**To get the ipaddress** 🡺

ifconfig 🡺 to get the ip address

**Linux file system** 🡺

/boot 🡺 contains file that is used by the boot loader (grub.cfg)

/root 🡺 root user home directory. it is not same as /

/dev 🡺 system devices (ex: disk,cdrom,speaer,flashdrive,keyboards etc>

/etc 🡺 configuration files.

/bin 🡺 everyday user commands . in new version OS you would get /usr/bin

/sbin 🡺 system file system commands

/opt 🡺 optional add on applications (not part of OS apps)

/proc 🡺 running process (only exists in memory) . when you restart it will be empty.

/lib 🡺 c programming library files needed by commands and apps

/tmp 🡺 directory for temporary files

/home 🡺 directory for regular user

/var 🡺 system logs

/run 🡺 system daemons that start very early to store temporary run time files like PID files.

/mt 🡺 to mount external filesyetm (ex: NFS)

**What is ROOT 🡺**

There are 3 types of root on Linux system

root account 🡺 root is an account or a username on Linux machine and it is the most powerful account which has access to all the commands and files.

root as / 🡺 the very first file directory in Linux is also referred as root directory

root home directory 🡺 the root user account also has a directory located in /root which is called root home directory.

**Difference between find & locate 🡺**

* locate uses a prebuilt database, which should be regularly updated , while find iterates over a file system to locate files.
* Thus, locate is much faster than find, but can be inaccurate if the database is not updated
* to update locate database run updated

**Changing password 🡺**

passwd <userid> 🡺 it will ask old password then new password/renter new password

passwd root 🡺 it will change the password for root

**Wildcard 🡺**

* A wildcard is a character that can be used as a substitute for any of a class of characters in a search.

\*🡺 represents zero or more charecters

? 🡺 represents a single char

[ ] 🡺 a range of chars

**create files starting from a to z like a.txt, b.txt,c.txt**

touch {a..z).txt 🡺 it will create .txt files starting with a to z

**Soft link & Hard link 🡺**

* you cannot create soft or hard link within the same directory with the same name.
* Inode 🡺 pointer or number of a file on the hard disk

**ls**

**ls** 🡺list out all object directory, file,link

**ls -1** 🡺display object in one single column

**ls -x 🡺** multi column output

**ls -a 🡺** display objects along with hidden objects . and ..

**ls -A** 🡺display all objects without . and ..

**ls -r** 🡺display in reverse order.

**ls -ar** 🡺display all objects with reverse mode

**ls -l** 🡺display in long listing format

**ls -F** 🡺symbolic format

**ls -R**  🡺 Recursive mode. Display all object along with child object

**ls -d**  🡺displays . always use arguments. without using argumnet shows only (.)

**ls -t**  🡺 displays objects with modification of time ad recently modified time in the top.

**ls -ltu**  🡺display objects respected to used time,accessed time

**ls -S**  🡺size in decreasing order

**ls -n**  🡺same as l as group id

**ls -m**  🡺displays objects separated by (,)

**ls -of**  🡺open files

**/ directory**

**@ link**

**\* executable file**

**long listing format**

* column 1: 10 fields **-rwx\_wx\_\_x** : type of object and permission
* column 2: link
* column 3:username
* column 4:group name
* column 5:size
* column 6: date and time
* column 7: object name

-rwxrwxrwx

-file

d directory

l link

s socket

c charachter device

b block device

**permission**

2 types:

* absolute
* relative

**Absolute permission: represent the permission in octal format**

r 4

w 2

x 1

directory default value: 755: drwx r\_x r\_x

file default permission: 644: -rw\_r\_\_r\_\_

link default value: 777: @rwxrwxrwx

umask used to set the default permission for file and directory

umask default value : 022

777-022=755

**Relative permission:**

u: user

g: group

o: other

**+ give permission**

**- revoke permission**

**chmod u+x a.txt**

**chmod ugo+x**

**chmod u+x,g+r,o+r**

ugo=a

**chmod a+x**

**chmod +x by default a**

**chmod u-x,g+w,o+r a.txt**

**chmod ug+rwx,o+w**

**chmod u-rw,g+rw,o+x**

chmod -R u+x /raj give permission to all files of raj

mkdir -m 755 /raj create directory with specified permission

**filter |**

**wc a.txt** 🡺 number of lines, words, charcters

filter acts as inter process communication

**cat a.txt|wc -w**

**wc -l a.txt 🡺** number of lines

**wc -w** a.txt 🡺number of words

**wc -c** a.txt 🡺number of character

**head**

**by default top 10 lines**

**head a.txt**  🡺top 10 lines

**head -5 a.txt**  🡺top 5 lines

**head -n3 a.txt**  🡺top 3 line along with line number

**head +5 a.txt**  🡺from 5th line to top

**cat a.txt|head -3 🡺** top 3 lines

**Tail**

last 10 line by default

**tail -3 a.txt** 🡺 last 3 lines

**tail -f a.log**  🡺status of the log file. continues flow

**tail +5 a.txt** 🡺 from 5th line to end end

**vi** **editor**

**vi a.txt**

**3 types of mode in vi editor**

* command mode 🡺 **esc key**
* insert mode 🡺 **I, a, I, A**
* colon mode 🡺 **esc :**

**Insert mode:**

i 🡺insert data left to the cursor

a 🡺insert data right to the cursor

I 🡺insert data at beginning of line

A 🡺insert data at end of line

**command mode:**

**j**  🡺move cursor down

**k** 🡺move cursor up

**h** 🡺move cursor left

**l**  🡺move cursor right

**0** 🡺move cursor at beginning of line

**1**  🡺move cursor at beginning of line

**^**  🡺move cursor at beginning of line

**10|**  🡺move cursor to 10th charecter

**20|**  🡺move cursor to 20th character

**G** 🡺move cursor to the last line. begining of last line

**10G**  🡺move cursor to the 10th line

**ZZ**  🡺save and exit the file.(upper case)

**o** 🡺creates a blank space below the cursor and came to insert mode (oh)

**x** 🡺deletes the under cursor character

**w**  🡺moves the cursor to the begining of next word

**e** 🡺moves the cursor to the end of same word

**b** 🡺moves the cursor to the begining of previous word

**dd**  🡺delete the line under cursor

**u** 🡺undo the last change

**U** 🡺undo all changes in the line

**J** 🡺join the line wherever the cursor is

**2J** 🡺join 2 lines

**set nu**  🡺set the line number in a file

**set no nu**  🡺remove line number

**d$** 🡺delete from current cursor position to end of the line

**D** 🡺delete from current cursor position to end of the line

**dG** 🡺deletes from current cursor position to end of file

**d^** 🡺delete current cursor position to begining of line

**.** 🡺 dot is used to execute the previous command

**dH** 🡺delete from current cursor position to begining of the page

**ctrl+f** 🡺used to see the file page by page

**ctrl+b** 🡺scroll the page in backward direction

**yy** 🡺copy the current line

**p** 🡺paste

**3G 3yy 12G p** 🡺 copy from 3rd line to 5th line.paste it after 12th line

**Yw**  🡺 copy the word

**dd**  🡺cut and paste

**r**  🡺 replace under cursor character with single character

**R**  🡺 replace under cursor character with multiple char

**s**  🡺substitute with multiple under cursor character

**(**  🡺move the cursor to the beginning of the sentence

**)**  🡺move the cursor to the end of the sentence

**{**  🡺move to beginning of paragraph

**}**  🡺move to end of paragrah

**searching**

pattern: substring of a string

/Pattern 🡺 search in forward direction from cursor position

/hi

**n**  🡺go to next pattern

**N**  🡺reverse the search.previous pattern

**?**  🡺search the pattern in backward direction

**colon mode:**

we can execute the command in colon mode

**:1,$w test.txt**

**:1,$ w! test.txt** 🡺 forcefully

**:1,5 w a.txt** 🡺copy from one file to another

**:10,15 w a.txt**

**.w >> a.txt** 🡺current cursor to a.txt

**5,10 t 16** 🡺copy 5 to 10 line paste after 16th line

**5,10 t $** 🡺copy 5 to 10th line paste it in end

**5,10 m 16** 🡺 cut 5 to 10 line and paste after 16th line

**e! c.txt**  🡺open different file in vi editor

**1G 5 yy**  🡺copy 1 to 5th line

**e! c.txt**  🡺open c.txt

**4G p**  🡺paste the te copied line

**vi editor to prompt**

**:sh**

**ctrl + d**  🡺return to vi editor

**:!cal**

**:!echo hi**

$vi a.txt b.txt c.txt it opens a.txt. :n b.txt

:n c.txt

**substitution**

**:1,2 s/old/new**

**:1,2 s/hi/bye**  🡺substitute the pattern in first appearance of line

**:1,$ s/hi/bye**  🡺 substitute the pattern from 1st line to end

**:1,5 s/hi/bye/g**  🡺substitute globally. every occurrence

for save we must go colon mode

**:W**  🡺save the file

**:wq**  🡺save and exit

**:q!**  🡺quit forcefully without saving.

**:x**  🡺save and exit

**grep**

global regular expression and pattern

**grep option pattern file\_name**

**grep hi a.txt**  🡺it displays the content those contain hi

**grep -i hi a.txt**  🡺 used to ignore the case

**grep -n hi a.txt**  🡺shows the result along with line umber

**grep "hello hi" a.txt**

**grep -v hi a.txt**  🡺shows the line those dont contain hi

**grep -c hi a.txt**  🡺counts the pattern number as per the first appearance in line. 6 or 7 or number of times pattern present

**grep -o hi a.txt**  🡺show the pattern in a single column

**grep -o hi a.txt|wc -w**  🡺displays the no. of patterns

**grep -e hi -e bye a.txt**  🡺displays multiple pattern search

**grep -w hi a.txt**  🡺search the exact pattern

**grep -v ^ $ a.txt>b.txt**  🡺remove blank line from file a and store it in file b

**grep -l "hi" \***  🡺search all files that contains pattern h

**grep '[0-9]\{10\}' tel.txt**  🡺search the mobile numbers

\{6\} 6 times

**E GREP:**

extended global regular expression and pattern.it supports bre + ERE

+ 1 or more occurrence of previous character

? 0 or 1

| alternat

egrep "[aA]g+[ra][ar]wal" a.txt

egrep "(t|m|n)ina" a.txt 🡺search tina mina nina

**FGREP**

file grep or fixed grep

1c

hello

log

promo

zpq

a35

🡺 search all above patterns in a.txt

* copy all these patterns to a new file
* call it in fgrep

**fgrep -f a.txt hello.txt**

**fgrep -f a.txt < hello.txt**

**crontab:**

* for periodic scheduling we use crontab
* crontab is a file which contains 6 fields.each fields are separated by space
* if one field having multiple option then options are separated by ,
* cron is a daemon process
* it wake and sleep at every minute wheather there are any jobs to run or execute in crontab.
* /etc/cron.allow contains username those are allowed to schedule
* /etc/cron.deny

**crontab -e** schedule opens crontab file in vi editor.

**Min hour date month day\_of\_the\_week script\_name**

00-59 00-23 1-31 1-12 0-6 a.sh

**Runa a job 7 am to 9 am in every 2 mins**

\*/2 7-9 00 \* \* \* b.sh

**Run a job at 7 to 9.25 with 15 mins interval**

00,15, 7,8 \* \* \* c.ksh

30,45

00,15,25 9 \* \* \* c.sh

**Run a job with 15 mins of interval**

00,15, \* \* \* \* d.ksh

30,45

**Run a job in every 5 seconds**

#!/bin/ksh

while true

do

/home/nihar/backup.ksh

sleep 5

done

**Run a job interval of 5 min between 7:30 to 8**

30/5 7-8 \* \* \* e.ksh

**Run a job in every 2 hour**

\* \*/2 \* \* \* f.sh

**Run a job alternate date 2 pm to 5 pm every 3 min**

\*/3 14-17 \*/2 \* \* g.ksh

**alternate Sunday**

\* \* \* \* 0/2 h.ksh

**1st Sunday of every month**

\* \* 1-7 \* 0 f.ksh

**2nd Saturday**

\* \* 8-14 \* 6 j.ksh

**run a job in crontab in background**

\* \* \* \* \* k.sh &

crontab -l contains username those are allowed to schedule display all scheduled jobs

crontab -r contains username those are allowed to schedule delete the crontab file

* The crontab files are stored in /var/spool/cron/crontab/
* crontab -e -u username Root user will edit the crontab file of a user

**fork() & exec()**

fork() is responsible for creating a child process and creates like a clone of parent.It is a system call.

It is responsible for copying all the programme to child and execute it.

**tar**

for archiving/backup we use tar command

a.txt

+

b.txt

+

c.txt >> a.tar

**tar -cvf a.tar a.txt b.txt**  🡺creating tar file

**tar -xvf a.tar**  🡺extract the tar file

**tar -tvf a.tar**  🡺it will display files inside a tree

**tar -xvfz a.tar.gz**  🡺extracted directly

**ZIP:**

it is used to process the file

* **zip🡺unzip .z**
* **gzip🡺gunzip .gz**
* **compress🡺uncompress**

**gzip a.tar**

**gunzip a.tar.gz**

**gzip -d a.tar.gz**  🡺extract the files inside a directory

**gzip -R /home/dir1/nihar**  🡺It will zip all the content of directory

**gunzip -R dir1**

**gzip -l a.tar**  🡺it shows the compression size of the file

**zcat a.tar.gz**  🡺displays all contents of files

**zless**

**zmore a.tar.gz**

**tar -uvf a.tar p.txt m.txt**  🡺add two more files in tar file

**FIND**

find <address\_from\_where\_you\_want\_to\_search> search\_criteria\_option

/

**-name**

**-type**

**-user**

**-inum**

**-perm**

**-size**

**-mtime days**

**-atime**

**-ctime**

**-newer**

**-mmin minutes**

**-amin**

**find / -type f -name \*.txt -print** 🡺search all txt files from root

find / -name "ora" -type f -print

find / -type d -name "ora" -print search directories

**searching files having permission**

find . -type f -perm 567 -print

find / -user Raj -type f -perm 666 -print 🡺 for a particular user

find / -type f -inum 12345 -print particular inode number

**search files having size empty**

find / -type f -size 0 -print

**delete the files having size zero**

find / -type f -size 0 -delete

**display all files having size 20mb**

find / -type f -size 20480 c -print

**file greater than 20 mb**

find / -type f -size +20480 c

**files less than 20 mb**

find / -type f -size -20480 c

**search both text and .sh files**

find / -type f -name \*sh -name \*txt -print

find / -type f \(-name \*sh -o \*txt \) -print

**display all files modified 2 days before**

find / -type f -mtime +2

**modified within 2 days**

find / -type f -mtime -2

**modified exact 2 days**

find / -type f -mtime 2

**display files of user raj modifoed within 5 raj**

find / -user Raj -type f -mtime -5

**search all files modified between 5 to 10 days**

find / -type f -mtime +5 -mtime -10 -print

**modified within 2 hour**

find / -type f -mmin -120 -print

find / -type f -amin -120 -print

**one file created name a.txt before 2 days. display all files created after a.txt**

find / -type f -newer a.txt -print

**display files created after 15th jan**

touch -m 201201150000 a.txt

find / -type f -newer a.txt

**display all files created in between 15jan2012 to 15feb2012**

touch -m 201201150000 a.txt

touch -m 201202150000 b.txt

find / -type f \(-newer a.txt -a -newr b.txt\)

