Linux

Index

- **File ownerships, permissions**
    - Change mode
    - Change ownerships
- **Control Operators in Linux**
    - Ampersand(&)
    - Semicolon(;)
    - AND(&&)
    - OR(||)
    - Concatenation\
- **Regular Expressions**
- **Filters**
    - grep
    - egrep
    - cut
    - tr
    - wc
    - sort
    - uniq
    - comm
- **Find Utility**
- **sed**
- **awk**
- **Crontab**
- **Hard Link and Soft link**
- **SSH connection**
- **Shell script**

➢ What is Linux

Linux is a free open source operating system (OS) based on UNIX that was created in 1991 by Linus Torvalds.

Linux is a UNIX Clone

Why use Linux

High security: Most users do not log in as the root; hence, they cannot do much damage to the system, except to their own files and programs, since the downloaded file/malware will have limited privileges

High stability: The Linux system is very stable and is not prone to crashes. The Linux OS runs exactly as fast as it did when first installed, even after several years

Free: Linux is completely free and users do not need to pay for anything. All the basic software required by a typical user and even an advanced user are available

Open source: The most important aspect of Linux is that its source code is available as it falls under the FOSS category (Free and Open Source Software)

Linux Distributions

A Linux distribution is an operating system made from a software collection

- Fedaora
  - -Redhat (RHEL)
  - -CentOS
  - -Oracle Linux
- Debian
  - Ubuntu
  - Linux Mint
- SuSE
- Gentoo
- Arch

➢ File Hierarchy

/ : Root directory.

/bin : Command-line executable directory.

/dev : Device directory, not actually located on the hard disk

/etc : System configuration files and executable directory.

/lib : The library directory.

/home : Consists of the users' home directories

/usr : Unix System Resources directory. should contain shareable, read only data

/var : Files that are unpredictable in size, such as log, cache

/opt : store optional software

/tmp : Applications and users should use /tmp to store temporary data when needed

➢ Package managers

What they do?

- install package
- remove package
- update packages

1.Debian Package Manager

1.1 dpkg

- install software
- Upgrade software
- Remove software
- Low level too

Example

a. dpkg --install foo_VVV-RRR.deb (Install a package)
b. dpkg --remove foo. (Remove a package (but not its configuration files)
c. dpkg --purge foo (Remove a package (including its configuration files)
d. dpkg --list 'foo*' (List the installation status of packages containing the string foo*)

1.2 apt

Advanced packaging tool

High level tool

Handle all package dependency automatically

Example

a. apt update (To update the list of packages known by your system)
b. apt install foo (To install the foo package and all its dependencies)
c. apt remove foo (To remove the foo package from your system)
d. apt purge foo (To remove the foo package and its configuration files from your system)
e. apt list --upgradable (To list all packages for which newer versions are available)
f. apt list –installed (To list installed packages)
g. apt upgrade (To upgrade all the packages on your system)
h. apt full-upgrade (To upgrade all the packages on your system, and, if needed for a package upgrade, installing extra packages or removing packages)

2. Redhat package manager

2.1 rpm

- Low level tool
- Install package
- Updates package
- Remove package

Example

a. rpm -ivh {rpm-file} Install the package
b. rpm -Uvh {rpm-file} Upgrade package
c. rpm -ev {package} Erase/remove/ an installed package
d. rpm -qa Display list all installed packages

2.2 yum

- Yellowdog Updates, modified
- Manage dependency
- Utilizes online repositories

Example

a. yum install {package_name} Install package_name
b. yum remove {package_name} Remove package_name
c. yum autoremove {package_name} Remove package_name and dependent packages
d. yum list available List all available packages
e. yum list installed List all installed packages

➢ Basic Commands

username@hostname$
Or:
root@hostname #

#→ root user

$ →normal user

#env Environment variable

#history

 # whoami Displays the names of all the users who have currently logged in

 # date

 # cal

 # cal 2018

 # cal may 2018

 # clear

 # who
Displays the names of all the users who have currently logged in

 # id
Displays your UID, Primary GID, and Secondary GID's

 # uname
The uname utility prints information about the current system on the standard output.

 # whereis
Displays the path/ location of a command

 # sleep 5
command is used to delay for a specified amount of time.

 # man mkdir
Type man followed by a command (for which you want help) and start reading

➢ Working with directories

pwd (Print Working Directory)

pwd

cd

change your current directory with the cd command

cd

cd ~

cd is also a shortcut to get back into your home directory

#cd ..

go to the parent directory

#cd ../../../

#cd –

to go to the previous directory

Absolute and Relative paths

- You should be aware of absolute and relative paths in the file tree.
- When you type a pathstarting with a slash (/), then the root of the file tree is assumed.
- If you don't start your pathwith a slash, then the current directory is the assumed starting point.

