What is a server?
high configuration computer

Mobile: RAM   Storage(HDD)   Processor(CPU) + android OS
Laptop: RAM   Storage(HDD)   Processor(CPU) + windows OS
Servers: RAM   Storage(HDD)   Processor(CPU) + Linux OS ==> linux server
Hardware components?

1.RAM   2.Storage(HDD) 3.Processor(CPU)
if we can connect these H/W parts, Operating system(OS)
What is OS?
----
it takes input from user, it will connect to H/W compoents & completes you are requests
windows servers ?   7-10%
Linux ? 90%
Linux:
------
was found in 1970's ==> unix

What is Opensource?
----
secret ==> Compositions & how do they make
windows ==> to develop windows OS there are lot of programms/code
will be written, which will be kept secretly
unix   ==> to develop linux OS there are lot of programms/code will be written,
 which is avaialble in internet for free
linus tolwards ==> he took unix source code, developed his own version ==> linux

Windows:	Linux:
Paid OS	Free OS
is not opensource	its opensource
Single user based OS	multi user based OS
Less secure	More secure
recommended for personal usage	buisness level usage , app hosting
GUI	CLI

Android is linux based OS
ios is linux based OS

To start with Linux?
----
1. Format my PC . install LINUX os
2. using extra softwares install LInux
3. We can get linux servers for free directly from Cloud service providers (AWS)
   Virtual servers ==> we can rent for free from AWS
   RAM:1GB HDD:8GB CPU: 1core