**3 terminologies used in find command**

**| xargs**

**- exece**

**- ok**

ok and exec are same but ok is interactive

**search pattern in all those files in entire directory structure**

find / -type f |xargs grep -i "ora"

**by default xargs takes 20 files**

find / -type f |xargs -50 grep "ora" it will take 50 lines from find output

**search all .sh files and give execute permission**

find / -type f -name \*.sh -exec chmod u+x

**search all .sh files and make tar of it**

find / -type f -name \*.sh |xargs tar -cvf a.tar

**search all logfiles older than 5 days and remove it**

find / -type f -name \*.log -mtime +5 |xargs rm -f {}\;

**send all errors to a different file.dont show errors in the result**

find / -type f -name \*.sh -exec cp {} {}.bak\; 2>/dev/null

find / -type f -name \*sh 1>a.txt

2>b.txt

**Associate operator**

>

&

find / -type f -name \*.s 1>a.txt

2>&1 🡺it will copy the error logs to the field descriptor. That means the same files contains o/p a.txt

1>/dev/null 2>&1

**Touch**

create an empty file

it sets the time stamp

touch a.txt if the file does not exist ,it will create time stamp

if it already exists, it changes the time stamp to current date

**touch -m 201201150000 a.txt**  🡺modification time

**touch -a 201201150000 a.txt** 🡺access time

soft link 🡺

* link will be removed if the file is removed or renamed
* **ln -s /home/nihar/s.txt**
* Inode number is different in both file and link

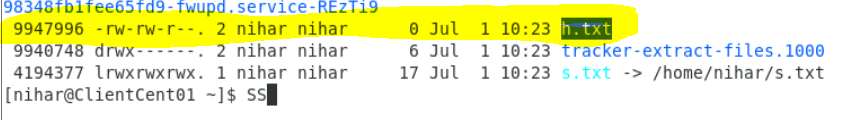
it will create the link as below

A picture containing text

Description automatically generated

Hard link 🡺

* deleting renaming or moving the original file will not affect the hard link
* **ln /home/h.txt**
* hard link is a file which will point to the same inode number of the source file.
* that’s why when you delete the source file, it is still accessible



Table

Description automatically generated

**Access control list 🡺**

* Access control list (ACL) provides an additional, more flexible permission mechanism for file systems.
* It is designed to assist with UNIX file permissions.
* ACL allows you to give permissions for any specific user or group to any disc resource
* Use of ACL : Think of a scenario in which a particular user is not a member of group created by you but still you want to give some read or write access, how can you do it without making user a member of group, here comes in picture Access Control Lists.
* ACL helps us to do this trick. Basically, ACLs are used to make a flexible permission mechanism in Linux.
* **setfacl and getfacl** are used for setting up ACL and showing ACL respectively

For example : getfacl test/seinfeld.txt

Output:

# file: test/seinfeld.txt

# owner: iafzal

# group: iafzal user::rwgroup::rwother::r--

To add permission for an user

* setfacl -m "u:user:permissions" /path/to/file

To add permissions for a group

* setfacl -m "g:group:permissions" /path/to/file

To allow all files or directories to inherit ACL entries from the directory it is within

* setfacl -dm "entry" /path/to/dir

To remove a specific entry

* setfacl -x "entry" /path/to/file

To remove all entries

* setfacl -b path/to/file

For example :

* setfacl -m u:iafzal:rwx test/seinfeld.txt

Modifying ACL using setfacl :

To add permissions for a user (user is either the user name or ID):

# setfacl -m "u:user:permissions"

To add permissions for a group (group is either the group name or ID):

# setfacl -m "g:group:permissions"

To allow all files or directories to inherit ACL entries from the directory it is within:

# setfacl -dm "entry"

Example :

setfacl -m u:iafzal:r-x test/seinfeld.txt

View ACL : To show permissions :

Observe the difference between output of getfacl command before and after setting up ACL permissions using setfacl command.

# getfacl filename

Remove ACL : If you want to remove the set ACL permissions, use setfacl command with -b option. For example : remove set permissions

If you compare output of getfacl command before and after using setfacl command with -b option, you can observe that there is no particular entry for user nihar in later output.

You can also check if there are any extra permissions set through ACL using ls command.

check set acl with ls Observe the first command output in image, there is extra “+” sign after the permissions like -rw-rwxr– +, this indicates there are extra ACL permissions set which you can check by getfacl command

**Help Commands 🡺**

3 types of help commands

* whatis
* --help
* man
* whatis ls
* ls --help
* man ls

A screenshot of a computer

Description automatically generated with medium confidence

**INput & Output redirects 🡺**

* standard input (stdin) and the file descriptor number is 0
* standard output (stdout) and it has file descriptor number is 1
* standard error (stderr) and it has file descriptor is 2
* ls -lrt 1>> a.txt
* ls -lrt >>a.txt
* ls -lrt 2>>a.txt

**standard output to a file (tee) 🡺**

* tee command is used to store and view (both at the same time) the output of any command
* the command is name after the T-splitter used in plumbing.
* It basically breaks the output of a program so that it can be both displayed and saved in file. It does both the tasks simultaneously. copies the result into the specified files or variables and also displays the result.

**FIle display commands 🡺**

cat

more 🡺 views connect one page at a time

less 🡺 same as more , shows page by page, however you can do other operation as well. like view one line by line. you can use up n down arrow key to scroll.

head

tail

**Cut Processor commands 🡺**

cut

awk

grep and egrep

sort

uniq

wc

**CUT 🡺**

* cut is a command line utility that allows you to cut parts of lines from specified files or piped data and print the result to standard output.
* It can be used to cut parts of a line by delimiter , byte position and character.

cut filename 🡺 does not work

cut --version 🡺 check version

cut -c1 filename 🡺 list one char

cut -c1,2,4 🡺 pick and choose char

cut -c1-3 filename 🡺 list range of chars

cut -b1-3 filename 🡺 list by byte size

cut -d: -f 6 /etc/passwd 🡺 delimeter : and field 6

cut -d: -f 6-7 /etc/filename 🡺 list 6th n 7th field delimited by :

**AWK🡺**

* awk is a utility /language designed for data extraction. Most of time it is used to extract fields from a file or from an output.
* cat awk.txt | awk ‘{print $1}’ 🡺 print the first column or char, depends upon the file
* cat awk.txt | awk ‘{print $1,$4}’ 🡺 print 1st and 4th column
* cat awk.txt| awk ‘{print $NF}’ 🡺 last field of the output or the file
* cat awk.txt| awk ‘{print NF }’ 🡺 total number of field
* awk ‘/nihar/ {print}’ awk.txt 🡺 search for a specific word
* awk -F: ‘{print $1}’ /etc/password 🡺 checks the field descriptor as : and print first field
* echo “hello Tom” | awk ‘{$2=”Adam”; print $0} 🡺replace tom with adam and print on the screen
* ls -lrt | awk ‘{if($4==”root”) print $0;}’ 🡺 checks if 4th field is root, if it is root it will print.
* ls -lrt | awk ‘{if $4==”root”) ; print $0}’ 🡺 if you put ; before print 0, first if check operation will end there and print $0 will print the whole file without the if operation. it wont give proper out put

cat awk.txt|awk ‘{if ($4==”A”) $4=”A is found”; print $0}’ 🡺 check if $4 is A, if it is A it will display the whole file by changing the 4th field to “A is found

when you put ; before print $0. It will print the whole file

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

A black screen with white text

Description automatically generated with low confidence

Text

Description automatically generated

**SORT & UNIQ 🡺**

sort command sorts in alphabetical order

uniq command filters out the repeated or duplicate line

sort --version 🡺 sows the version of sort

sort file 🡺 sort file in alphabetical order

sort -r a.txt 🡺 sort in reverse order

sort -k2 a.txt 🡺 sort by field no 2. means sort by 2nd column

uniq a.txt 🡺 it wont work. it will show the o/p but it wont give any uniq result.

sort a.txt|uniq 🡺 always use sort first before using uniq

sort a.txt | uniq -c 🡺 uiq and list count

**sort file | uniq -d 🡺 only show repeated line or duplicate lines**

Text

Description automatically generated

**Compare files 🡺**

diff 🡺 line by line

cmp 🡺 byte by byte

diff file1 file2 🡺 it will show difference lines in both the files. << shows the file1 and >>file2. it will display only lines which are different or not present in each other.

cmp file1 file2 🡺 it will show only line numbers, which are different. it will not show the content.

Text

Description automatically generated

**Truncate 🡺**

truncate is used to reduce the file size to any specified size

truncate -s 40 filename.txt 🡺 it will increase or decrease the file size to 40.

if the file size is 50, it will 10 bytes data and reduces to 40, in thecae there is data loss.

if the size is 30 and you have truncate it to 40 it will increase the file by adding some extra chars.

**SPlitting files** 🡺

split -l 300 sourcefilename newfile 🡺 it will split the sourcefilename by taking 300 line and put into a new file called newfile. however if the sourcefile is having 1200 lines. it will create 4 files named as newfileaa, newfileab,newfileac,newfilead

**SED 🡺 streamline editor**

replace a string in a file with a new string

find and delete a line

remove empty line

to replace tabs with spaces

show defined lines from a file

substitute within vi editor

**USE OPTION -i to** get the changes reflected in original file. If you don’t use option -I ,

it will only display in the screen.

sed ‘s/hi/bye/g’ a.txt 🡺 search hi and replace with bye globally in file , but it will not change the file, it will display only in the screen.

sed -i ‘s/old/new/g’ a.txt 🡺 I option is insert mode, here it will update the original file.

sed ‘1s/old/new/g’ a.txt 🡺 search and replace in line 1 (only display, not update in file)

sed ‘1!s/old/new/g’ a.txt 🡺 search and replace except line 1 (only display not update in file)

sed ‘s/ /\*\*/g’ a.txt 🡺 search and replace space with \*\* . (only display)

sed ‘s/ //g’ a.txt 🡺 search and replace space with no space . (only display)

sed ‘s/\t/ /g’ a.txt 🡺 search and replace /t (tab) with space . (only display)

sed 3,4d a.txt 🡺 delete 3 & 4th line in a.txt . (only display)

sed -i 3,4d a.txt 🡺 delete 3 & 4th

**sed ‘/^$/d’ a.txt 🡺 delete the blank line ^$ (blank)**

**sed ‘/\t/d’ a.txt 🡺 delete the tab space \t**

**sed ‘s/\t/ /g’ a.txt 🡺 search and replace /t (tab) with space . (only display)**



**User account management 🡺**

useradd

groupadd

userdel

groupdel

usermod

files

**/etc/passwd** 🡺 Traditionally, the /etc/passwd file is used to keep track of every registered user that has access to a system.

The /etc/passwd file is a colon-separated file that contains the following information:

User name

Encrypted password

User ID number (UID)

User's group ID number (GID)

Full name of the user (GECOS)

User home directory

Login shell

**/etc/group** 🡺

The **/etc/group** file contains basic group attributes. This is an ASCII file that contains records for system groups. Each record appears on a single line and is the following format:

Name**:**Password**:**ID**:**User1,User2,...**,**Usern

**/etc/shadow**  🡺 A shadow password file, also known as /etc/shadow, is a system file in Linux that stores encrypted user passwords and is accessible only to the root user, preventing unauthorized users or malicious actors from breaking into the system

useradd -g <groupname> -s /bin/bash -c “user description” -m -d /home/nihar nihar

**Password ageing 🡺**

/etc/login.def 🡺 this file contains information about the password ageing information.

change -m mindays -M maxdays -d lastday -I inactiveday -E expiry date

-W warndays username

change -m 5 -M90 -W10 -I 3 nihar 🡺 change the password ageing for user nihar, it will expiry in 90 days, will give warning before 10 days and will be inactive in 3 days after it gets expired.

**Switch users and Sudo access** 🡺

root is the master user and can only give su access to any user.

to give any user to root level permission you need to add the user in visudo.

su - user name 🡺 switch user

sudo 🡺 switch to sudo mode if you have password

**visudo 🡺 it will open the sudo file in vi mode, here you can add the user for sudo access.**

file 🡺 /etc/sudoers

visudo file 🡺

Text

Description automatically generated

**Monitor users 🡺**

**who** 🡺 it will display who is logged in now

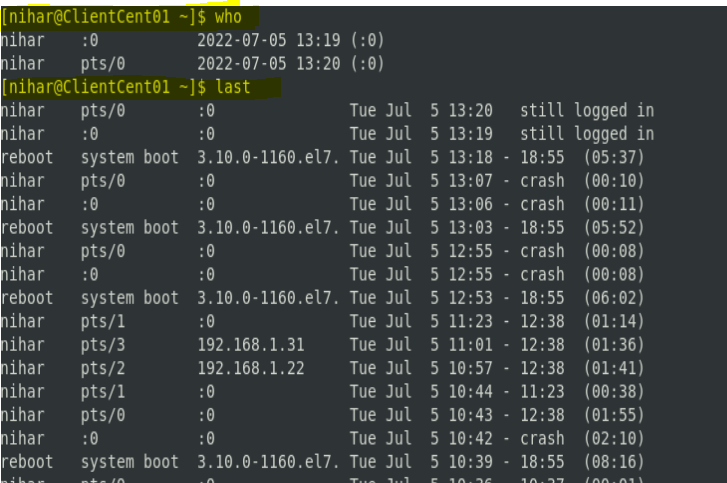
**last** 🡺 it will display who has logged in in details from day 1

**w** 🡺 w is same as who but it will give information about load average and some more information

finger

id 🡺 information of the user.

id nihar 🡺 information about the specific user.



Text

Description automatically generated

Graphical user interface, application, website

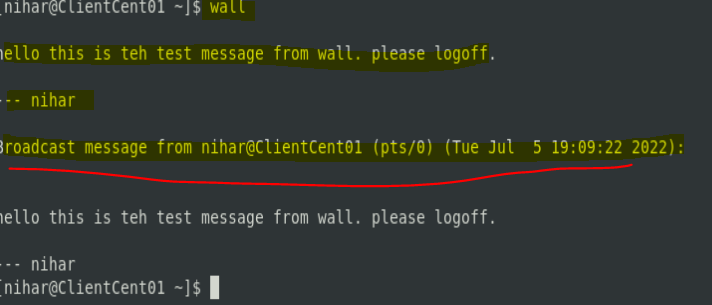
Description automatically generated

**communication between the users 🡺**

users 🡺 it will show the user name only who has current session opened.

wall 🡺 It will write message on screen of every user who has open session.

wall then enter… write message. use ctrl +d to save and exit.