#cd /home/user/doc/mydir

#cd doc/mydir

#mkdir

#mkdir –p

#mkdir –p dir1/dir2/dir3/dir4

#rmdir

Listing files

ls –l

Explanation of ls -l

```
[root@localhost mohit]# ls -l
total 16
drwxrwxr-x. 2 mohit mohit 4096 Jun  5 17:19 max
drwxr-xr-x. 2 root  root  4096 Jun  5 17:21 nat
-rw-r--r--. 1 root  root    16 Jun  5 17:20 rr
-rw-rw-r--. 1 mohit mohit   33 Jun  5 17:19 tri
```

1 2 3 4 5 6 7

Explanation of ls -l

1 = file type and permissions (filetype, user permissions, group permissions, other permissions)
2 = contents of dir/file (links)
3 = user owner
4 = group name
5 = Size
6 = Date
7 = Name of dir/file

Letter	Permission	Value
r	read	4
w	write	2
x	Execute	1

Other options with ls

ls -a list all files including hidden file starting with '.'
ls -d list directories - with ' */'
ls -i list file's inode index number
ls -l list with long format - show permissions
ls -la list long format including hidden files
ls -lh list long format with readable file size
ls -ls list with long format with file size
ls -r list in reverse order
ls -R list recursively directory tree
ls -s list file size
ls -S sort by file size
ls -t sort by time & date

➢ Playing with files

Files on Linux (or any Unix) are **case sensitive**.
This means that **FILE1**is different from**file1** and **/etc/hosts**is different from **/etc/Hosts**
Linux computer)
In Linux**everything is a file**
A directory is a special kind of file

Checking file type

file myimage.jpg
file hello.java
file /etc/passwd

touch
create an empty file is with touch

 # touch myfile

Remove file
#rm
#rm –i
prevent yourself from accidentally removing a file
#rm –rf
#rm -r will not remove non-empty directories. Use rm –rf for forcefully

File Related
Copying a file cp
Moving a file mv
Renaming a file mv
Removing a file rm
Displaying a file cat

Playing with file contents

1. head

- head is used to display the **first parts of a file**, it outputs the first 10 lines by default.
- You can use the –n num flag to specify the number of lines to be displayed:

```
# head myfile
# head -4 myfile
```

2. tail

- tail outputs the **last parts** (10 lines by default) of a file.
- Use the -n num switch to specify the number of lines to be displayed.

```
# tail myfile
# tail -5 myfile
```

3. cat

cat for display # cat myfile
cat for concatenate # cat myfile > newfile

cat creating new file
cat > newfile.txt
My file
How are you
Ctrl+d

Then type one or more lines, finishing each line with the enter key.
Afterthe last line, press ctrl d to end file

4. tac

- opposite to cat for display content (display the content from bottom to top)

```
# tac myfile
```

5. more

- Useful for displaying files that take up more than one screen.

- Output will scroll off your screen

more myfile
cat myfile | more
It shows file content in a page like format, where users can press [Enter] to view more information.

6. Less

- 'less' command is same as 'more' command but include some more features.
- It automatically adjust with the width and height of the teminal window
- It opens the output in a separate window in an uncluttered way

less filename
Enter q to exit

7. Tee

The following command writes the output of “ls” only to the file and not to the screen.
$ ls > file

The following command (with the help of tee command) writes the output both to the screen (stdout) and to the file.
$ ls | tee file

➢ System Commands

Memory details

 # free –m
Getting RAM details

 # df
Hard disk details
-h human readable
-m size in mb

#du
Directory usage

 # vmstat –s
Memory usage statistics

Process related commands

Processes are the running instance of a program. Several processes run on a computer, and each process is assigned a unique identification number called a process ID (PID)

 # ps
Display processes that are running on the current terminal (TTY).

The ps command is usually used with a set of parameters.

