple, use nohup command line-utility which allows to run command/process orshell script that can continue running in the background after you log out from a shell:

nohup command-name &

nohup /path/to/command-name arg1 arg2 &

Where,

command-name is name of shell script or command name. You can pass argument to command or a shell script.

& nohup does not automatically put the command it runs in the background; you must do that explicitly, by ending the command line with an & symbol.

Use jobs -l command to list all jobs running in background:

jobs -l

exit To fail a shell script programmatically

exit 0 Success

exit 1 General errors, Miscellaneous errors, such as "divide by zero" and other impermissible operations

exit 2 Misuse of shell builtins (according to Bash documentation)

1) **See all process crated by user:** ps -u username , top -U
2) **List Open Files for Process:**

   First use ps command to get PID of process, enter: ps -aef | grep {process-name}, ps -C firefox -o pid=

   Next pass this PID to pfiles command: pfile {PID}, ls -l /proc/7857/fd

lsof

1) **List User Specific Opened Files:** lsof -u tecmint
2) **Find Processes running on Specific Port:** lsof -i TCP:22
3) **List Open Files of TCP Port ranges 1-1024:** lsof -i TCP:1-1024
4) **Find Out who's Looking (which user) What Files and Commands:** lsof -i -u username
5) **List all Network Connections 'LISTENING & ESTABLISHED':** lsof -i
6) **List open files by pid:** lsof -p 351
7) **List processes using a particular file:** lsof filename (or) fuser filename

Permission and user related commands

chmod

chmod is used to change the permissions of files or directories.

	User	Group	Other
Read	4	4	4
Write	2	2	2
Execute	1	1	1
	U	G	O
	X	X	X

Chmods:		
777	=	rwxrwxrwx
755	=	rwxr-xr-x
644	=	rw-r--r--
700	=	rwx------
750	=	rwxr-x---

1. Deny execute permission to everyone. : **chmod a-x file.txt**
2. Allow read permission to everyone. : **chmod a+r file.txt**
3. Make a file readable and writable by the group and others. : **chmod go+rw file.txt**
4. Make a shell script executable by the user/owner. : **chmod u+x file.sh**
5. Allow everyone to read, write, and execute the file and turn on the set group-ID. : **chmod =rwx,g+s file.sh**
6. Recursively (-R) Change the permissions of the directory myfiles, and all folders and files it contains, to mode 755: User can read, write, and execute; group members and other users can read and execute, but cannot write. : **chmod -R 755 folder**
7. Set the permission of file.txt to "read and write by everyone." : **chmod 666 file.txt**

Chown

chown command is used to change the owner and group of a file or directory.

1) To change owner to oracle and group to dba on a file. i.e Change both owner and group at the same time.

chown oracle:dba dbora.sh

2) Use -R to change the ownership recursively.

chown -R oracle:dba /home/oracle

adduser / useradd

To add/create a new user, all you've to follow the command 'useradd' or 'adduser' with 'username'. The 'username' is a user login name, that is used by user to login into the system.

Only one user can be added and that username must be unique (different from other username already exists on the system).

For example, to add a new user called 'newuser', use the following command.
Passwd

useradd newuser

1) **Create a User with Different Home Directory**

   By default 'useradd' command creates a user's home directory under /home directory with username. Thus, for example, we've seen above the default home directory for the user 'newuser' is '/home/ newuser '.

   However, this action can be changed by using '-d' option along with the location of new home directory (i.e. /data/projects). For example, the following command will create a user 'username' with a home directory '/data/projects'.

   useradd -d /data/projects username

2) **Add a User to Multiple Groups**

   useradd -G admins,webadmin,developers username

- **-c comment**: Add a comment for the user
- **-d home-directory**: Create a new home-directory
- **-e yyyy-mm-dd**: Date for the account to be disabled
- **-f days**: Number of days after the password expires until the account is disabled. (If 0 is specified, the account is disabled immediately after the password expires. If -1 is specified, the account is not be disabled after the password expires.)
- **-g group-name**: Primary Group name or group number for the user's default group (group name must exist)
- **-G group-list**: List of secondary additional (other than default) group names or group numbers, separated by commas, of which the user is a member (group name must exist)
- **-m**: Create the home directory
- **-M**: Do not create the home directory
- **-s**: User's login shell (default /bin/bash)

passwd

1) change your password from command line using passwd. This will prompt for the old password followed by the new password.