**write** 🡺 it will write or ping the user with specified text. it is like the ping we do in skype.

write nihar 🡺

hey nihar

ctrl cS

Text

Description automatically generated

message wil display like below

Text

Description automatically generated

**Linux Account authentication 🡺**

Types of accounts :

Local accounts

Domain/directory accounts

**System utility commands 🡺**

uname

hostname

uptime

which

cal

bc

date

Process and jobs 🡺

application = service ex: apache,ntp

script :- shell scripts or list of instructions

process : when a service runs 🡪 creates a process id

daemon : runs until interrupted

threads : service 🡪 process 🡪 thread 1

job

**systemctl** 🡺

systemctl command is a new tool to control system services.

it is available in version 7 and later, it replaces service command.

systemctl start|stop|status servicename.service

systemctl enable servicename.servuce

systemctl restart|reload servicename.service

systemctl list-units --all

Text

Description automatically generated

To add a service under systemctl management

create a unit file in /etc/system/system/servicename.service

**You can control system with systemctl**

**systemctl poweroff**

**systemctl halt**

**systemctl reboot**

PS 🡺

ps 🡺 shows the process of the current shell

ps -e 🡺 shows all running process

ps aux 🡺 shows all running process in BSD

ps -ef 🡺 show all running process in full format listing

ps -u nihar 🡺 shows all running process by username nihar

**top** 🡺

top command is used to show the linux process and provides a real time view of the of the running system.

this command shows the summary information of the syetm and list of process or threads which are currently managed by linux kernel

when the top command is executed then it goes to interactive mode and you can exit by hitting q .

top 🡺 shows all running process along with load average

top -u nihar 🡺 all running process by a particular user

top then press c 🡺 shows absolute path of the commands

top then press k 🡺 kill a process by pid within top session

top then M and P 🡺 to sort all Linux running process by memory usage

Note : top command refreshes the information every 3 seconds

Graphical user interface, text

Description automatically generated

A picture containing text

Description automatically generated

Text

Description automatically generated

Kill 🡺

kill is used to terminate a process.

kill option pid

option = signal name or signal number/id

pid = process id

kill -l 🡺 (small L )to get a list of all signal names or signal number

kill pid 🡺 kill a process with default signal

kill -1 🡺 restart

kill -2 🡺 interrupt from the keyboard just like ctrl c

kill -9 🡺 forcefully kill the process

kill -15 🡺 normal kill

killall 🡺 kill all the process

pkill 🡺 kill by process name

crontab 🡺

crontab -e 🡺 edit the crontab table

crontab -l 🡺 list the crontab entries

crontab -r 🡺 remove crontab

crond 🡺 crontab daemon /service that manages scheduling

systemctl status crond 🡺 to manage the crond service

**Process management** 🡺

jobs 🡺 list out all back ground process

bg 🡺 execute the halt process in background

background 🡺 ctrl + z , jobs, bg

foreground 🡺 fg

run a process even after exit 🡺 nohup process &

or

nohup process > /dev/null 2>$1 &

kill a process by name 🡺 pkill

process priority 🡺 nice

nice -n 5 processid

the niceness scale goes to from -20 to 19.

The lower the number more the priority that task gets.

process monitoring 🡺 top

list process 🡺 ps

**System Monitoring 🡺**

top

df 🡺 disk portioning information, it would show total size of the disk and the files consumed the space

du 🡺 disk usage of each file. it would show the file and the space taken

dmesg

iostat 1

netstat

free

cat /proc/cpuinfo

cat /proc/meminfo

dmesg 🡺 all error message / warning of the system

dmesg | more 🡺 page by page all system related warning and error messages

iostat 🡺 input output stats. how we are communicating system peripheral devices, system internal devices.

iostat -1 🡺 it will show the same result by refreshing every 1 second

netstat -rnv 🡺 gateway information, subnet mask

netstat| more 🡺 what is connected and disconnected to the system and name n owner of the process

free 🡺 physical memory and its usage

cat /proc/cpuinfo 🡺 cpu info

cat /proc/meminfo 🡺 memory information

**Log monitoring🡺**

log directory = /var/log 🡺 primary log directory of linux system. you can get below log information in the var/log directory

boot :

chronyd : NTP

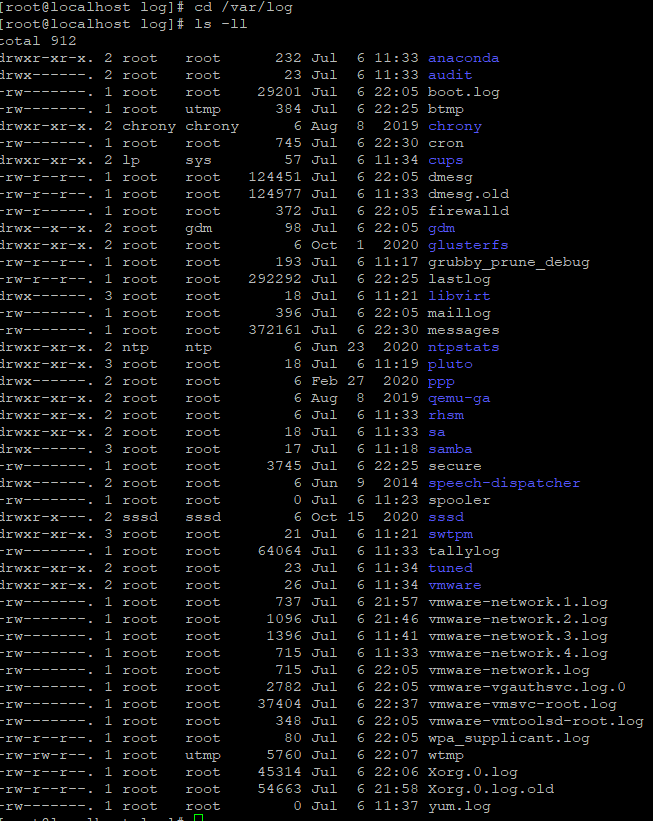
cron

maillog

secure 🡺 more secure 🡺 gives information of all users who has tried logging in and the error details.

message 🡺 this file has all hardware/software/process information logs.

httpd



**System Maintenance commands 🡺**

shutdown 🡺 shutdown the system

init 0 🡺 normal shutdown

init 3 🡺 halt the machine

init 4 🡺 power off normal

init 5 🡺 reboot

init 6 🡺 reboot with kexec

init 13 🡺 immidatly halt the machine

init 14 🡺 immidialt powers off

init 15 🡺 immediately reboot

init 16 🡺 immidialty reboot with kexec.

reboot 🡺 reboot

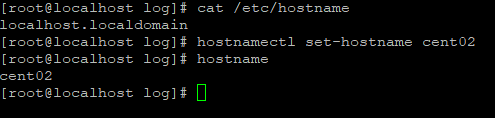
halt 🡺 shuts down the system right way like pressing your power button.

**changing system hostaname**🡺

* hostnamectl set-hostaname newhostanme

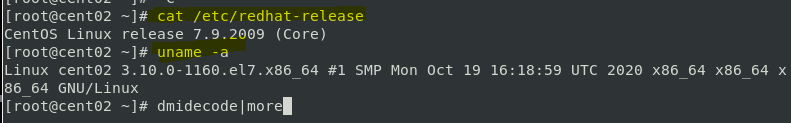
ex: hostnamectl set-hostname cent02

file for hostname /etc/hostname 🡺 you can modify the file to change the hostname as well, followed by a reboot.



**Finding system Information** 🡺

* cat /etc/redhat-releasse 🡺 information about the os distribution
* uname -a 🡺 os distribution and microprocessor details
* demidcode | more 🡺 bios information, memory, product,seriel number, manufacture,processor information and many more in details
* arch 🡺 shows if the system is 32 bit or 64 bit architecture



Text

Description automatically generated

**Terminal control keys 🡺**

CTRL + u 🡺 erase everything you have typed

CTRL + c 🡺 stop/kill a command

CTRL + z 🡺 kill the stuck process and get out of the process, suspend the command

CTRL +d 🡺 exit from an interactive program

**Terminal commands 🡺**

clear 🡺 clear your screen

exit 🡺 exit out of the shell , terminal or a user session

script 🡺 the script command stores terminal activities in a log file that can be named by a user, when a name is not provided by a user, the default filename , typescript is used.

Text

Description automatically generated

**Recover Root Password** 🡺

restart you computer

edit grub

change password

reboot

**SOS Report** 🡺

collect and package diagnostic report

package name : sos-version

command: sosreport

**Environment variables🡺**

set of defined rules and values to build and environment,

printenv 🡺 to view all environment variable

env 🡺 to view all environmental variable

echo $SHELL 🡺 to view one environment variable use echo

export TEST=1 🡺 set environment variable

echo $TEST

vi .bashrc 🡺 to set environment variable permanently

TEST=’123’

export TEST

vi /etc/profile or /etc/barshrc 🡺 **to set global environment variable permanently**

Test=’123’

export TEST

Special Permission with setuid,setguid and sticybit 🡺

chmod u+s xyz.sh 🡺 to assign special permission at the user level

cmod g+s xyz.sh 🡺 to assign special permission at the group level

chmod u-s xyz.sh

chmod g-s xyz.sh 🡺 to remove special permission

find / -per /600 -type f 🡺 to find all executable in linux with setuid and setguid

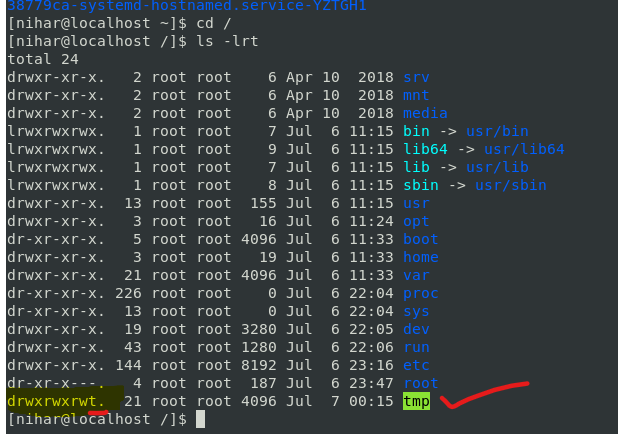
stcikybit 🡺 it is assigned to the last bit of the permission

-rwx rwx rwt

ex: /tmp directory

that means any user having access to /tmp can not delete the tmp directory

chmod +t file.txt 🡺 assigning sticky bit to the file



**Aliases** 🡺

user: Applies only to a specific user profile

global : applies to everyone who has account on the system

user=/home/user/.bashrc

global=/etc/bashrc

alias hh=”hostname”

* you can add alias in bashrc file to access the aliases globally
* aliases can be setup using alias command for user alias.
* The alias command allows you to define new commands.
* Useful for creating shortcuts for longer commands.

The syntax is.

alias alias-name=executed\_command

Some examples:

alias m=more

alias rm="rm -i"

alias h="history -r | more"

To view all current aliases:

alias

To remove a previously defined alias:

unalias alias\_name

**History** 🡺

history 🡺 shows the history

history | grep awk 🡺 commands executed awk

cat /home/users-dire-name/.bash\_history 🡺 file to check history of any user

!history\_number

ex: !456 🡺 it will execute the 456 number of command from history

**Networking , Services & system updates**

Network components 🡺

IP: internet protocol

subnet mask

gateway

static vs DHCP

Interface

mac address

**Network files & commands 🡺**

Interface detection

Assigning IP address

interface configuration files

/etc/nsswitch.conf 🡺

/etc/hostname 🡺 file where we configure system hostname and ip adreess

/etc/sysconfig/network 🡺 contains hostname

/etc/sysconfig/network-script 🡺 networking interface details

/etc/resolv.conf 🡺

ping

ifconfig

ifup or ifdown

tcpdump 🡺 traces every single transaction to / from your machine

ex : tcpdump -I enps03 🡺 enp0s3 is the interface

NIC information 🡺

NIC is network interface card.

NIC can have multiple ports.

it is the port assigned to the ethernet port.

ethtool enp0s3 🡺

different type of NIC:

enp0s3

lo: the loopback device is a special interface that your computer uses to communicate with itself. it is used mainly for diagnostics and troubleshooting and to connect to servers running on the local machine.

virb0: the virbr0 or virtual bridge 0 interface is used for NAT (Network address translation). Virtual environment sometimes use it to connect to the outside network.

**NIC Bonding 🡺**

NIC bonding is also known as network bonding. it can be defind as the aggregation or combination of multiple NIC into a single bond interface.

its main purpose is to provide high availability and redundancy

modprobe bonding

modinfo bonding

**FTP- File Transfer Protocol**

Default port fr ftp = 21

protocol 🡺 set of rules used by computers to communicate

Install and configure FTP on the remote server

rsync - Remote synchronization

rsync is a utility ficiently transferring and synchronizing files within the same computer or to a remote computer by comparing the modification times and sizes fo files.

rsync us a lot faster than ftp and scp

the usitlity is mostly used to backup the files and directories from one server to another

default rsync prt =22 same as ssh/scp

rsync options source destination

How to install rsync 🡺

yum install rsync

apt-get install inst rsync

rsync a file on local machine 🡺

tar cvf backup.tar

mkdir /tmp/backups

rsync -zvh backup.tar /tmp/backups/

rsync a directory on a local machine

rsync -azvh /home/nihar /tmp/backups/

rsync a file to a remote machine

rsync -avz backup.tar [nihar@192.1681.1.0:/tmp/backups](mailto:nihar@192.1681.1.0:/tmp/backups)

rsync a file from a remote machine

rsync -avzh [nihar@192.168.1.0:/home/nihar/filetobecopy](mailto:nihar@192.168.1.0:/home/nihar/filetobecopy) /tmp/destinationfile

**system updates and Repos 🡺**

yum 🡺 centos

apt-get 🡺 othe rlinux

rpm 🡺 redhat package manager

System upgrade/patch management 🡺

two types of upgrades :

major version : 5,6,7

minor version: 7.3 to 7.4

major version = yum will not do

minor version = yum will do

ex” yum update -y

difference vetweeb update and upgrade🡺

upgrade - delete package and install new version

update: preserve the old package and then install new

Advance package management 🡺

install 🡺 yum install package-name

upgrade 🡺

delete🡺 rpm -e pkg-name or yum remove pkg-name

inofmrtaion about the pkg 🡺 rpm -qi pkg-name

rpm -qf pkg-name 🡺 pkg details

Rollback updates and patches 🡺

roll back a package or patc:

yum install pkg-name

yum history undo <id>

rollback an update : downgrading a system to minor version (is not recomened)

yum update

yum upgrade

yum history undo <id>

SSH and Telnet🡺

telnet = unsecured

ssh = secured connection

two types of packages for most of the services

client package

server package

ps -ef| grep sshd 🡺 this will show how many connections from other system to your system

systemctl stop sshd 🡺 stop ssh service

DNS: Domain Name system 🡺

purpose 🡺

DNS is a service which does below functions

hostname to IP ( A record )

IP to hostname (PTR Rceord)

Hostame to hostname (cname record)

files🡺 /etc/named.conf

service 🡺 systemctl restart named

**Firewall** 🡺

What is Firewall 🡺 A wall that prevents the spread of fire

When data moves in and out of a server its packet information is tested against the firewall rules to see if it should be allowed or not

In simple words, a firewall is like a watchman, a bouncer, or a shield that has a set of rules given and based on that rule they decide who can enter and leave •