In order to show more columns consisting of more information, use -f (stands for full)
#ps -f

 # ps –e
(e stands for every)
Display every active process on a Linux system

 # ps –ef

perform a full-format listing, add the -f

#ps –ef | grep 'java'

#top
Cpu and memory utilization by each process

KILL

KILL is a signal used to terminate
a process. The events such as Ctrl + C, and Ctrl + Z also send two types of signals. The kill command is used to send signals to processes

1.In order to list all the signals available, use:
$ **kill -l**
It will print signal numbers and corresponding signal names.

2. Terminate a process as follows:
$ **kill PROCESS_ID_LIST**
The kill command issues a TERM signal by default. The process ID list is to be specified with space as a delimiter between process IDs.

3. We frequently use force kill for processes. In order to force kill a process, use:
$ **kill -s SIGKILL PROCESS_ID**
Or:
$ **kill -9 PROCESS_ID**

Let's walk through the additional commands used for terminating and signalling processes.
The kill family of commands
The kill command takes the process ID as the argument. There are also a few other commands in the kill family that accept the command name as the argument and send a signal to the process.

The killall command terminates the process by name as follows:
$ **killall process_name**

In order to send a signal to a process by name, use:
$ **killall -s SIGNAL process_name**

In order to force kill a process by name, use:
$ **killall -9 process_name**

For example:

$ **killall -9 gedit**
#kill <pid>
kill 9 <pid> forcefully
#pkill <pname>
Pkill java

Network Related

 # hostname short name
hostname –f fully qualified hostname
#hostname –i get ip address
ip a

#ifconfig
ifconfig is the command that is used to configure and display details about network interfaces, subnet mask, and so on. On a typical system, it should be available at /sbin/ifconfig.

#ping google.com
#ping –c 7 <ip> Ping with -c option exit after N number of request (success or error respond).

 # traceroute google.com

traceroute is a network troubleshooting utility which shows number of hops taken to reach destination also determine packets traveling path

#netstat -r
Netstat (Network Statistic) command display connection info, routing table information etc. To displays routing table information use option as -r.

 # dig www.google.com

Dig (domain information groper) query DNS related information like A Record, CNAME, MX Record etc. This command mainly use to troubleshoot DNS related query.

#nslookup www.google.com
nslookup command also use to find out DNS related query

CPU Related

 # cat /proc/cpuinfo

```
$ cat /proc/cpuinfo | grep 'vendor' | uniq          #view vendor name
$ cat /proc/cpuinfo | grep 'model name' | uniq      #display model name
$ cat /proc/cpuinfo | grep processor | wc -l        #count the number of processing units
$ cat /proc/cpuinfo | grep 'core id'                #show individual cores
```

#lscpu
Shows CPU Architecture Info

➢ User Management

User

1. /etc/passwd – User account information.
2. /etc/shadow – Secure account information.
3. /etc/group – Group account information.

#cat /etc/passwd

#useradd or adduser ajit

#userdel -r <userName>
you can delete home directory along with user account.

#passwd ajit

#cat /etc/shadow : files are the encrypted user passwords

Groups

#groupadd <groupName>

#cat /etc/group

#groupmod -n <oldGroup> <newGroup>

#usermod -a -G <group> <userName>
#id <userName> (Check the group is added)

#groupdel <group>

➢ File ownerships, permissions

listing all local user accounts -
cut -d: -f1 /etc/passwd | column

Change mode

Permissions of a file can be listed by using the ls -l command:

-rw-r--r-- 1 ajit root 2497 2010-02-28 11:22 myfile.py

The first column of the output specifies the following, with the first letter corresponding to:

- – if it is a regular file
d – if it is a directory
c – for a character device
b – for a block device
l – if it is a symbolic link
s – for a socket
p – for a pipe

User: is the owner of the file.

Group: is the collection of users (as defined by the system administrator) that are permitted some access to the file.