   passwd

2) Super user can use passwd command to reset others password. This will not prompt for current password of the user.

   passwd USERNAME

3) Remove password for a specific user. Root user can disable password for a specific user. Once the password is disabled, the user can login without entering the password.

   passwd -d USERNAME

Linux list all users command

cat /etc/passwd

INFO COMMANDS

uname

uname command displays important information about the system such as — Kernel name, Host name, Kernel release number, Processor type, etc.,

To check the linux version

```
cat /etc/os-release (or) lsb_release -a
```

whereis

When you want to find out where a specific Unix command exists (for example, where does ls command exists?), you can execute the following command.

whatis

Whatis command displays a single line description about a command.

who

who command is used to find out the following information:

1. Time of last system boot
2. Current run level of the system
3. List of logged in users and more.

a) To display all details of current logged in user: **who -a**
b) To display current run level of the system: **who -r**
c) To show time of the system when it booted last time: **who -b -H**
d) To show list of users logged in to system: **who -u**

whoami

To display system's username

w

To display list of users and their activities

Archive (tarbal, zip, gzip)

1) **Create tar Archive File**

   tar -cvf sample.tar <source_location>c – Creates a new .tar archive file.
   v – Verbosely show the .tar file progress.f – File name type of the archive file.

2) **Untar tar Archive File**
   a. **Untar files in Current Directory**

      tar -xvf sample.tar <source_location>
   b. **Untar files in specified Directory**

      tar -xvf public_html-14-09-12.tar -C /home/public_html/videos/x-extract archive

3) **Create tar.gz Archive File (compressed)**

   tar -cvzf sample.tar.gz <source_location>

4) **Uncompress tar.gz Archive File**

   tar -xvf thumbnails-14-09-12.tar.gz

5) **To zip files**

   zip [options] zipfile files_list
   zip myfile.zip filename1.txt filename2.txt …..

6) **Extracting files from zip file**

   unzip myfile.zip

Linux boot process and Run levels

The stages involved in Linux Booting Process are:

1) BIOS
2) Boot Loader
    a. MBR
    b. GRUB
3) Kernel
4) Init
5) Runlevel scripts
6) User Interface

STEP 1: BIOS

BIOS determine all the list of bootable devices available in the system. Prompts to select bootable device which might be Hard Disk, CD/DVD-ROM, Floppy Drive, USB Flash Memory Stick etc. Operating System tries to boot from Hard Disk where the MBR contains primary boot loader.

STEP 2: Boot Loader

To be very brief this phase includes loading of the boot loader (MBR and GRUB/LILO) into memory to bring up the kernel.

MBR (Master Boot Record)

It is the first sector of the Hard Disk with a size of 512 bytes. The first 434 - 446 bytes are the primary boot loader, 64 bytes for partition table and 6 bytes for MBR validation timestamp.

NOTE: Now MBR directly cannot load the kernel as it is unaware of the file system concept and requires a boot loader with file system driver for each supported file systems, so that they can be understood and accessed by the boot loader itself. To overcome this situation GRUB is used with the details of the file system in /boot/grub.conf and file system drivers

GRUB (Grand unified Boot Loader)

This loads the kernel in 3 stages.

GRUB Stage 1:

The primary boot loader takes up less than 512 bytes of disk space in the MBR - too small a space to contain the instructions necessary to load a complex operating system. Instead the primary boot loader performs the function of loading either the stage 1.5 or stage 2 boot loader.