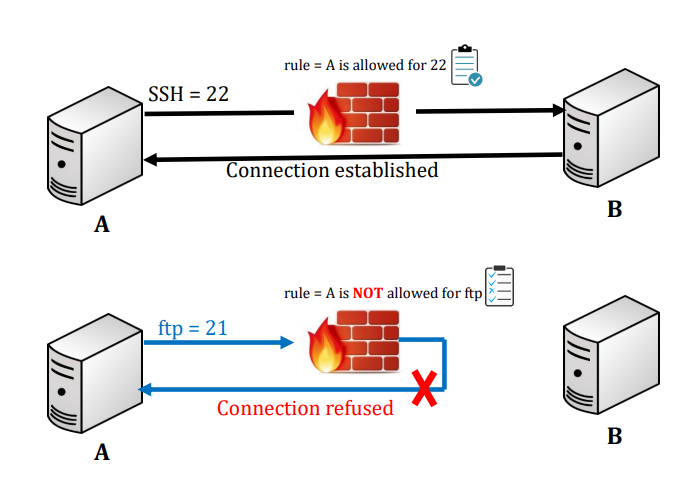
There are 2 type of firewalls in IT

Software = Runs on operating system

Hardware = A dedicated appliance with firewall software

Diagram

Description automatically generated



* There are 2 tools to manage firewall in most of the Linux distributions •
  + iptables = For older Linux versions but still widely used •
  + firewalld = For newer versions like 7 and up •

You can run one or the other .In this lecture we will work with iptables to manage firewall .Before working with iptables make sure firewalld is not running and disable it

service OR systemctl stop firewalld 🡺 To stop the service •

systemctl disable firewalld 🡺 To prevent from starting at boot time •

systemctl mask firewalld 🡺 To prevent it from running by other programs •

rpm –qa | grep iptables-services 🡺 Now check if you have iptables-services package installed

yum install iptables-services 🡺If not installed then •

systemctl start iptables 🡺 Start the service

systemctl enable iptables 🡺 Start the service

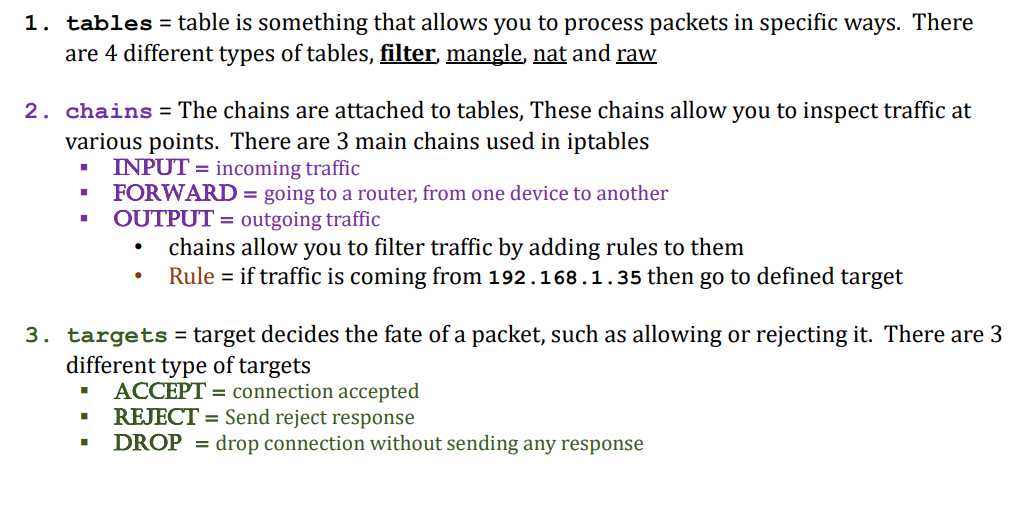
iptables –L 🡺 To check the iptables rules •

iptables -F 🡺 to flush iptables

IP TABLES 🡺

The function of iptables tool is packet filtering •

The packet filtering mechanism is organized into three different kinds of structures: tables, chains and targets



Diagram

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Drop all traffic coming from a specific IP (192.168.0.25)

• iptables –A INPUT –s 192.168.0.25 –j DROP

• Drop all traffic coming from a range of IPs (192.168.0.0)

• iptables –A INPUT –s 192.168.0.0/24 –j DROP

• List all rules in a table by line numbers

• iptables –L --line-numbers

• Delete a specific rule by line number

• iptables –D INPUT 1

• To flush the entire chain

• iptables –F

• To block a specific protocol with rejection (e.g. ICMP)

• iptables -A INPUT -p icmp -j REJECT

• To block a specific protocol without rejection (e.g. ICMP)

• iptables -A INPUT -p icmp -j DROP

• To block a specific port # (e.g. http port 80)

• iptables -A INPUT -p tcp --dport 80 -j DROP

• Block connection to a network interface

• iptables -A INPUT -i enps03 -s 192.168.0.25 -j DROP

• Drop all traffic going to [www.facebook.com](http://www.facebook.com)

• host -t a www.facebook.com = find IP address

• iptables –A OUTPUT –d 31.13.71.36 –j DROP

• Block all outgoing traffic to a network range

• iptables –A OUTPUT –d 31.13.71.0/24 –j DROP

• Block all incoming traffic except SSH

• iptables -A INPUT -p tcp --dport 22 -j ACCEPT

• iptables -P INPUT DROP

• After making all the changes save the iptables. Again make sure firewalld is not running

• iptables-save = The file is save in /etc/sysconfig/iptables

• iptables saved file can also be restored

• iptables-restore /LOCATION/FILENAME

• By default everything is logged in • /var/log/messages

**Firewalld** 🡺



Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

accept and reject traffic from ip address 🡺

Graphical user interface, text

Description automatically generated

**Tune system Performance 🡺 Tune-d**

Linux system comes fined tunned by default when you install, however there are a few tweaks that can be done based on system performance and application requirements In this lesson we will learn…

Optimize system performance by selecting a tuning profile managed by the tuned daemon

Prioritize or de-prioritize specific processes with the nice and renice commands

What is tuned?

Tuned pronounced as tune-d

Tune is for system tuning and d is for daemon

It is systemd service that is used to tune Linux system performance

It is installed in CentOS/Redhat version 7 and 8 by default

tuned package name is tuned

The tuned service comes with pre-defined profiles and settings (List of profile will be discussed in the next page)

Based on selected profile the tuned service automictically adjust system to get the best performance. E.g. tuned will adjust networking if you are downloading a large file or it will adjust IO settings if it detects high storage read/write

The tuned daemon applies system settings when the service starts or upon selection of a new tuning profile.



• Check if tuned package has been installed

rpm –qa | grep tuned

• Install tuned package if NOT installed already

yum install tuned

• Check tuned service status

systemctl status|enable|start tuned

systemctl enable tuned (To enable at boot time)

• Command to change setting for tuned daemon

tuned-adm

• To check which profile is active

tuned-adm active

• To list available profiles

tuned-adm list.

To change to desired profile

tuned-adm profile profile-name

• Check for tuned recommendation

tuned-adm recommend

• Turn off tuned setting daemon

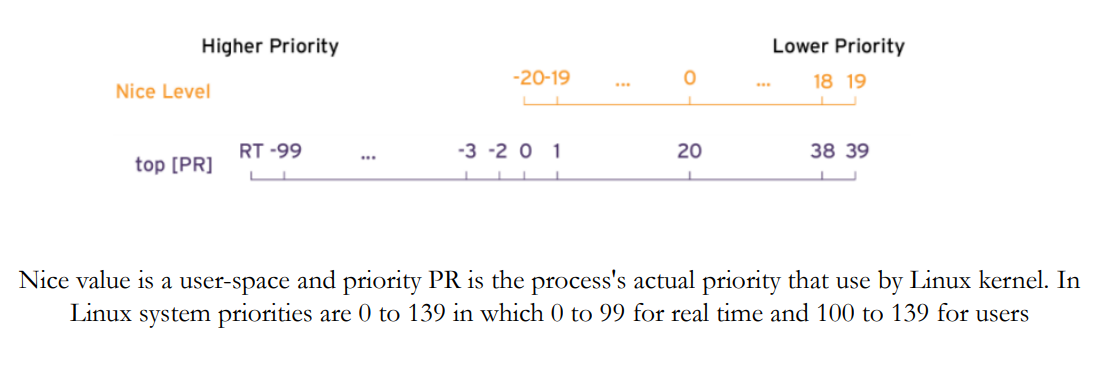
tuned-adm off

• Change profile through web console

Login to https://192.168.1.x:9090 Overview → Configuration → Performance profile

**NICE/RENICE 🡺**

* Another way of keeping your system fine-tuned is by prioritizing processes through nice and renice command
* If a server has 1 CPU then it can execute 1 computation/process at a time as they come in (first come first served) while other processes must wait
* With nice and renice commands we can make the system to give preference to certain processes than others
* This priority can be set at 40 different levels
* The nice level values range from -20 (highest priority) to 19 (lowest priority) and by default, processes inherit their nice level from their parent, which is usually 0.



To check process priority

top

Process priority can be viewed through ps command as well with the right options

$ ps axo pid,comm,nice,cls --sort=-nice

To set the process priority

nice –n # process-name

e.g. nice –n -15 top

To change the process priority

renice –n # process-name

e.g. renice –n 12 PID.

**DHCP stands for Dynamic Host Configuration Protocol**

• In order to communicate over the network, a computer needs to have an IP address

• DHCP server is responsible to automatically assign IP addresses to servers, laptops, desktops, and other devices on the network

• Wait a second…

• Right now in our home how IPs are assigned to our devices?

• Answer → The router or gateway given to you by your ISP provider •

**How IPs are assigned in corporate world?**

• Answer → Dedicated routers run DHCP service to assign IPs on the network

Step by steps instructions

• Pick a server to be your DHCP and take a snapshot

• Assign a static IP to the DHCP server

• vi /etc/sysconfig/network/enp0s3

• Or simply run nmtui command to use GUI based network tool

Diagram

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

• Install dhcp server package

• yum install dhcp (version 7)

• dnf install dhcp-server (version 8)

• Edit the configuration file with desired parameters

• vi /etc/dhcp/dhcp.conf

• cp /usr/share/doc/dhcp-x.x.x/dhcpd.conf.example /etc/dhcp/dhcpd.conf

Chart, line chart

Description automatically generated

Start dhcpd service

systemctl start dhcpd

systemctl enable dhcp

Disable firewalld or allow dhcp port over firewall

systemctl stop firewalld

OR

firewall-cmd --add-service=dhcp –permanent

firewall-cmd –reload

Switch DHCP service from your router/modem to your new DHCP server

Login to your ISP provided router

Disable dhcp and enable forwarding to the new dhcp server.

NTP 🡺

NTP port : 123

purpose : Time synchronization

File : /etc/ntp.conf

Service: systemctl restart ntpd

command : ntpq 🡺 it will come to ntpq command mode

ntpq > peers 🡺 shows the servers connected to the network for time

Chronyd 🡺

* it is use for time synchronization.
* package name is chronyd
* configuration file : /etc/chronyd.conf
* log file : /var/log/chrony
* service: systemctl start/restart chronyd
* command : chronyc
* chronyc 🡺 after enter 🡺 sources

timedatectl 🡺

timedatectl 🡺 to check time status

timedatectl list-timezones 🡺 to view all available zones

timedatectl set-timezone “America/new\_york” 🡺 to set time zones

timedatectl set-time YYYY-MM-DD 🡺 to set date

timedatectl set-time ‘2015-11-20 16:14:50’ 🡺 to set date n time

timedatectl set-ntp true 🡺 to start automatic time synchronization with a remote NTP server

sendmail 🡺

* send and receive mail
* files : /etc/mail/sendmail.mc
* /etc/mail/sendmail.cf
* /etc/mail
* service: systemctl restart sendmail
* mail -s “ subject here” [nihar.unix@gmail.com](mailto:nihar.unix@gmail.com)

webserver (httpd) 🡺

* purpose of webserver is to serve webpages
* service or package name : httpd
* files: /etc/http/conf/httpd.conf
* /var/www/html/index.html
* service : systemctl restart/enable httpd
* log file : /var/log/httpd

Central Logger (RSYSLOG) 🡺

* Generate logs or collect logs from other servers. it is a central ogger where logs of 1000s of servers can be kept and checked whenever required. no need to login to server.
* service /package name: rsyslog
* configuration file : /etc/rsyslog.conf
* service: systemctl restart rsyslog
* systemctl enable rsyslog

OS Harding/securing OS 🡺

User Account 🡺

* User account naming convention
* User account user IDs •
* User password policies – chage –l username chage –help /etc/shadow vi /etc/login.defs
* Disable old password cd /etc/pam.d/system-auth
* User or service account files and directories permission

Remove unwanted packages 🡺

stop unused services 🡺

* List all running services
* systemctl (List only running service)
* systemctl –a (List every service running or not)
* telnet, ftp, NFS etc.

check on listening ports 🡺

* netstat -tunlp

secure ssh configuration 🡺

* disable direct root login
* change ssh port

enable firewall (iptables/firewalld) 🡺

* Older version = iptables
* New version = firewalld
* firewall-config (GUI)
* firewall-cmd
* iptables
* older version = /etc/sysconfig/iptables-config
* new version = /etc/firewalld/

enable SELinux🡺

* Security-Enhanced Linux (SELinux) is a security architecture integrated into the 2.6.x kernel using the Linux Security Modules (LSM). It is a project of the United States National Security Agency (NSA) and the SELinux community. SELinux integration into Red Hat Enterprise Linux was a joint effort between the NSA and Red Hat.
* SELinux defines the access and transition rights of every user, application, process, and file on the system
* /etc/sysconfig/selinux
* enforcing — The SELinux security policy is enforced.
* permissive — The SELinux system prints warnings but does not enforce policy. This is useful for debugging and troubleshooting purposes.
* disabled — SELinux is fully disabled. SELinux hooks are disengaged from the kernel and the pseudo-file system is unregistered.
* Commands = sestatus
* Find status of a file = stat filename

Other commands = chcon, checkpolicy, newrole, getsebool, setsebool, fixfiles, semanage Documentation attached within the hand-out section

change listening services port numbers 🡺

keep your OS up to date (Patching) 🡺

OPEN LDAP Installation 🡺

* what is openLDAP
* openLdap service 🡺 slapd
* start /top service 🡺 systemctl start/enable slapd
* config file : /etc/openldap/slapd.d

Trace Network Traffic (traceroute) 🡺

* traceroute command is used in Linux to map the journey that a packet of infomrtaion undertakes from its source to its destination. one use for the traceroute is to locate when data loss occurs through out a network, which could signify a node that’s down.
* because each hop in the record reflects a new server or router between the originating pc and the intended target, reviewing the results of a traceroute scan also lets you identify slow points that may adversely affect your network traffic.
* ex: traceroute google.com

**Configuring SSH 🡺**

port : 22

* ssh is secure communication through hosts and its encrypted. but there should be some additional configuration can be done to make it more secure.
* following are the most common config as admin should take care to secure ssh.
* configure idle timeout interval: avoid having an unattended ssh session , you can see an idle timeout interval
* become root
* edit your /etc/ssh/sshd\_config file and add the following line
* clinetaliveinterval 600
* clientAliveCountMax 0
* #systemctl restart sshd

the idele timeout interval you are setting is in seconds (600 secs = 10 mins). once the interval has passed, the idel user will be automatically logged out.