Create an acoount in AWS ?
----
signup https://aws.amazon.com/

```
email id , phone number , Debit card / credit card ( dont use rupay cards),
card is enabled international transactions

we will launch an server in AWS & we will connect to that?
----------------------------------------------------------
Servers will be available in aws by name ec2

in order to acccess servers we have use an s/w ==>
 gitbash ( https://git-scm.com/download/win )
--------------------------------------------------------------------------------
------------
Note:
-----
1. Different names of servers are , SERVER == VIRTUAL MACHINES (VMs) ==
EC2INSTANCE ==
 COMPUTERS == MACHINES == all are same
2. Screen which we see after logging in is called as Shell  / Terminal
3. EC2 instance  ==> free for 750 hours / months
4. Observations in screen:

   [ec2-user@ip-172-31-21-241 ~]$
   ec2-user          ==> username using which we have logged in to
server/ec2-instance
   ip-172-31-21-241  ==> ip address ( unique number )

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++
Linux:
------
Linux doesent have a GUI
in linux we have to use commands only

linux is case sensitive
that is , it will accept commands only lower case letters.
Linux OS:
---------
what are we going to learn in linux

as a technical person ==> create files, modify files / folders, edit files, copy
files
transfer files from 1 server to another server,

File based commands.

COMMANDS:
---------
File operations commands in linux:
----------------------------------
1. Touch command:
   -------------
   Touch command is used to create a blank file

   syntax: touch <file_name>
```

```
2. ls (list) command:
   -----------------
    ls command is used to list all files

folders are called as directories in linux
mkdir <directory/folder name>

3. mkdir command:
   ---------------
   mkdir command is used to create directories / folders
      syntax: mkdir <directory_name>

   Note: in linux we call folders as directory
3. pwd (present working directory) command:
   ----------------------------------------
   pwd shows in which directory we are in currently

   whenever we loggin to ec2instance, default directory (home directory)
 we will be at is /home/ec2-user/
4. cd (change directory) command:
   -----------------------------
   cd command is used to change directory (or folder)
   syntax: cd <directory_name>

   Note on navigation inside directories:
   --------------------------------------
   to come one directory backwards ==> cd ..
   to come two directory backwards ==> cd ../..
   to come three directory backwards ==> cd ../../..
   to come to users default directory (i.e /home/ec2-user) from any location ==>
cd
5. How to remove / delete a file & directory?
   ------------------------------------------
   rm (remove)
   rm -f <file_name> ==> it will delete file
   rm -rf <directory_name> ==> it will delete directory
6. Clear:
   ------
   clear command will clear your screen / terminal / shell.

vi (vi editor is used to edit the file)
----------------------------------------
syntax: vi <your_file_name>

note: make sure you can see insert using i button,
Esc(keyboard) is used to come out of insert mode

  comeout of file with saving    ==> Esc(keyboard) :wq! ==> save(w)  quit(q) !
forcefully
  comeout of file without saving ==> Esc(keyboard) :q!
cat (concatanate)
--------------
Display the content of a file
syntax: cat <your_file_name>
```

```
cp (copy)
----------------
used to copy content of a file to another file
syntax: cp <file_to_be_copied> <destinationFilename>
if <destinationFilename> dosent exist , copy will create that
mv (rename)
----------------
used to rename file
syntax: mv <your_old_fileName> <newName>
vi (vi editor is used to edit the file)
----------------------------------------
syntax: vi <your_file_name>

note:  make sure you can see insert option( using i button (keyboard) while
adding content
       Esc(keyboard) is used to come out of insert mode

  comeout of file with saving    ==> Esc(keyboard) :wq! ==> save(w)  quit(q) !
forcefully
  comeout of file without saving ==> Esc(keyboard) :q!

cat (concatanate)
--------------
cat command is used to display the content of a file
syntax: cat <file_name>
cat -n <file_name> ==> disaplays the content of file along with line numbers

cp (copy)
----------------
used to copy content of a file to another file
syntax: cp <file_to_be_copied> <destinationFilename>

if <destinationFilename> dosent exist , copy will creates that file & copies
content

mv (rename)
----------------
used to rename file
syntax: mv <your_current_fileName> <newDesired_FileName>

create multiple blank(empty) files:
-----------------------------------
touch file1 file2 file3

create multiple directories:
----------------------------
mkdir dir1 dir2 dir3
ls -lrt
--------
list the files & also list recently created or recently modified file will
be dispalyed in bottom of terminal (or screen or shell)
date:
-----
date command is used to display current time
```

```
How to craete a directory inside another directory (nested directory)
--------------------------------------------------------------------
mkdir -p dir1/dir_tobe_created
  p: path
mkdir -p dir1/dir2/dir3
  -p means path of directories.
   All directories in the specified path will be created.
   First dir1 will be created and in that dir2 will be created and
within dir2 directory dir3 will gets created.
to delete multiple files:
--------------------------------------------------------------------------
files ==> rm -f <filename1> <filename2>
directories/folders  ==> rm -rf <directoryname1> <directoryname2>
Create hidden files
---------------------------------------------------------------------------
to create hidden file / directory we need to use
.(dot) at the front of your file / directory name
ls -a (or) ls -lrta ===>to list hidden files

more:
------
more is used to display content of a file pagewise

syntax: more <fileName>

  in more we can only scroll down & we cant scroll up
  use spaceBar button to move down
  press q to come out of the mode
less:
-----
used to display content of a file pagewise
  syntax: less <fileName>

  using less we can scroll up & scroll down.
  use arrow key buttons to move up & down
  press q to come out of the mode

++++++++++++++++++++++++++++++++++++++++++++++++
1.Vi mode
  - to set numbers in file: Esc(keyboard) & type set nu
  - to goto any particular line in vi mode ==>
Esc(keyboard) : <lineNumber> & Enter(keyboard)
  - to delete a line ==> Esc(keyboard) & type dd (no colons)
2. CD
  - to move inside directorys using path cd
<dirname>/<dirname>/<dirname>/<dirname>

Assignment:
-----------
copy all files present in /home/ec2-user/ directory into directory called dirX
--------------------------------------------------------------------
Redirect >
---------------------------------------------------------------------------------
---------
```

```
Redirect is used to write output of a command to file
  eg: ls -lrt > myFileName
      ls > myFileName

  it will overwrite / override the content of file each time if file already
exists

Append >>
--------------------------------------------------------------------------------
------
append is used to write output of a command to file, but append will not
overwrite/
override content of  file, it will attach to the end of a file.
   eg: ls >> myFileName
       echo "hello all, welcome to devops classes" >> myFileName
 cat , more , less
Head
--------------------------------------------------------------------------------
-------
head command is used to display top most part of a file
  syntax: head -5 <fileName> ==> it will show first 5 lines of a file

  i.print first 15 lines of my file
    head -15 <fileName>

  ii. print first 1st line of my file
     head -1 <fileName>
 Note:
  -----
  if you dont specify any number, head by default will print first 10 lines
  head <fileName>
Tail:
--------------------------------------------------------------------------------
---------
tail command is used to display the bottom part of file.

  i. to display last 5 lines of a file
  syntax: tail -5 <fileName>
  ii. to display very last line of a file
  syntax: tail -1 <fileName>
echo:
-----
echo is used to print a statement in terminal.

syntax: echo "<your_content>"

eg: echo "hello all, welcome to devops classes"
What is Piping?
================================================================================
======
    |
   A way of connecting commands together.
   A way of passing data from the output of one command as the input to another
command.
```

usecase(i): to print only line number 99 from myFile

```
         [ec2-user@ip-172-31-16-223 ~]$ tail -2 numbers_file
          Line 99
          Line 100
          [ec2-user@ip-172-31-16-223 ~]$
```

observation: tail -2 numbers_file will print last 2 lines of the file.
but we need only first line (i.e line 99) from the printed ouput, here i can join 2 commands
using piping | which will take input from previous command & generate new output.

```
              [ec2-user@ip-172-31-16-223 ~]$ tail -2 numbers_file | head -1
          Line 99
          [ec2-user@ip-172-31-16-223 ~]$
```