GRUB Stage 1.5:

Stage 1 can load the stage 2 directly, but it is normally set up to load the stage 1.5. This can happen when the /boot partition is situated beyond the 1024-cylinder head of the hard drive. GRUB Stage 1.5 is located in the first 30 KB of Hard Disk immediately after MBR and before the first partition. This space is utilized to store file system drivers and modules. This enabled stage 1.5 to load stage 2 to load from any known location on the file system i.e. /boot/grub

GRUB Stage 2:

This is responsible for loading kernel from /boot/grub/grub.conf and any other modules needed and Loads a GUI interface i.e. splash image located at /grub/splash.xpm.gz with list of available kernels where you can manually select the kernel or else after the default timeout value the selected kernel will boot The original file is /etc/grub.conf of which you can observe a symlink file at /boot/grub/grub.conf

STEP 3: Kernel

This is the heart of operating system responsible for handling all system processes. Kernel is loaded in the following stages: Kernel as soon as it is loaded configures hardware and memory allocated to the system. Next it uncompresses the initrd image (compressed using zlib into zImage or bzImage formats) and mounts it and loads all the necessary drivers. Loading and unloading of kernel modules is done with the help of programs like insmod, and rmmod present in the initrd image. Looks out for hard disk types be it a LVM or RAID. Unmounts initrd image and frees up all the memory occupied by the disk image. Then kernel mounts the root partition as specified in grub.conf as read-only. Next it runs the init process

STEP 4: Init Process

Executes the system to boot into the run level as specified in /etc/inittab You can check current runlevel details of your system using below command on the terminal Next as per the fstab entry file system's integrity is checked and root partition is re-mounted as read-write (earlier it was mounted as read-only).

STEP 5: Runlevel Scripts.

Depending on your default init level setting, the system will execute the programs from one of the following directories.

- Run level 0 (**init0**) – /etc/rc.d/rc0.d/ **System halt (Shutdown)** - no activity, the system can be safely powered down.
- Run level 1 (**init1**) – /etc/rc.d/rc1.d/ **Single user** - rarely used.
- Run level 2 (**init2**) – /etc/rc.d/rc2.d/ **Multiple users** - no NFS (network filesystem); also used rarely.
- Run level 3 (**init3**) – /etc/rc.d/rc3.d/ **Multiple users** - command line (i.e., all-text mode) interface; the standard runlevel for most Linux-based server hardware.
- Run level 4 (**init4**)– /etc/rc.d/rc4.d/ **User-definable**
- Run level 5 (**init5**) – /etc/rc.d/rc5.d/ **Multiple users -** GUI (graphical user interface); the standard runlevel for most Linux-based desktop systems.
- Run level 6 (**init6**) – /etc/rc.d/rc6.d/ **Reboot** - used when restarting the system.

Security and Firewall

Iptables and firewall

Iptables is a Linux command line firewall that allows system administrators to manage incoming and outgoing traffic via a set of configurable table rules.
There are currently 3 types of tables:

1) **FILTER** – this is the default table, which contains the built-in chains for:
   - INPUT – packages destined for local sockets
   - FORWARD – packets routed through the system
   - OUTPUT – packets generated locally

2) **NAT** – a table that is consulted when a packet tries to create a new connection. It has the following built-in:
   - PREROUTING – used for altering a packet as soon as it's received
   - OUTPUT – used for altering locally generated packets
   - POSTROUTING – used for altering packets as they are about to go out

3) **MANGLE** – this table is used for packet altering. Until kernel version 2.4 this table had only two chains, but they are now 5:
   - PREROUTING – for altering incoming connections
   - OUTPUT – for altering locally generated packets
   - INPUT – for incoming packets
   - POSTROUTING – for altering packets as they are about to go out
   - FORWARD – for packets routed through the box

1) Start/Stop/Restart Iptables Firewall

Sudo systemctl start iptables
Sudo systemctl stop iptables
Sudo systemctl restart iptables

2) Check all IPtables Firewall Rules

iptables -L -n -v

3) Block Specific IP Address in IPtables Firewall

iptables -A INPUT -s xxx.xxx.xxx.xxx -j DROP

4) Unblock IP Address in IPtables Firewall

iptables -D INPUT -s xxx.xxx.xxx.xxx -j DROP

5) Block Specific Port on IPtables Firewall

iptables -A OUTPUT -p tcp --dport xxx -j DROP