Cockpit – Linux webbased admin 🡺

* cockpit is a server administration tool sponsored by red hat, focused on providing a modern looking and user friendly interface to manage and administer servers.
* cockpit is the easy to use integrated and open web based interface for your servers.
* the application is available in most of the linux distributions such as centos, redhat, ubuntu and fedora
* it is installed in redhat 8 by default and it is optional in version 7
* it can monitor system resourcses , add or remove accounts, monitor system usage, sut down the system and perform quite a few other tasks all through a very accessible web connection.

**SHELL SCRIPTING**

**1. Understanding Shebang:**

Every Bash script should start with a shebang (#!) line, specifying the path to the Bash interpreter. This is typically #!/bin/bash. For example:

#!/bin/bash

**2. Variables:**

Variables store data. They don't have a specific data type, and you can assign any value to them.

# Variable assignment

NAME="John"

AGE=25

**# Accessing variables**

echo "Hello, $NAME! You are $AGE years old."

**3. User Input:**

Use the read command to get input from the user.

echo -n "Enter your name: "

read NAME

echo "Hello, $NAME!"

**4. Arithmetic Operations:**

Use the `(( ))` construct for arithmetic operations.

result=$((5 + 3))

echo "5 + 3 = $result"

**5. Conditions:**

Use if, elif, and else for conditional statements.

if [ "$AGE" -ge 18 ]; then

echo "You are an adult."

else

echo "You are a minor."

fi

**6. Loops:**

Use for and while loops for repetitive tasks.

Simple For Loop

A simple for loop in bash has the following structure:

**for variable in list**

do

# Commands to be repeated for each value in the list

done

Example:

#!/bin/bash

# Simple for loop to print numbers from 1 to 5

for number in 1 2 3 4 5

do

echo "Number: $number"

done

Output:

Number: 1

Number: 2

Number: 3

Number: 4

Number: 5

**For Loop with a Range**

You can use the {start..end} notation to create a range of values.

Example:

#!/bin/bash

# For loop with a range to print numbers from 1 to 5

for number in {1..5}

do

echo "Number: $number"

done

Output:

Number: 1

Number: 2

Number: 3

Number: 4

Number: 5

**For Loop with Variables**

You can also use a variable as the list in a for loop.

Example:

#!/bin/bash

# For loop with a variable as a list

fruits="apple orange banana"

for fruit in $fruits

do

echo "Fruit: $fruit"

done

Output:

makefile

Copy code

Fruit: apple

Fruit: orange

Fruit: banana

**For Loop with Command Substitution**

You can use command substitution to generate a list dynamically.

Example:

#!/bin/bash

# For loop with command substitution to iterate over files in the current directory

for file in $(ls)

do

echo "File: $file"

done

Output:

makefile

File: example.sh

File: otherfile.txt

File: some\_directory

# ... and so on

**# While loop**

A while loop in Bash scripting is used to repeatedly execute a block of code as long as a certain condition is true. The basic syntax of a while loop in Bash is as follows:

while [ condition ]; do

# code to be executed

done

Here's a simple example to illustrate how a while loop works in Bash:

#!/bin/bash

# Initialize a counter

counter=1

# Set the condition for the while loop (loop as long as counter is less than or equal to 5)

while [ $counter -le 5 ]; do

echo "This is iteration number $counter"

# Increment the counter

((counter++))

done

echo "Loop complete!"

Explanation:

The script initializes a variable counter with a value of 1.

The while loop is set up to continue as long as the condition [ $counter -le 5 ] is true, meaning "counter is less than or equal to 5."

Inside the loop, it prints a message indicating the current iteration and increments the counter using ((counter++)).

The loop will execute five times (for values of counter from 1 to 5), printing messages for each iteration.

Finally, it prints "Loop complete!" after the loop has finished.

A computer screen shot of text

Description automatically generated

**7. Functions:**

Define functions to organize your code.

greet() {

echo "Hello, $1!"

}

greet "Alice"

**8. Command Substitution:**

Capture the output of a command and use it as a variable.

current\_date=$(date)

echo "Current date is $current\_date"

**9. Positional Parameters:**

Access command-line arguments using $1, $2, etc.

#!/bin/bash

# Access command-line arguments

echo "Script name: $0"

echo "First argument: $1"

echo "Second argument: $2"

**10. File Operations:**

Use commands like touch, cp, mv, and rm for file operations.

# Create a file

touch myfile.txt

# Copy a file

cp myfile.txt newfile.txt

# Move a file

mv newfile.txt ~/Documents/

# Remove a file

rm ~/Documents/newfile.txt

**11. Arrays:**

Arrays store multiple values in a single variable.

# Array declaration

fruits=("Apple" "Banana" "Orange")

# Accessing elements

echo "First fruit: ${fruits[0]}"

echo "All fruits: ${fruits[@]}"

**12. Error Handling:**

Use set -e to exit the script if any command returns a non-zero status.

#!/bin/bash

set -e

#!/bin/bash

# Function to handle errors

handle\_error() {

echo "Error occurred!"

exit 1

}

**# Trap the error signal**

trap 'handle\_error' ERR

# Example: Force an error

echo "Before error"

ls non-existent-directory

echo "After error"

**13. Comments:**

Add comments to explain your code.

# This is a comment

**14. Special Variables:**

$0: Name of the script.

$1, $2, ...: Positional parameters.

$#: Number of parameters.

$?: Exit status of the last command.

Example Script:

Here's a simple example combining these concepts:

#!/bin/bash

# Get user input

echo -n "Enter your name: "

read NAME

# Check if the user is an adult

if [ "$AGE" -ge 18 ]; then

echo "You are an adult."

else

echo "You are a minor."

fi

# Function to greet the user

greet() {

echo "Hello, $1!"

}

# Call the function

greet "$NAME"

Save this script to a file (e.g., myscript.sh), make it executable (chmod +x myscript.sh), and run it (./myscript.sh).

**15. String Manipulation:**

Bash provides various string manipulation features.

string="Hello, World!"

# Length of a string

length=${#string}

echo "Length of the string: $length"

# Substring

substring=${string:0:5}

echo "Substring: $substring"

# Concatenation

new\_string="$string - Welcome!"

echo "$new\_string"

Realtime Shell script example :

* **shell script to execute sql query**

#!/bin/bash

# Database connection details

DB\_USER="your\_username"

DB\_PASSWORD="your\_password"

DB\_HOST="your\_host"

DB\_PORT="your\_port"

DB\_SID="your\_sid"

# SQL SELECT query

SELECT\_QUERY="SELECT \* FROM your\_table;"

# Construct the connection string

CONNECTION\_STRING="${DB\_USER}/${DB\_PASSWORD}@//${DB\_HOST}:${DB\_PORT}/${DB\_SID}"

# Execute the SELECT query using sqlplus

result=$(echo "$SELECT\_QUERY" | sqlplus -S "$CONNECTION\_STRING")

# Check for errors

if [ $? -eq 0 ]; then

# Print the result

echo "Query result:"

echo "$result"

else

# Print an error message

echo "Error executing the SELECT query."

Fi

* **shell script to connect to servers and execute one script**

#!/bin/bash

# Server details

remote\_server="server2"

remote\_user="your\_remote\_username"

remote\_script\_path="/path/to/remote\_script.sh"

# SSH connection and script execution

ssh "$remote\_user@$remote\_server" "bash -s" < "$remote\_script\_path"

# Optional: Check the exit status of the remote script

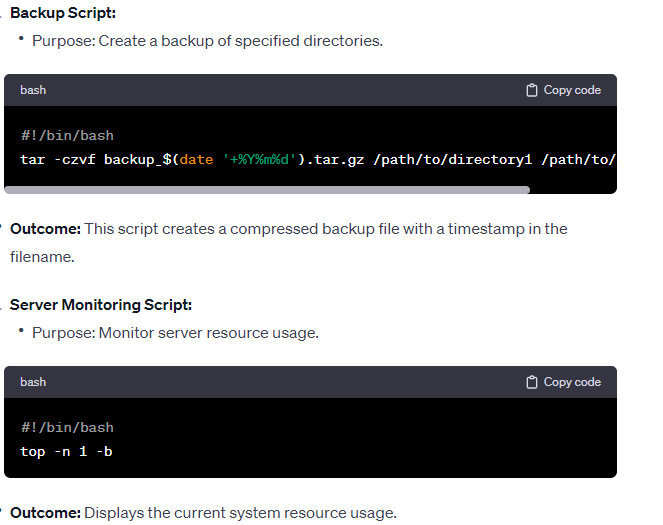
if [ $? -eq 0 ]; then

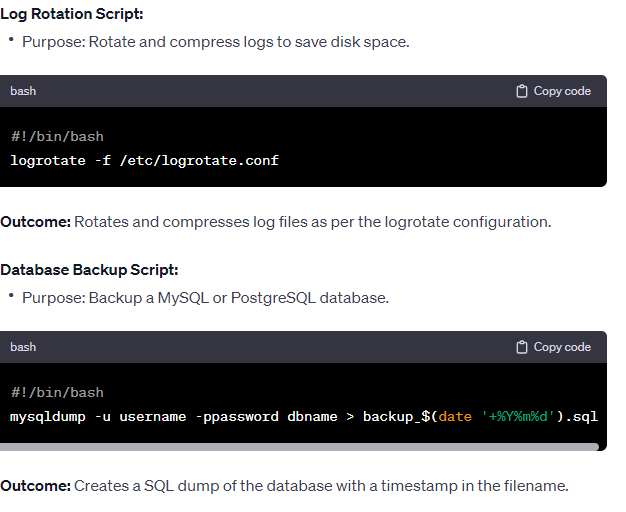
echo "Script executed successfully on $remote\_server."

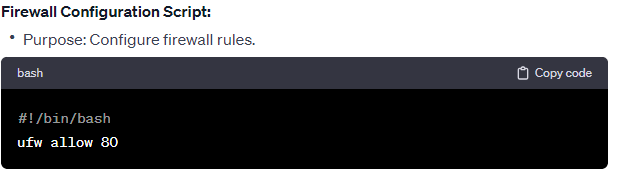
else

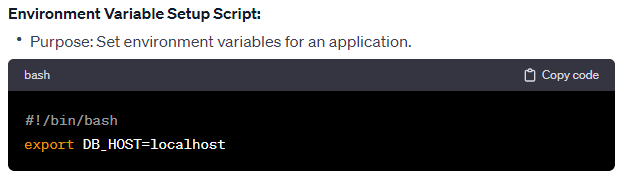
echo "Error executing script on $remote\_server."

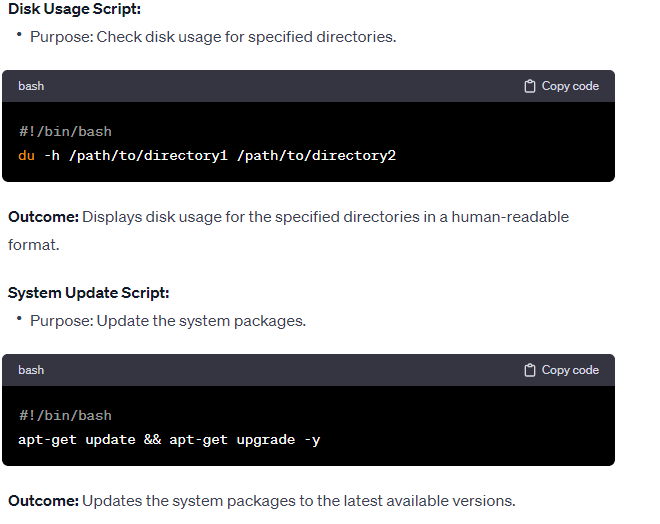
Fi

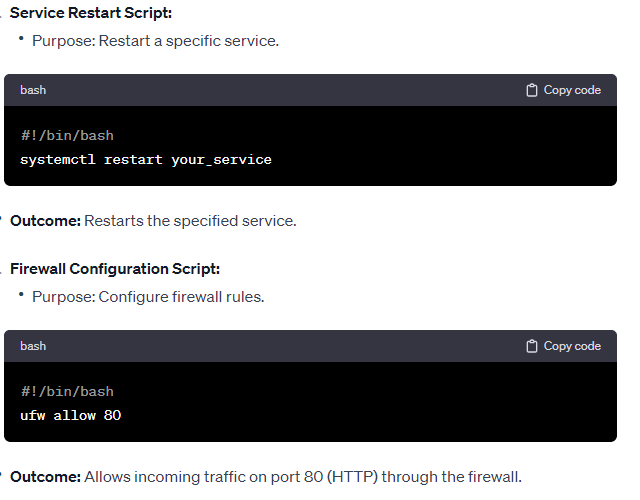




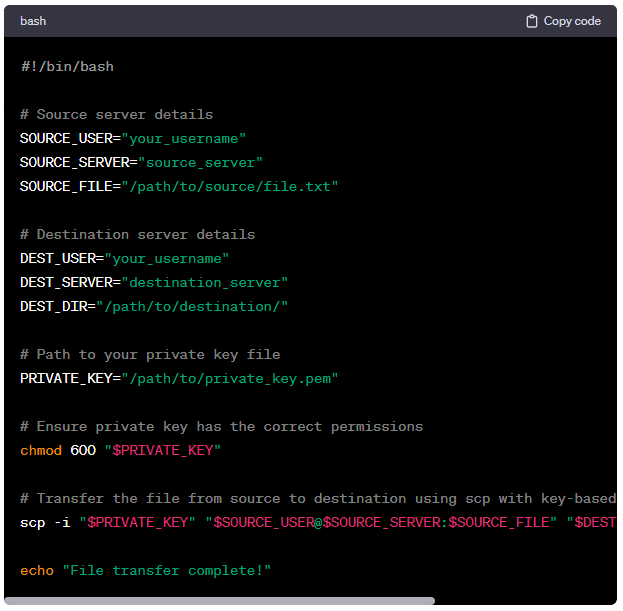




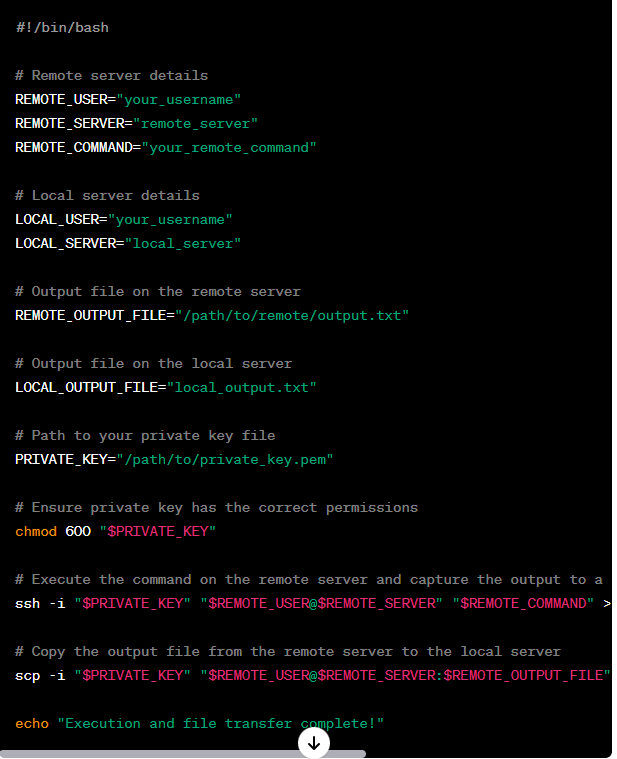




* Copy files from one server to another server using SCP and private key



* **PRIVATE\_KEY**: Specify the path to your private key file.
* **chmod 600 "$PRIVATE\_KEY"**: This command ensures that the private key has the correct permissions (read and write only by the owner).
* **scp -i "$PRIVATE\_KEY"**: This uses the private key for authentication during the **scp** command.