(ii) to see / display 97th line only

```
                    tail -3 numbers_file | head -1
```

(iii) to display 9th line of a file

```
                     head -10 numbers_file | tail -1
```

Grep
----
grep is used to search the pattern (keyword) in a file.
if pattern is found, it will print all matched lines /entire line

syntax: grep <pattern_to_search> fileName

Note: Pattern = name =string all are same

SI	City_Name	Population	State
1	bengaluru	1.4 crore	karnataka
2	mumbai	2.5 crore	maharashtra
3	newdelhi	1 crore	delhi
4	chennai	90 lakhs	tn
5	kolkata	1.2 crore	westbengal
6	ahmedabad	60 lakhs	gujarat
7	hyderabad	70 lakhs	andhrapradesh
8	pune	45 lakhs	maharashtra
9	Hubli_dharwad	10 lakhs	karnataka
10	gurgaon	10 lakhs	haryana

----

to search pattern ==> "karnataka"

```
syntax: grep name/pattern fileName
        grep karnataka census_data

        grep mumbai census_data

        grep chennai census_data
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Next class Topics:
------------------
- word count (wc)
- egrep , vgrep & rgrep
- links(Hard link and soft link)
- File permissions
Note:
------
content for head & tail with pipeing options
This is Line number 1
This is Line number 2
This is Line number 3
This is Line number 4
This is Line number 5
This is Line number 6
This is Line number 7
This is Line number 8
This is Line number 9
This is Line number 10

This is Line number 38
This is Line number 39
This is Line number 40
This is Line number 41
This is Line number 42

This is Line number 61
This is Line number 62
This is Line number 63
This is Line number 64

ThiThis is Line number 100
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++
Agenda:
=======
- word count (wc)
- egrep , rgrep
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++
Note:
-----
Directories
1. to copy 1 directory to another directory
   syntax: cp -r <source_directory> <target_directory>

   Here: -r ==> recursiveness (consider all files,subdirectories inside that
dierctory)

2. MV (move) command can be used in two ways
   - for Files ==> we can use to rename a file
   - for Directories ==> apart from renaming, mv(move) is also used to move
 a directory to another directory (i.e cut & paste directories)
```

```
3. in linux * this indicates all
word count:
--------------------------------------------------------------------------------
------------
word count is used to count number of lines, number of words & number of
charecters in a file
  syntax: wc <fileName>

other options with word counts
   i.  to count only number of lines
       syntax: wc -l <fileName>

   ii. to count only number of words
       syntax: wc -w <fileName>

   iii. to count only number of charecters
        syntax: wc -c <fileName>
Grep command:
-----------------------------------------------------------------------------
to search a pattern (keyword) in a file
if a pattern(keyword) is found it will print all the matched line

syntax: grep <pattern_tobe_searched> <filename>

        grep hyderbad census_data

        to search for a pattren without case sensisnsivity

Other use cases of grep command:
-----------------------------------------------------------------------------
- To search ignoring case sensitivities:
  grep -i <pattern_to_search> fileName
  i ==> search by ignoring case sensitivity
- egrep:
  -----
  its used to search multiple patterns in same file

  grep -e "<pattern_1>" -e "<pattern_2>" fileName
- rgrep (RECURSIVE grep):
  ---------------------------------------------------------------------------
  Search pattern in present working directory & also in all sub-directories.

  grep -r <pattern_to_search> <directory_name>

   - rgrep will list filename containing pattern & also prints the
 entire line which contains charecter.
   - To search patterns in all files of present working directory:
      syntax: grep -r <pattern_to_search> *
*******IMPORTANT***********

What is a link file and how many types?
-------------------------------------------------------------------------------
Link file is a short cut file to the original file.
```

```
There are two types of links files available in Linux.
(i) Soft link
(ii) Hard link

What is soft link and how to create it?
---------------------------------------------------------------------------------
-
- Soft link is nothing but a short cut file.

Syntax: ln -s <original_file_or_directoryname>
<linkfilename_or_directorywithpath>

- If original file is deleted, no use of softlink  (ie.,
we cannot access the original data by selecting the link file)
- Soft link can be applied on both directories and files.
- inode number of original and soft link files are different.
- If we edit original file, the softlink are also updated automatically.
What is hard link and how to create it?
---------------------------------------------------------------------------------
-
- Hard link in nothing but a backup of a file.

Syntax: ln <original_file_name> <linkfilename>

- If the original file is deleted, there is no effect on hard link file.
(i.e., we can access the original file data even though the link file is
deleted.)
- Hard links can only be applied on files only not on directories.
- inode number of original and hard link files are same.

Note on hardlinks:
- If the original is edited, the updations are applied on both original and hard
link files.
- The size of the hard link file is same as the size of the original file.
Note:
----
  what is inode number?
  Linux creates a unique number for all files & directories that gets created,
  to check i node number, ls -li <file/directory name>, first column shows inode
number
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++
File system in linux:
-------------------------------
In Windows: all OS files will be stored in C drive

In Linux: all OS files will be stored in /

[ / is called as ROOT directory ]

/ ==> ROOT directory , is like c drive in windows & it will
contain all linux os related files & folders.
to go to ROOT directory ==> cd /
```