Others: are any entities other than the user or group owner of the file.

chmod u=rwx filename

chmod o+x filename

Use + to add permission to a user, group, or others and use - to remove the permissions

chmod a+x filename
Here a means all. Add permissions to all users groups and others

chmod a-x filename
In order to remove a permission, use -

Read, write, and execute permissions have unique octal numbers as follows:
r-- = 4
-w- = 2
--x = 1

chmod 764 filename
Applying permissions recursively to files

chmod 777 . –R
Sometimes it may be required to recursively change the permissions of all the files and directories inside the current directory
The -R option specifies to apply change to a permission recursively

Changing ownership

chown: change owner

chown <newOwner> <fileName> ch

chown user.group filename
chown user.group . -R
chgrp: change group

#chgrp <newGroup> <fileName>

#umask

While creating a file or directory, by default a set of permissions are applied. These default permissions are viewed by umask command.

➢ Control Operators in Linux

Ampersand (&)

The function of '&' is to make the command run in background. Just type the command followed with a white space and '&'. You can execute more than one command in the background, in a single go
semi-colon (;)

ping www.google.com &

semi-colon (;)

it makes possible to run, several commands in a single go and the execution of command occurs sequentially.

mkdir mydir ; touch file;ls

AND(&&)
The AND Operator (&&) would execute the second command only, if the execution of first command SUCCEEDS

mkdir mydir && cd mydir

OR (||)
The OR Operator (||) is much like an 'else' statement in programming. The above operator allow you to execute second command only if the execution of first command fails

#mkdir mydir || cd mydir

Concatenation Operator (\)
The Concatenation Operator (\) as the name specifies, is used to concatenate large commands over several lines in the shell

vi myf\
Ile.txt

combining && and ||
You can use this logical AND and logical OR to write an if-then-else structure on the command line.
#rm file1 && echo It worked! || echo It failed!

#rm file1 && echo It worked! || echo It failed!

- "A ; B" Run A and then B, regardless of success of A
- "A && B" Run B if A succeeded
- "A || B" Run B if A failed
- "A &" Run A in background

pound sign
Everything written after a pound sign (#) is ignored by the shell.This is useful to write useful comment lines
mkdir test # we create a directory
cd test #### we enter the directory
ls # is it empty ?

\ escaping special characters

Pipe(|)

- A pipe is a form of redirection (transfer of standard output to some other destination) that is used in Linux
- Direct connection between commands/ programs/ processes allows them to operate simultaneously and permits data to be transferred between them continuously rather than having to pass it through temporary text files or through the display screen. Pipes are unidirectional i.e data flow from left to right through the pipeline.

command_1 | command_2 | command_3 | .... | command_N

➢ Regular Expressions

regex	Description	Example
^	This specifies the start of the line marker.	^tux matches a line that starts with tux.
$	This specifies the end of the line marker.	tux$ matches a line that ends with tux.
.	This matches any one character.	Hack. matches Hack1, Hacki, but not Hack12 or Hackil; only one additional character matches.
[]	This matches any one of the characters enclosed in [chars].	coo[kl] matches cook or cool.
[^]	This matches any one of the characters except those that are enclosed in [^chars].	9[^01] matches 92 and 93, but not 91 and 90.
[-]	This matches any character within the range specified in [].	[1-5] matches any digits from 1 to 5.
?	This means that the preceding item must match one or zero times.	colou?r matches color or colour, but not colouur.
+	This means that the preceding item must match one or more times.	Rollno-9+ matches Rollno-99 and Rollno-9, but not Rollno-.
*	This means that the preceding item must match zero or more times.	co*l matches cl, col, and coool.
()	This treats the terms enclosed as one entity	ma(tri)?x matches max or matrix.
{n}	This means that the preceding item must match n times.	[0-9]{3} matches any three-digit number. [0-9]{3} can be expanded as [0-9][0-9][0-9].
{n,}	This specifies the minimum number of times the preceding item should match.	[0-9]{2,} matches any number that is two digits or longer.
{n, m}	This specifies the minimum and maximum number of times the preceding item should match.	[0-9]{2,5} matches any number that has two digits to five digits.
|	This specifies the alternation—one of the items on either of side of | should match.	Oct (1st | 2nd) matches Oct 1st or Oct 2nd.
\	This is the escape character for escaping any of the special characters mentioned previously.	a\.b matches a.b, but not ajb. It ignores the special meaning of . because of \.