In this script:

REMOTE\_COMMAND: Replace this with the command you want to execute on the remote server.

REMOTE\_OUTPUT\_FILE: Specifies the file on the remote server to capture the command output.

LOCAL\_OUTPUT\_FILE: Specifies the local file to store the output after copying it from the remote server.

ssh -i "$PRIVATE\_KEY" "$REMOTE\_USER@$REMOTE\_SERVER" "$REMOTE\_COMMAND" > "$REMOTE\_OUTPUT\_FILE": Executes the command on the remote server and redirects the output to the specified file.

scp -i "$PRIVATE\_KEY" "$REMOTE\_USER@$REMOTE\_SERVER:$REMOTE\_OUTPUT\_FILE" "$LOCAL\_USER@$LOCAL\_SERVER:$LOCAL\_OUTPUT\_FILE": Copies the remote output file to the local server.

**LINUX QUESTIONS :**

**How do you convert sql table data into a flat file ?**

exp your\_username/your\_password@your\_database\_name tables=your\_table\_name file=output\_file.dmp

**Add $ at the end of the every line ?**

sed 's/$/$/' input\_file > output\_file

This command reads the contents of input\_file, adds a $ at the end of each line, and then writes the modified content to output\_file. Adjust the file names as needed.

sed -i 's/$/$/' your\_file 🡺 if you want to modify the file in place (i.e., overwrite the original file), you can use the -i option with sed:

**Duplicate lines of a file ?**

This command sorts the lines in the file and then uses uniq -d to display only the duplicate lines.

sort your\_file | uniq -d

This will display the duplicate lines along with the count of occurrences.

sort your\_file | uniq -cd

If you want to keep the original order of the lines and just find duplicates, you can use awk:

awk '!seen[$0]++' your\_file

**Delete 15 days old files ?**

Find /path/to/your/directory -type file -mtime +15 -delete

**List of users having access to server ?**

cut -d: -f1 /etc/passwd

If you want to list only the currently logged-in users, you can use the who or w command:

who 🡺These commands show information about the users currently logged into the system.

w 🡺These commands show information about the users currently logged into the system

**How to open a new port ?**

Check if firewalld is installed:

sudo systemctl status firewalld

If it's not installed, you can install it using:

sudo yum install firewalld

Enable and start the firewalld service:

sudo systemctl enable firewalld

sudo systemctl start firewalld

Open the desired port (replace PORT\_NUMBER with the actual port you want to open):

sudo firewall-cmd --zone=public --add-port=PORT\_NUMBER/tcp --permanent

If the port uses UDP, you can replace tcp with udp.The --permanent option makes the rule persistent across reboots.

Reload the firewall for the changes to take effect:

sudo firewall-cmd --reload

Check the status to verify that the new rule is added.This command will display a list of allowed ports.

sudo firewall-cmd --list-ports

**What is TCP/UDP ?**

TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are two of the most common transport layer protocols used in networking, including Linux systems. Both are used to transmit data between applications over a network, but they have some key differences.

**TCP (Transmission Control Protocol):**

Connection-Oriented:

TCP is a connection-oriented protocol. Before data transfer begins, a connection is established between the sender and the receiver.

It ensures reliable, ordered, and error-checked delivery of data.

Reliability:

TCP provides a reliable and error-checked communication stream. It guarantees that data sent by one end is received in the same order by the other end, and it manages retransmission of lost packets.

Flow Control:

TCP performs flow control to prevent a fast sender from overwhelming a slow receiver. It uses mechanisms like windowing and acknowledgments to control the flow of data.

Usage:

Commonly used for applications where reliable and ordered delivery of data is crucial, such as web browsing, email, file transfers (FTP), and remote shell access (SSH).

**UDP (User Datagram Protocol):**

Connectionless:

UDP is a connectionless protocol. It does not establish a connection before sending data; each packet is sent independently of the others.

There is no guarantee of delivery, and packets may arrive out of order or not at all.

Reliability:

UDP does not provide reliability mechanisms like acknowledgment, retransmission, or flow control. It's a "best-effort" protocol, suitable for applications where occasional packet loss is acceptable.

Low Overhead:

UDP has lower overhead compared to TCP because it lacks the additional mechanisms for reliability. This can make it faster for certain types of applications.

Usage:

Commonly used in scenarios where low latency and quick data transmission are more critical than ensuring every packet is delivered, such as real-time applications like online gaming, video streaming, and DNS (Domain Name System) queries.

In summary, TCP is focused on providing reliable, ordered, and error-checked delivery of data, making it suitable for applications that require these assurances. UDP, on the other hand, sacrifices some reliability for lower latency and is more appropriate for applications where occasional packet loss can be tolerated. The choice between TCP and UDP depends on the specific requirements of the application.

**How do you change ownership of a file in Linux?**

To change the ownership of a file in Linux, you can use the chown command. The basic syntax is as follows:

sudo chown new\_owner:new\_group file\_path

sudo chown newuser:newgroup /home/user/documents/example.txt

Here's a breakdown of the command:

sudo: This is used to execute the command with superuser privileges, which is often required when changing file ownership.

chown: Stands for "change owner."

new\_owner: The new owner you want to assign to the file.

new\_group: The new group you want to assign to the file. This is optional, and if not specified, the file will be assigned the default group of the new owner.

file\_path: The path to the file whose ownership you want to change.

Explain the meaning of the three permission sets in a Linux file.

**How can you recursively copy files and directories in Linux?**

Recursively copy a directory to another location:

cp -r /path/to/source /path/to/destination

This command copies the entire content of the source directory to the destination directory.

Recursively copy a directory with its contents:

cp -r /path/to/source/. /path/to/destination

Note the dot (.) after the source directory. This ensures that the contents of the source directory are copied to the destination rather than copying the source directory itself.

Recursively copy a directory to a new directory (creating the destination if it doesn't exist):

cp -r /path/to/source /path/to/new\_destination

If new\_destination doesn't exist, cp will create it and copy the content of source into it.

**How do you check if a file exists in a directory using a shell script?**

#!/bin/bash

file\_path="/path/to/directory/file.txt"

if [ -e "$file\_path" ]; then

echo "File exists!"

else

echo "File does not exist."

fi

**What is the purpose of the sudo command?**

The sudo command in Linux stands for "SuperUser DO" and is used to execute commands with elevated privileges or as another user, usually the superuser or root. The primary purpose of sudo is to enable authorized users to perform administrative tasks without logging in as the root user.

Key purposes and features of the sudo command include:

Administrative Privileges:

sudo allows a permitted user to execute a command as the superuser or another user, according to the security policy configured in the /etc/sudoers file.

Limited Privileges:

sudo provides a way to grant specific users or groups specific permissions to execute certain commands with elevated privileges, without giving them full root access.

Auditability:

sudo logs each command run using it, providing an audit trail that can be reviewed later. This helps in tracking who executed which commands and when.

Security:

The use of sudo enhances security by reducing the need for users to log in directly as the root user. It allows users to perform administrative tasks temporarily, minimizing the risk associated with running commands with elevated privileges.

Command-Level Access Control:

The /etc/sudoers file allows system administrators to specify precisely which commands a user is allowed to execute with sudo and under what conditions.

Password Authentication:

By default, using sudo requires the user to enter their own password, providing an additional layer of security.

Example Usage:

sudo command\_to\_execute

For instance, to install a package using apt on Debian-based systems:

sudo apt-get install package\_name

Or, to edit a system configuration file using a text editor:

sudo nano /etc/config\_file.conf

**How do you schedule a task to run periodically using cron?**

crontab -e

Minute Hour Day Month Day\_of\_week Command\_to\_be\_executed

Minute (0 - 59): The minute when the cron job will run.

Hour (0 - 23): The hour when the cron job will run.

Day (1 - 31): The day of the month when the cron job will run.

Month (1 - 12): The month when the cron job will run.

Day\_of\_week (0 - 6): Sunday to Saturday (0 and 7 both represent Sunday).

Run a script every day at 2:30 AM:

30 2 \* \* \* /path/to/your/script.sh

Run a command every Monday at 9:00 PM:

0 21 \* \* 1 /path/to/your/command

Explain the purpose of the /etc/hosts file.

The /etc/hosts file is a plain-text file on Unix-like operating systems, including Linux, that serves the purpose of mapping IP addresses to hostnames. This file is used as a local DNS (Domain Name System) resolver for hostname resolution, allowing the system to translate human-readable hostnames into IP addresses.

The typical format of an entry in the /etc/hosts file is as follows:

<IP\_address> <hostname> <alias1> <alias2> ... <aliasN>

127.0.0.1 localhost mymachine

**What is the function of the /etc/resolv.conf file?**

The /etc/resolv.conf file in Linux is a configuration file used by the system's resolver to determine the DNS (Domain Name System) settings. It contains information about the DNS servers, search domains, and other resolver options that the system uses to resolve domain names to IP addresses and vice versa.

Key Functions of /etc/resolv.conf:

DNS Server Configuration:

The primary purpose of the file is to specify the IP addresses of the DNS servers that the system should query for domain name resolution. These servers are used to translate human-readable domain names (like www.example.com) into IP addresses.

Search Domains:

The search directive in /etc/resolv.conf specifies a list of domain names that the resolver appends to single-component hostnames before attempting to resolve them. For example, if the search domain is set to example.com, a request for web will be treated as web.example.com.

search example.com

Name Server Timeouts and Retries:

The file may include directives for controlling the timeout and retry behavior of the resolver when querying DNS servers. For example:

options timeout:2

options attempts:3

This would set a timeout of 2 seconds for each DNS query and allow up to 3 attempts before giving up.

Example /etc/resolv.conf File:

# Sample /etc/resolv.conf

nameserver 8.8.8.8

nameserver 8.8.4.4

search example.com

In this example, the file specifies two DNS servers (Google's public DNS servers, in this case) and a search domain of example.com.

Note:

Dynamic Changes:

Some Linux distributions dynamically generate the /etc/resolv.conf file based on network configuration. Changes made directly to the file might be overwritten. In such cases, it's advisable to configure DNS settings through the system's network configuration tools.

DHCP:

Systems that obtain their network configurations via DHCP (Dynamic Host Configuration Protocol) may have their /etc/resolv.conf file updated automatically by the DHCP client.

Security Considerations:

As /etc/resolv.conf may contain sensitive information (such as DNS server addresses), it should be protected against unauthorized access.

**How do you add a new user in Linux?**

sudo useradd -m -s /bin/bash username

Explanation of options:

-m: Create the user's home directory if it doesn't exist.

-s /bin/bash: Set the user's default shell to Bash. You can replace /bin/bash with the preferred shell (e.g., /bin/zsh).

Set a password for the new user:

sudo passwd username ==>You will be prompted to enter and confirm the new password.

Additional Tips:

If you want the new user to have administrative privileges (sudo access), you can add the user to the sudo group:

sudo usermod -aG sudo username

You can customize the home directory and other user properties by providing additional options to the useradd command.

If you prefer a more interactive approach, you can use the adduser command, which provides a series of prompts for user information:

sudo adduser username

This command will guide you through the process of setting up the user, including the home directory, password, and other details.

Verify the New User:

You can check if the new user has been successfully added by listing the contents of the /etc/passwd file or by using the id command:

**Explain the significance of the /etc/fstab file.**

The /etc/fstab file in Linux is a critical configuration file that defines how storage devices, such as hard drives and partitions, should be mounted and accessed by the operating system. The file contains information about file systems, their mount points, options, and other parameters required for proper disk management.

# /etc/fstab

UUID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx / ext4 defaults 0 1

/dev/sdb1 /data ext4 defaults 0 2

**How do you restart a service in Linux?**

sudo systemctl restart service\_name

sudo systemctl restart apache2

sudo systemctl restart nginx

sudo systemctl stop service\_name

sudo systemctl start service\_name

**How do you check the system logs in Linux?**

To view the system log on Debian/Ubuntu-based systems

cat /var/log/syslog

Red Hat/Fedora-based systems, you can use similar commands for the /var/log/messages file

cat /var/log/messages

**What is the purpose of the top command?**

top provides a high-level overview of system statistics, including the total number of processes, system uptime, load averages, and the percentage of CPU and memory usage.

How do you check the available memory on a Linux system?

**Explain the use of the ping command.**

The ping command is a network utility used in Linux and other operating systems to test the reachability of a host on an Internet Protocol (IP) network. It also measures the round-trip time for messages sent from the originating host to a destination host and back. The primary purpose of ping is to troubleshoot and diagnose network connectivity issues.

ping www.example.com

Common Options:

-c count: Specifies the number of packets to be sent. For example, -c 4 sends four packets and then stops.

ping -c 4 www.example.com

-s packet\_size: Sets the size of the packets to be sent. The default is often 56 bytes.

ping -s 100 www.example.com

-i interval: Specifies the interval between sending each packet in seconds.

ping -i 2 www.example.com

-W timeout: Sets the timeout for waiting for a reply, in seconds.

ping -W 5 www.example.com

**How ping Works:**

Sending Echo Requests:

ping sends ICMP (Internet Control Message Protocol) Echo Request packets to the specified host or IP address.

Receiving Echo Replies:

If the destination host is reachable and responsive, it sends ICMP Echo Reply packets back to the source.

Round-Trip Time (RTT) Calculation:

ping measures the round-trip time, which is the time taken for a packet to travel from the source to the destination and back.

Use Cases:

ping is commonly used to diagnose network connectivity issues. If a host is not reachable, it could indicate a problem with the network or the destination host.

It's often used to check whether a remote host is online and responding to network requests.

By analyzing the round-trip time, users can get an idea of the latency between the source and destination hosts.

Tips:

Continuous Ping:

If you want ping to continue sending packets until manually stopped, you can use the -t option.

ping -t www.example.com

IPv6:

For IPv6 addresses, use the -6 option.

ping -6 ipv6.example.com

**How do you find the MAC address of a network interface?**

To find the MAC address (Media Access Control address) of a network interface in Linux, you can use the ifconfig or ip command. Here are the steps using both commands:

ifconfig

ip link show

**What is the purpose of the ifconfig command?**

The ifconfig command in Linux is a network utility used to configure, display, and manage network interfaces on a system. While ifconfig has been a widely used tool for many years, it is now considered somewhat deprecated in favor of newer tools like ip from the iproute2 package. However, ifconfig is still available on many systems and remains familiar to many users.

**Viewing Network Interface Configuration:**

ifconfig displays information about all network interfaces currently available on the system. This includes details such as IP addresses, MAC addresses, network status (up or down), and various statistics related to data transfer.

ifconfig

Enabling or Disabling Network Interfaces:

ifconfig can be used to enable (up) or disable (down) a network interface. This is useful for bringing an interface online or taking it offline.

ifconfig eth0 up

ifconfig eth0 down

Setting IP Addresses:

You can use ifconfig to assign an IP address and netmask to a network interface.

ifconfig eth0 192.168.1.2 netmask 255.255.255.0

Changing MAC Addresses:

ifconfig allows you to change the MAC address of a network interface.

ifconfig eth0 hw ether 00:11:22:33:44:55

Note: Changing the MAC address is sometimes referred to as "MAC spoofing" and should be used responsibly and within legal and ethical boundaries.

Displaying Only Specific Interface Information:

You can use ifconfig with a specific interface name to display information only for that interface.

ifconfig eth0

**How do you add a route in Linux?**

In Linux, you can add a route using the ip route command. The ip route command allows you to manipulate the kernel's IP routing table. Here's the basic syntax to add a route:

sudo ip route add destination\_network/mask via gateway

destination\_network/mask: The network address and subnet mask of the destination network.

via gateway: The IP address of the next-hop gateway through which the destination network is reachable.

sudo ip route add 192.168.2.0/24 via 192.168.1.1

192.168.2.0/24 is the destination network.

192.168.1.1 is the next-hop gateway through which the 192.168.2.0/24 network is reachable.

**Explain the function of the netstat command.**

The netstat command in Linux is a network utility tool that provides information about network connections, routing tables, interface statistics, masquerade connections, and more. It is used to troubleshoot network-related issues, monitor network performance, and gather information about network connections on a system. The functionality of netstat can vary slightly depending on the operating system and version.

List All Connections:

netstat -a

Display TCP Connections:

netstat -t

Display UDP Connections:

netstat -u

Display Listening Sockets:

netstat -l

Display Routing Table:

netstat -r

Display Summary Statistics:

netstat -s

Display Process ID and Program Name:

netstat -p

Display numerical addresses and ports instead of resolving them to hostnames and service names:

netstat -n

**How do you install a package using apt in Debian-based systems?**

sudo apt install package\_name

You can install multiple packages in a single command by listing them with spaces between the package names.

sudo apt install package1 package2 package3

You can install a specific version of a package by specifying the version number.

sudo apt install package\_name=version\_number

To skip the confirmation prompt, you can use the -y option.

sudo apt install -y package\_name

Before installing a package, it's a good practice to update the local package information to ensure you are installing the latest version.Always run sudo apt update before installing a new package to ensure that the local package information is up to date.

sudo apt update

To upgrade all installed packages to the latest versions, you can use the following commands:

sudo apt update

sudo apt upgrade

**Explain the purpose of the yum command in Red Hat-based systems.**

The yum command in Red Hat-based Linux systems is a package management tool used for installing, updating, removing, and managing software packages. It is specifically designed for systems that use the Red Hat Package Manager (RPM) as the package format, and it simplifies the process of handling dependencies and software installations.

sudo yum [options] [command] [package(s)]

Install a Package:

sudo yum install package\_name

sudo yum install htop

Remove a Package:

sudo yum remove package\_name

sudo yum remove htop

Update Packages:

sudo yum update

sudo yum update

Search for Packages:

yum search keyword

yum search nginx

List Installed Packages:

yum list installed

**How do you list all installed packages on a Linux system?**

1. For Debian-based Systems (e.g., Ubuntu):

dpkg -l | grep '^ii'

This command lists all installed packages and filters the output to show only the installed ones (status starting with 'ii').

2. For Red Hat-based Systems (e.g., CentOS, Fedora):

yum list installed

This command lists all installed packages using yum.

3. For Systems with the dnf Package Manager (Recent Fedora, RHEL):

dnf list installed

On systems with dnf, this command lists all installed packages.

4. For Systems with the zypper Package Manager (openSUSE):

zypper search --installed-only

This command lists all installed packages using zypper.

5. For Systems with the pacman Package Manager (Arch Linux):

pacman -Q

This command lists all installed packages using pacman.

6. For Systems with the apk Package Manager (Alpine Linux):

apk info -vv | grep -e '^-' | awk '{print $1}'

This command lists all installed packages using apk.

7. For Systems with the rpm Package Manager (General RPM-based Systems):

rpm -qa

This command lists all installed packages using rpm.

**What is the purpose of the dpkg command?**

dpkg is primarily used for installing and removing software packages on Debian-based systems. It handles the installation process at a lower level by directly interacting with the package files.

Install a Package:

sudo dpkg -i package\_file.deb

This command installs a Debian package from a .deb file.

Remove a Package:

sudo dpkg -r package\_name

This command removes a previously installed package. It leaves configuration files in place.

Remove a Package and Its Configuration Files:

sudo dpkg -P package\_name

This command removes a package along with its configuration files.

List Installed Packages:

dpkg -l

This command lists all installed packages along with their version numbers and descriptions.

Query Package Information:

dpkg -s package\_name

This command provides detailed information about a specific installed package.

Check for Broken Packages:

sudo dpkg --audit

This command checks for broken dependencies and suggests solutions.

Reconfigure an Installed Package:

sudo dpkg-reconfigure package\_name

This command allows you to reconfigure an installed package. It may prompt you with configuration options.

**How do you pass arguments to a shell script?**

When executing a shell script, you can pass arguments to the script from the command line. These arguments are accessible within the script and can be used for various purposes. Here's how you can pass arguments to a shell script:

1. Positional Parameters:

In a shell script, the command-line arguments are referred to as positional parameters. They are represented by $1, $2, $3, and so on, where $1 corresponds to the first argument, $2 to the second, and so forth.

Example:

Create a simple script named myscript.sh:

#!/bin/bash

echo "First argument: $1"

echo "Second argument: $2"

Run the script with arguments:

bash myscript.sh arg1 arg2

Output:

First argument: arg1

Second argument: arg2

2. Special Variables:

$0: Represents the name of the script itself.

$#: Represents the number of arguments passed to the script.

$@: Represents all the arguments passed to the script as an array.

Example:

#!/bin/bash

echo "Script name: $0"

echo "Number of arguments: $#"

echo "All arguments: $@"

Run the script with arguments:

bash myscript.sh arg1 arg2 arg3

Output:

Script name: myscript.sh

Number of arguments: 3

All arguments: arg1 arg2 arg3

3. Using shift to Iterate Over Arguments:

You can use the shift command to shift the positional parameters to the left, discarding the first argument. This is useful for iterating over all arguments in a script.

Example:

#!/bin/bash

echo "Script name: $0"

echo "Number of arguments: $#"

while [ $# -gt 0 ]; do

echo "Argument: $1"

shift

done

Run the script with arguments:

bash myscript.sh arg1 arg2 arg3

Output:

Script name: myscript.sh

Number of arguments: 3

Argument: arg1

Argument: arg2

Argument: arg3

Passing Named Arguments:

While shell scripts typically use positional parameters, you can also implement named or flag-based arguments by processing the arguments manually within the script.

Example:

#!/bin/bash

while [[ $# -gt 0 ]]; do

case "$1" in

-f|--file)

file="$2"

shift 2

;;

-d|--directory)

dir="$2"

shift 2

;;

\*)

echo "Unknown argument: $1"

exit 1

;;

esac

done

echo "File: $file"

echo "Directory: $dir"

Run the script with named arguments:

bash myscript.sh -f myfile.txt -d mydir

Output:

makefile

Copy code

File: myfile.txt

Directory: mydir

In this example, the script uses a while loop and a case statement to process named arguments. The arguments are passed in pairs, and the script extracts the values based on the specified flags (-f, --file, -d, --directory).

Explain the purpose of the echo command in shell scripting.

The echo command in shell scripting is used to display text or messages to the standard output (usually the terminal or console). It is a simple and frequently used command for printing information, variables, or other messages to the screen.

echo [option(s)] [string(s)]

The primary purpose of echo is to print text or messages to the terminal.

echo "Hello, World!"

Printing Variables:echo is often used to display the values of variables.

name="John"

echo "My name is $name"

This command prints "My name is John" to the terminal.

Newline Control:By default, echo appends a newline character after the output. This behavior can be modified using the -n option.

echo -n "Hello, "

echo "World!"

This prints "Hello, World!" without a newline in between.

Escape Characters:echo interprets escape characters, allowing for special formatting.

echo "This is a new line.\nThis is another line."

This command produces a multi-line output.

Redirecting Output:echo output can be redirected to a file using the > or >> operators.

echo "Hello, World!" > output.txt

This writes "Hello, World!" to a file named output.txt.

Example Usage:

#!/bin/bash

name="Alice"

age=25

echo "Name: $name"

echo "Age: $age years"

This script declares two variables (name and age) and uses echo to print their values to the terminal.

Note:

While echo is simple and widely used, it may not be suitable for all cases, especially when dealing with complex formatting or special characters. In such cases, the printf command is often preferred.

When working with echo and variables, it's a good practice to enclose variables in double quotes to handle cases where the variable contains spaces or special characters.echo "My name is $name" # Preferred

echo 'My name is $name' # Without variable expansion

**How do you use conditional statements in a shell script?**

In shell scripting, conditional statements are used to make decisions based on certain conditions. The most common conditional statements in shell scripts are the if, elif (else if), and else statements. These statements allow you to control the flow of your script based on whether certain conditions are true or false.

if [ condition ]; then

# Code to execute if the condition is true

fi

Example:

#!/bin/bash

# Check if a number is positive or negative

number=5

if [ $number -gt 0 ]; then

echo "The number is positive."

else

echo "The number is non-positive."

fi

In this example, the script checks whether the value of the variable number is greater than zero. If it is, it prints "The number is positive," otherwise, it prints "The number is non-positive."

elif Statement:

You can use elif to test multiple conditions in sequence.

#!/bin/bash

# Check if a number is positive, negative, or zero

number=-3

if [ $number -gt 0 ]; then

echo "The number is positive."

elif [ $number -lt 0 ]; then

echo "The number is negative."

else

echo "The number is zero."

fi

In this script, it checks if the number is positive, negative, or zero using elif for additional conditions.

Logical Operators:

You can use logical operators like -eq (equal), -ne (not equal), -lt (less than), -le (less than or equal), -gt (greater than), and -ge (greater than or equal) to create more complex conditions.

#!/bin/bash

# Check if a number is even or odd

number=6

if [ $((number % 2)) -eq 0 ]; then

echo "The number is even."

else

echo "The number is odd."

fi

In this example, it uses the modulus operator (%) to check if the number is even or odd.

Nested if Statements:

You can also nest if statements to handle more complex conditions.

#!/bin/bash

# Nested if statements to check multiple conditions

age=25

if [ $age -lt 18 ]; then

echo "You are a minor."

else

if [ $age -ge 18 ] && [ $age -lt 65 ]; then

echo "You are an adult."

else

echo "You are a senior."

fi

fi

In this script, it checks if a person is a minor, adult, or senior based on their age using nested if statements.

Case Statements:

The case statement is another way to handle multiple conditions more efficiently.

#!/bin/bash

# Using case statement to check the day of the week

day="Monday"

case $day in

"Monday")

echo "It's the start of the week."

;;

"Friday")

echo "It's almost the weekend!"

;;

\*)

echo "It's a regular day."

;;

esac

This script uses a case statement to check the day of the week and execute different code blocks based on the value of the day variable.

**What is the purpose of the awk command in shell scripting?**

The awk command in shell scripting is a powerful text processing tool that is used for pattern scanning and processing. It is particularly well-suited for handling structured text data, such as columns and rows, making it a valuable tool for tasks like data extraction, reporting, and transformation.

awk 'pattern { action }' input\_file

**How do you use loops in shell scripting?**

In shell scripting, loops are used to execute a set of commands repeatedly. The two main types of loops are the for loop and the while loop. Here are examples of how to use each type:

1. for Loop:

The for loop is often used when you have a predefined list of items to iterate over.

Syntax:

for variable in list

do

# Commands to be executed for each item in the list

done

Example:

#!/bin/bash

# Print numbers from 1 to 5 using a for loop

for i in {1..5}

do

echo $i

done

This script prints numbers from 1 to 5 using a for loop.

2. while Loop:

The while loop is used when you want to execute a set of commands as long as a specified condition is true.

Syntax:

while [ condition ]

do

# Commands to be executed as long as the condition is true

done

Example:

#!/bin/bash

# Print numbers from 1 to 5 using a while loop

counter=1

while [ $counter -le 5 ]

do

echo $counter

((counter++))

done

This script prints numbers from 1 to 5 using a while loop.

Loop Control Statements:

1. break:

The break statement is used to exit a loop prematurely.

# Example using break in a while loop

counter=1

while true

do

echo $counter

((counter++))

if [ $counter -gt 5 ]; then

break

fi

done

This script prints numbers using a while loop and breaks out of the loop when the counter exceeds 5.

2. continue:

The continue statement is used to skip the remaining commands in the loop and proceed to the next iteration.

# Example using continue in a for loop

for i in {1..5}

do

if [ $i -eq 3 ]; then

continue

fi

echo $i

done

This script prints numbers from 1 to 5 using a for loop but skips the iteration where i is equal to 3.

Example Use Case:

Suppose you want to print the squares of numbers from 1 to 5 using a for loop:

#!/bin/bash

# Print the squares of numbers from 1 to 5 using a for loop

for i in {1..5}

do

square=$((i \* i))

echo "Square of $i is $square"

done

This script calculates and prints the squares of numbers from 1 to 5 using a for loop.

**Explain the purpose of the ufw command.**

Simplified Firewall Configuration:

ufw provides a simplified and user-friendly syntax for configuring firewall rules, making it easier for users who may not be familiar with the intricacies of iptables.

Basic Firewall Operations:

It allows users to enable or disable the firewall, as well as configure default policies for incoming and outgoing traffic. These operations can be done with simple commands.

# Enable the firewall

sudo ufw enable

# Disable the firewall

sudo ufw disable

Configuring Rules:

Users can easily define rules to allow or deny specific types of traffic based on ports, services, or protocols.

# Allow SSH traffic

sudo ufw allow 22

# Allow HTTP traffic

sudo ufw allow 80/tcp

Managing Applications:

ufw supports application profiles, allowing users to enable or disable predefined sets of rules for specific applications.

# Allow OpenSSH

sudo ufw allow OpenSSH

Logging:

ufw provides logging options that allow users to log denied packets. This can be useful for monitoring and troubleshooting.

# Enable logging for denied packets

sudo ufw logging on

Users can easily check the status of the firewall, view rules, and get information about the configuration.

# Show status and rules

sudo ufw status verbose

Some Linux distributions integrate ufw with their package management systems, allowing for easy configuration of firewall rules when installing or removing software packages.

Example Usage:

# Enable the firewall

sudo ufw enable

# Allow SSH traffic

sudo ufw allow 22

# Allow HTTP and HTTPS traffic

sudo ufw allow 80/tcp

sudo ufw allow 443/tcp

# Show the status and rules

sudo ufw status verbose

In this example, the firewall is enabled, and rules are defined to allow SSH, HTTP, and HTTPS traffic. The status command provides information about the current configuration.