➢ Filters

grep

Global Regular Expressions Print
filter lines of text containing (or not containing) a certain string

vi myfile.txt

THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.
this line is the 1st lower case line in this file.
This Line Has All Its First Character Of The Word With Upper Case.

Two lines above this line is empty.
And this is the last line.

grep ajit myfile.txt prints line matching ajit

grep –i Ajit myfile.txt search by case insensitive way

grep –v sanju myfile.txt outputs lines not matching the string.

grep "CA*" myfile.txt Word starts with CA

grep -n "go" myfile.txt Show line number while displaying the output

egrep

Extended Global Regular Expressions Print
egrep is extended version of grep or egrep is equal to grep –E

search for lines which contains exactly two consecutive p characters
egrep p{2} *
OR
#grep pp *
OR
#grep -E p{2} *
egrep "S$|A$" * all lines which ends with "S" OR "A"

cut

new file

state.txt

Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Maharashtra
Chhattisgarh

cut state.txt

cut: you must specify a list of bytes, characters, or fields

Try 'cut --help' for more information.

List without ranges
cut -b 1,2,3 state.txt

List with ranges
cut -b 1-3,5-7 state.txt

cut -b 1- state.txt
In this, 1- indicate from 1st byte to end byte of a line

cut -b -3 state.txt
In this, -3 indicate from 1st byte to 3rd byte of a line

select columns from files, depending on a delimiter
cut -d "delimiter" -f (field number) file.txt

cut -d: -f1,3 /etc/passwd
cut -d" " -f1 tennis.txt space as the delimiter
cut -c2-7 /etc/passwd display the second to the seventh character

tr
translate characters with tr

cat tennis.txt | tr 'e' 'E'

AmEliE MaurEsmo, Fra
Kim ClijstErs, BEL
JustinE HEnin, BEl
SErEna Williams, usa
VEnus Williams,

cat tennis.txt | tr 'a-z' 'A-Z'

wc
Counting words, lines and characters

wc tennis.txt

counting the number of lines
wc –l tennis.txt

counting the number of words
wc –w tennis.txt

counting the number of characters
wc –c myfile.txt

sort
filter will default to an alphabetical sort

cat music.txt
Queen
Brel
Led Zeppelin
Abba

sort music.txt
Abba

Brel
Led Zeppelin
Queen

#sort -k1 country.txt ==> sort on column 1

Belgium, Brussels, 10
France, Paris, 60
Germany, Berlin, 100
Iran, Teheran, 70
Italy, Rome, 50

sort -k2 country.txt ==> sort on column 2
sort -k3 country.txt ==> alphabetical sort
sort -n -k3 country.txt ==> numerical sort

uniq

uniq you can remove duplicates from a sorted list.

cat music.txt
Queen
Brel
Queen
Abba

sort music.txt
Abba
Brel
Queen
Queen

sort music.txt |uniq
Abba
Brel
Queen

comm

Comparing streams (or files) can be done with the comm. By default comm will output three columns.

In this example, Abba, Cure and Queen are in both lists, Bowie and Sweet are only in the first file, Turner is only in the second.

```
# cat > list1.txt
Abba
Bowie
Cure
Queen
Sweet

# cat > list2.txt
Abba
Cure
Queen
Turner

# comm list1.txt list2.txt
        Abba
Bowie
        Cure
        Queen
Sweet
Turner
```

➢ Find Utility

 # find . -print
Print lists of files and folders

Search based on filename or regular expression match
find /var/lib -name '*.txt' Find text files in /var/lib directory
find . -iname "Filename*" ignore case

If we want to match either of the multiple criteria, we can use OR conditions as shown in the following:

find . \(-name "*.jar" -o -name "*.war" \)
find . ! -name "*.txt" –print

find can also exclude things that match a pattern using !:
This will match all the files whose names do not end in .txt. The following example shows the result of the command:

Search based on the directory depth

find . -maxdepth 2 -name "f*" -print
traverses up to at most two descending levels of subdirectories.

Search based on file type
find . -type f -empty Find an empty file within the current directory.
#find . -name "te*.txt" Find a file called testfile.txt in current and sub-directories.
find . -type d -print List only directories including descendants
find . -type f -print List only regular files
find . -type l -print List only symbolic links

Search on file times

Access time (-atime): It is the last timestamp of when the file was accessed by a user

Modification time (-mtime): It is the last timestamp of when the file content was modified

Change time (-ctime): It is the last timestamp of when the metadata for a file (such as permissions or ownership) was modified

Print all the files that were accessed within the last seven days as follows:
 # find . -type f -atime -7 -print

Print all the files that are having access time exactly seven-days old as follows:
#find . -type f -atime 7 -print

Print all the files that have an access time older than seven days as follows:
$ find . -type f -atime +7 -print

There are some other time-based parameters that use the time metric in minutes. These are as follows:
-amin (access time)
-mmin (modification time)
-cmin (change time)

To print all the files that have an access time older than seven minutes, use the following command:
 # find . -type f -amin +7 -print

find all the files that have a modification time greater than that of the modification time of a given file.txt file as follows:
find . -type f -newer file.txt -print

Search based on file size
Based on the file sizes of the files, a search can be performed as follows:

 # find . -type f -size +2k
Files having size greater than 2 kilobytes

M: Megabyte
G: Gigabyte

Deleting based on the file matches
The -delete flag can be used to remove files that are matched by find.
Remove all the .swp files from the current directory as follows:
#find . -type f -name "*.swp" -delete

Match based on the file permissions and ownership
It is possible to match files based on the file permissions. We can list out the files having specified file permissions as follows:
find . -type f -perm 644 -print

➢ Sed

- sed stands for stream editor.
- Text replacement
- It is a very essential tool for text processing, and a marvelous
- Utility to play around with regular expressions

sed [options] filename

vi myfile
ajay manager account 45000
sunil clerk account 25000
varun manager sales 50000
amit manager account 47000
tarun peon sales 15000
deepak clerk sales 23000
sunil peon sales 13000
satvik director purchase 80000

sed -n '1p' myfile prints line 1
sed -n '1,4p' myfile prints line 1,2,3,4
sed -n '1~2p' myfile prints line 1,3,5,7
sed '2d' myfile Delete line 2

sed 's/pattern/replace_string/' file
cat file | sed 's/pattern/replace_string/'

By default, sed only prints the substituted text. To save the changes along with the substitutions to the same file, use the -i option

sed -i 's/text/replace/' file
q

These usages of the sed command will replace the first occurrence of the pattern in each line. If we want to replace every occurrence, we need to add the g parameter at the end, as follows:
$ sed 's/pattern/replace_string/g' file

we sometimes need to replace only the Nth occurrence onwards. For this, we can use the /Ng form

of the option.
echo thisthisthisthis | sed 's/this/THIS/2g'
thisTHISTHISTHIS

Removing blank lines

sed '/^$/d' file
Blanks can be matched with regular expression ^$
/pattern/d will remove lines matching the pattern

➢ Awk

- Advanced text processing
- it can operate on columns and rows.
- It supports many built-in functionalities, such as arrays and functions, in the C programming language

WHAT CAN WE DO WITH AWK ?

1. AWK Operations:

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

2. Useful For:

(a) Transform data files
(b) Produce formatted reports

3. Programming Constructs:

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

$0 all fields
$1 cloumn 1
$2 column 2
$3,$4.. Similarly specific fields

NR: It stands for the current record number, which corresponds to the current line number when it uses lines as records

NF: It stands for the number of fields, and corresponds to the number of fields in the current record under execution (fields are delimited by space)
vi myfile

ajay manager account 45000
sunil clerk account 25000
varun manager sales 50000
amit manager account 47000
tarun peon sales 15000
deepak clerk sales 23000
sunil peon sales 13000
satvik director purchase 80000

awk options 'selection _criteria {action }' input-file > output-file

```
# awk  '{ print $0} '   myfile              Prints all content

# awk  '{ print $1,$3}'  myfile            Prints column 1,3 content

# awk  '{ print NR,$0} '   myfile

#awk '{print NF,$0}'       myfile

# awk '/manager/ {print}' myfile                  Print the lines which matches manager

# awk 'length($0) > 7'  Printing lines with more than 7 characters
#  awk '{ if($3>25000) print $0;}'
```

➢ Crontab

- The crontab is used for running specific tasks on a regular interval.
- Linux crontab is similar to windows task schedules.
- Crontab is very useful for routine tasks like scheduling system scanning, daily backups etc. Crontab executes jobs automatically in the backend on a specified time and interval.