Important Notes:

ufw is typically pre-installed on Ubuntu-based systems, but it may need to be installed on other distributions.

**How do you secure SSH access to a Linux server?**

Strong Passwords:

If you use password authentication, ensure that users have strong, unique passwords.

SSH Keys:

Preferably, use SSH key authentication instead of passwords. SSH keys provide a more secure and convenient way to authenticate users.

**What is the purpose of the /etc/sudoers file?**

The /etc/sudoers file is a crucial configuration file on Unix-like operating systems, including Linux. It is used to define the sudo (substitute user do) permissions and policies, allowing authorized users to execute commands with the privileges of another user, usually the superuser (root).

Always use the visudo command to edit the /etc/sudoers file.

How do you check for open ports on a Linux server?

# Display listening ports

sudo netstat -tuln

# Display all open ports and associated processes

sudo netstat -tulpn

Advanced Topics:

Explain the use of the sed command.

The sed command, which stands for "stream editor," is a powerful and versatile text-processing tool in Unix-like operating systems, including Linux. sed is used for performing basic text transformations on an input stream (a file or input from a pipeline) or a specified file. It is particularly useful for making changes to a text stream or file based on specified patterns or rules.

Substitution (s/old/new/g):Replace occurrences of a pattern with a new value.

# Replace "apple" with "orange" in a file

sed 's/apple/orange/g' filename

Print Lines (p):Print lines that match a specified pattern.

# Print lines containing "pattern" in a file

sed -n '/pattern/p' filename

Delete Lines (d):Delete lines that match a specified pattern.

# Delete lines containing "pattern" in a file

sed '/pattern/d' filename

Insert Lines (i and a):Insert new lines before (i) or after (a) a specified line.

# Insert "New line" before lines containing "pattern" in a file

sed '/pattern/i\New line' filename

Replace Lines (c):Replace entire lines that match a pattern with new text.

# Replace lines containing "pattern" with "New text" in a file

sed '/pattern/c\New text' filename

Selective Printing (-n):Suppress automatic printing and selectively print lines.

# Print only lines between line 3 and line 7 in a file

sed -n '3,7p' filename

Multiple Commands:Execute multiple sed commands sequentially.

# Replace "apple" with "orange" and print lines containing "fruit" in a file

sed -e 's/apple/orange/g' -e '/fruit/p' filename

In-Place Editing:sed can be used for in-place editing, meaning it modifies the file directly.

# Replace "old" with "new" in a file and save changes

sed -i 's/old/new/g' filename

Examples:

Replace Tabs with Spaces:

sed 's/\t/ /g' filename

Remove Blank Lines:

sed '/^$/d' filename

Extract IP Addresses from Log:

sed -nE 's/.\*([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+).\*/\1/p' logfile

Print Line Numbers:

sed -n -e '/pattern/=' -e '/pattern/p' filename

Important Notes:

sed operates on a line-by-line basis, making it well-suited for processing text streams.

The use of regular expressions is common in sed patterns.

sed commands are case-sensitive by default.

The -i option for in-place editing can be risky; ensure you have a backup of the file or use the -i.bak option to create a backup.

sed is a powerful tool for text manipulation and is widely used in scripts and one-liners for tasks such as search and replace, text extraction, and formatting. Understanding regular expressions and sed commands enhances its effectiveness in text processing tasks.

**What is the function of the awk command?**

The `awk` command is a powerful text processing tool in Unix-like operating systems, including Linux. It is designed for pattern scanning and text processing. `awk` allows you to perform various operations on text files, including searching for patterns, extracting and transforming data, and generating reports. It is particularly well-suited for processing structured text data, such as columns and fields.

### Basic Syntax:

```bash

awk 'pattern { action }' filename

```

- \*\*pattern:\*\* Specifies a pattern to match in each line of the input.

- \*\*action:\*\* Specifies the action or set of actions to be performed when the pattern is matched.

- \*\*filename:\*\* The name of the file to be processed. If not provided, `awk` reads from the standard input (stdin).

### Key Features and Functions:

1. \*\*Pattern Matching:\*\*

- `awk` processes input line by line, and you can define patterns to match lines that meet specific criteria.

```bash

# Print lines where the second field is greater than 50

awk '$2 > 50 { print }' data.txt

```

2. \*\*Field and Column Processing:\*\*

- `awk` excels at working with fields and columns in structured data.

```bash

# Print the second column of each line

awk '{ print $2 }' data.txt

```

3. \*\*Built-in Variables:\*\*

- `awk` provides built-in variables like `NF` (number of fields), `NR` (current record number), and others, making it easy to perform computations.

```bash

# Print the number of fields in each line

awk '{ print NF }' data.txt

```

4. \*\*Mathematical Operations:\*\*

- `awk` supports mathematical operations, making it useful for calculations.

```bash

# Calculate the average of the third column

awk '{ sum += $3 } END { print sum / NR }' data.txt

```

5. \*\*String Concatenation:\*\*

- `awk` can concatenate strings and format output.

```bash

# Print a formatted message using string concatenation

awk '{ print "Name: " $1 ", Age: " $2 }' data.txt

```

6. \*\*Conditionals and Control Flow:\*\*

- `awk` supports if-else statements and other control flow structures.

```bash

# Print "Young" or "Old" based on the second column

awk '{ if ($2 < 30) print "Young"; else print "Old" }' data.txt

```

7. \*\*Regular Expressions:\*\*

- `awk` can use regular expressions to match patterns.

```bash

# Print lines containing the word "error"

awk '/error/ { print }' logfile.txt

```

8. \*\*User-Defined Functions:\*\*

- `awk` allows you to define your own functions for custom processing.

```bash

# Define a function to calculate the square of a number

awk 'function square(x) { return x \* x } { print square($1) }' data.txt

```

### Examples:

1. \*\*Print Lines Longer Than 80 Characters:\*\*

```bash

awk 'length > 80 { print }' data.txt

```

2. \*\*Calculate Total Sales:\*\*

```bash

awk '{ total += $3 } END { print "Total Sales: " total }' sales.txt

```

3. \*\*Print Unique Values in a Column:\*\*

```bash

awk '!seen[$1]++ { print $1 }' data.txt

```

4. \*\*Formatting Output:\*\*

```bash

awk '{ printf "Name: %-10s, Age: %2d\n", $1, $2 }' data.txt

```

### Important Notes:

- `awk` is often used in conjunction with other Unix commands through pipelines, allowing for complex text processing workflows.

- Understanding the structure of your input data is crucial for effective use of `awk`.

- `awk` is particularly useful for tasks involving text extraction, transformation, and reporting.

In summary, `awk` is a versatile text processing tool that allows you to manipulate and analyze structured text data efficiently. Its ability to work with fields, columns, and regular expressions makes it a powerful choice for various text processing tasks.

**How do you create and extract compressed archives in Linux?**

In Linux, the most common tools for creating and extracting compressed archives are `tar` and various compression utilities like `gzip`, `bzip2`, `xz`, and `zip`. Here are examples of how to create and extract compressed archives using these tools:

1. \*\*Create a `.tar.gz` Archive:\*\*

```bash

# Create a .tar.gz archive

tar -czvf archive.tar.gz /path/to/source

```

- `-c`: Create a new archive.

- `-z`: Compress the archive using gzip.

- `-v`: Verbose mode (optional, shows the progress).

- `-f`: Specify the archive file name.

#### 2. \*\*Create a `.tar.bz2` Archive:\*\*

```bash

# Create a .tar.bz2 archive

tar -cjvf archive.tar.bz2 /path/to/source

```

- `-j`: Compress the archive using bzip2.

#### 3. \*\*Create a `.tar.xz` Archive:\*\*

```bash

# Create a .tar.xz archive

tar -cJvf archive.tar.xz /path/to/source

```

- `-J`: Compress the archive using xz.

#### 4. \*\*Create a `.zip` Archive:\*\*

```bash

# Create a .zip archive

zip -r archive.zip /path/to/source

```

- `-r`: Recursively include directories.

### Extracting Compressed Archives:

#### 1. \*\*Extract a `.tar.gz` Archive:\*\*

```bash

# Extract a .tar.gz archive

tar -xzvf archive.tar.gz

```

#### 2. \*\*Extract a `.tar.bz2` Archive:\*\*

```bash

tar -xjvf archive.tar.bz2

```

#### 3. \*\*Extract a `.tar.xz` Archive:\*\*

```bash

tar -xJvf archive.tar.xz

```

#### 4. \*\*Extract a `.zip` Archive:\*\*

```bash

# Extract a .zip archive

unzip archive.zip

```

### Additional Notes:

- You can specify multiple files or directories to include in the archive after the archive file name.

```bash

# Create a .tar.gz archive with multiple sources

tar -czvf archive.tar.gz /path/to/source1 /path/to/source2

```

- To compress or decompress individual files, you can use the compression utilities directly:

```bash

# Compress a file with gzip

gzip filename

# Decompress a .gz file

gunzip filename.gz

```

**What is the purpose of the /proc directory in Linux?**

The `/proc` directory in Linux is a virtual filesystem that provides an interface to kernel data structures and kernel parameters. Unlike traditional filesystems that store data on disk, the `/proc` filesystem exists only in memory and is dynamically generated by the kernel. It serves as a window into the runtime state of the Linux kernel and allows users and processes to access information about the system, kernel, and running processes.

- Each running process on the system is represented by a directory with its process ID (PID) in the `/proc` directory. Inside each process directory, you can find information about the process, such as its command line, environment variables, file descriptors, and status.

# Example: Accessing process information for a specific process (PID 1234)

cat /proc/1234/status

- Various system-wide information and configuration settings are accessible through files in the `/proc` directory. This includes information about the kernel version, CPU information, memory usage, filesystems, interrupts, and more.

```bash

# Example: Displaying kernel version information

cat /proc/version

```

- Many kernel parameters and tunable settings can be viewed and modified through files in the `/proc/sys` directory. This provides a dynamic way to adjust kernel behavior at runtime without rebooting the system.

```bash

# Example: Viewing and modifying the kernel's maximum number of open file descriptors

cat /proc/sys/fs/file-max

echo "100000" > /proc/sys/fs/file-max

```

- The `/proc` directory contains information about the hardware configuration of the system, including details about CPU architecture, memory configuration, and PCI devices.

```bash

# Example: Displaying CPU information

cat /proc/cpuinfo

```

- Information about network-related settings and connections can be accessed through the `/proc/net` directory. This includes details about network interfaces, routing tables, and active connections.

```bash

# Example: Displaying network interfaces

cat /proc/net/dev

```

- The `/proc` directory provides files that can be used for performance monitoring and profiling. For example, information about the system's load average and uptime can be found in `/proc/loadavg`.

```bash

# Example: Displaying load average information

cat /proc/loadavg

```

- Information about loaded kernel modules can be found in the `/proc/modules` file. This includes details about the modules, their memory usage, and dependencies.

```bash

# Example: Displaying loaded kernel modules

cat /proc/modules

```

- The `/proc` directory contains information relevant to system security and debugging. For instance, information about process capabilities is available in `/proc/[PID]/status`.

```bash

# Example: Displaying process capabilities for a specific process (PID 1234)

cat /proc/1234/status | grep Cap

```

### Important Notes:

- While the `/proc` directory provides valuable information, it is important to note that the content is virtual and may change based on the kernel version and configuration.

- Writing to certain files in `/proc` can modify kernel parameters or trigger specific actions. However, not all files are writable, and modifying certain parameters may have system-wide consequences.

- Many utilities and commands in Linux, such as `ps`, `top`, and `sysctl`, gather information from the `/proc` filesystem to present it in a human-readable format.

In summary, the `/proc` directory in Linux serves as a dynamic interface to kernel information and configuration settings, allowing users, processes, and system utilities to access real-time information about the system's state and configuration.

**How do you set up a basic firewall using iptables?**

Setting up a basic firewall using `iptables` involves defining rules to allow or deny network traffic based on certain criteria. Below is a concise guide to creating a simple firewall configuration using `iptables`. This example assumes a default policy of allowing outgoing traffic and blocking incoming traffic unless explicitly allowed.

### Basic Firewall Setup:

1. \*\*View Current Rules:\*\*

```bash

sudo iptables -L

```

2. \*\*Flush Existing Rules (Optional):\*\*

```bash

sudo iptables -F

```

3. \*\*Set Default Policies:\*\*

- Allow outgoing traffic:

```bash

sudo iptables -P OUTPUT ACCEPT

```

- Drop incoming traffic:

```bash

sudo iptables -P INPUT DROP

```

4. \*\*Allow Established Connections:\*\*

```bash

sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

```

5. \*\*Allow Loopback Interface:\*\*

```bash

sudo iptables -A INPUT -i lo -j ACCEPT

```

6. \*\*Allow Specific Incoming Ports (e.g., SSH - Port 22):\*\*

```bash

sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT

```

7. \*\*Allow Specific Outgoing Ports (e.g., HTTP - Port 80):\*\*

```bash

sudo iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT

```

8. \*\*Save Rules (Persistence may vary based on Linux distribution):\*\*

- For CentOS/RHEL:

```bash

sudo service iptables save

sudo service iptables restart

```

- For Ubuntu/Debian:

```bash

sudo iptables-save > /etc/iptables/rules.v4

sudo systemctl restart iptables

```

### Additional Considerations:

- Adjust the rules based on your specific requirements. For instance, allow necessary ports for services like SSH, HTTP, or DNS.

- Ensure that you don't lock yourself out by allowing SSH or any necessary services before setting the default policy to DROP.

- To make the rules persistent across reboots, you may need to use distribution-specific tools or scripts. The examples above provide an immediate configuration.

- If you are using IPv6, consider configuring rules for the IPv6 version of `iptables` called `ip6tables`.

Remember that `iptables` is just one of many tools for configuring firewalls on Linux. Depending on your distribution, you might also encounter tools like `ufw` (Uncomplicated Firewall) or `firewalld`. Choose the tool that best fits your needs and system configuration.

**Explain the concept of inodes in Linux filesystems.**

In Linux, inodes are unique identifiers and metadata containers for files. To view the inode of a file, use `ls -i filename`; to check inode usage on a filesystem, use `df -i`.

**How can you find and kill a process using its port number?**

lsof -t -i:8080 | xargs kill

sudo fuser -k 8080/tcp

This command finds the process using the specified port and sends a kill signal to terminate it.

**what is .profile ?**

In Linux, the `.profile` file is a shell script that is executed when a user logs in. It is typically used to set environment variables, customize the user's shell environment, and define personal preferences. The `.profile` file is read by login shells such as Bash. Users can modify this file to customize their shell environment at login.

Example commands:

1. To edit the `.profile` file:

```bash

nano ~/.profile

```

2. Adding an environment variable (e.g., adding a directory to the PATH):

```bash

export PATH=$PATH:/usr/local/bin

```

3. Setting an alias (e.g., creating a shortcut for a command):

```bash

alias ll='ls -alF'

```

4. Configuring the default text editor for command-line use:

```bash

export EDITOR=nano

```

5. After modifying the `.profile` file, the changes take effect upon the next login or by sourcing the file:

```bash

source ~/.profile

```