Syntax

* * * * * COMMAND or /path/to/script.sh

MIN HOUR DOM MON DOW CMD

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

View Current Logged-In User's Crontab entries

crontab –l

Edit the crontab
#crontab -e

setup a crontab like below
10 5 * * * echo "Hello World from Crontab" >> /home/ubuntu/cronfile.txt

check the date using the date variable - it might be in UTC or ETC

Schedule a cron to execute at 2am daily.
0 2 * * * /bin/sh backup.sh

Schedule a cron to execute twice a day at 5 AM and 5 PM daily.

0 5,17 * * * /scripts/script.sh

Schedule a cron to execute on every Sunday at 5 PM.
0 17 * * sun /scripts/script.sh

Schedule a cron to execute on every 10 minutes.
*/10 * * * * /scripts/monitor.sh

Schedule a cron to execute on selected months January, May and August.
* * * jan,may,aug * /script/script.sh

Schedule a cron to execute on selected days on each Sunday and Friday at 5 PM..
0 17 * * sun,fri /script/script.sh

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

@hourly /scripts/script.sh

Removing cron

crontab -r

➢ Hard link and Soft Link

Links are pointers pointing to a file or a directory.
Creating links is a kind of shortcuts to access a file.
Links allow more than one file name to refer to the same file, elsewhere

There are two types of links :

1. Soft Link or Symbolic links
2. Hard Links

These links behave differently when the source of the link (what is being linked to) is moved or removed.

Hard link

```
# mkdir –p dir1/dir2/dir3/dir4  &&  cd dir1/dir2/dir3/dir4
#vi test_file.txt
#ln test_file.txt  ~/myhlink
#ls –li
# cd
# ls –li
#cd –
# vi test_file.txt
Added some line
# cd
# cat myhlink
#cd –
# rm –rf  test_file.txt
```

- Each hard linked file is assigned the same Inode value as the original
- Hard links more flexible and remain linked even if the original or linked files are moved throughout the file system,
- Links have actual file contents
- We cannot create a hard link for a directory to avoid recursive loops.
- ls -l command shows all the links with the link column shows number of links.
- if original file is removed then the link will still show the content of the file

Soft link

#vi test_file.txt
#ln –s test_file.txt ~/stest.txt
#ls –li
cd
ls –li
#cd –
vi test_file.txt
Added some line
cd
cat stest.txt
#cd –
rm –rf test_file.txt

- A soft link is similar to the file shortcut feature which is used in Windows Operating systems
- Each soft linked file contains a separate Inode value that points to the original file.
- As similar to hard links, any changes to the data in either file is reflected in the other
- soft links can be linked across different file systems, although if the original file is deleted or moved, the soft linked file will not work correctly (called hanging link).
- ls -l command shows all links with first column value 1? and the link points to original file.
- Soft Link contains the path for original file and not the contents.
- A soft link can link to a directory.
- Removing soft link doesn't affect anything but removing original file, the link becomes "dangling" link which points to nonexistent file.

➢ SSH (Secure Shell)

- Key based authentication
- SSH protocol recommended a method for remote login and remote file transfer.
- Secure shell, is an encrypted protocol used to communicate with servers
- The SSH depends upon the use of public key cryptography.
- The OpenSSH server offers this kind of setup under Linux or Unix-like system.

Uses

- Automated Login using the shell scripts
- Making backups
- Run commands from the shell prompt and more
- Login without password

```
Server1#  ssh-keygen
Server1#  cd ~/.ssh

Server2# passwd root
Server2# vi /etc/ssh/sshd_config

        PermitRootLogin        yes
PasswordAuthentication  yes
Server2# service ssh restart

Server1# ssh-copy-id root@<Server2 ip>
Server1# ssh root@<Server2ip>
```