Folders in root directory:

```
[ec2-user@ip-172-31-16-223 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-16-223 ~]$ cd /
[ec2-user@ip-172-31-16-223 /]$ pwd
/
[ec2-user@ip-172-31-16-223 /]$ ls
bin   dev  home  lib64  media  opt   root  sbin  sys  usr
boot  etc  lib   local  mnt    proc  run   srv   tmp  var
[ec2-user@ip-172-31-16-223 /]$
```

```
Important Folders in root directory:
1) bin:
   ---
    It contains the commands and binary files.
       User can access the commands.
       Ex:  mkdir, ls, cd, ps….
2) etc:
   ---
    It contains the configuration files.
3) home:
   ----
   It contains home directories for all users to store their personal files.
   Ex: /home/ec2-user or /home/bharath
4) var:
   ----
    It contains variable files.
    This includes system log files(/var/log), emails (/var/mail)
5) tmp :
   -----
    It contains the temporary files.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+
Note:
-----
1. Autocompletion:
   ----------------
   use tab key from your keyboard to use autocompletion
   - type first charecter then type tab
     ex: to cat census data file
         cat c<press_tab_key>
             cat census_data
2. Compute resources / Hardware resources means RAM, HDD , CPU
   in EC2 instance (free tier) ==> RAM 1GB , 8GB HDD , 1core cpu

3. whoami : whoami command is used to check which user you have logged in to
server as.

Hardware compoent's / COMPUTE resources check commands:
-------------------------------------------------------

-  free (To check RAM size):
  --------------------------
```

```
    free command is used to check system memory(RAM)
    syntax: free -m

    Note: in linux RAM is generally called as memory

- To check disk size / HDD / Storage:
   -----------------------------------
   df means disk fragmentation.
   It displays file system disk space usage

   syntax: df -kh .

   Filesystem      Size  Used Avail Use% Mounted on
   /dev/xvda1      8.0G  2.2G  5.8G  28% /
- To check number of cpu's :
  -------------------------
  nproc ==> it stands for number of cpu's in the server
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
File description:
-------------------------------------------------------------------
Whenever a new file is created, when we list the file using. ls -lrt
 commands we can seee description of those files like below

   • File type: whether its a file or directory ?
   • Date and time of creation
   • File size
   • Permissions on the file.
   • Owner and group owner of the file.

[ec2-user@ip-172-31-16-223 ~]$ ls -lrth
total 8.0K
-rw-rw-r-- 1 ec2-user ec2-user 792 Nov  4 01:39 samplefile
-rw-rw-r-- 1 ec2-user ec2-user 600 Nov  4 01:54 census_data
drwxrwxr-x 2 ec2-user ec2-user   6 Nov  7 05:09 d1
Observations:
First column is Permissions block:
permissions block is used to type of file & its permissions:
 - in first charecter of permissions block we can identify type of a file
whether its directory or file
   if it starts with - ==> file
   if it starts with d ==> directory
    Note:
   linux uses code for file permissions
    r ==> read    ==> code ==> 4
    w ==> write   ==> code ==> 2
    x ==> execute ==> code ==> 1
    - ==> no permissions ==> code ==> zero

 read ==> we can cat / more / less file
 write ==> we can edit a file
 example: permissions of below file in coded format
-rw-rw-r-- 1 ec2-user ec2-user 792 Nov  4 01:39 samplefile
 420420400
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
User adminstration:
--------------------------------------------------------------------------------
In linux we have 2 types of users
1. Root user:
   - root user ==> owner (adminstrator) present in all linux servers
   - root user has highest priviliges, he can do anything in this server
   - root user can install/uninstall any softwares
   - root user can add/delete user
   - root user can create/edit/delete any files in the server.
   2. normal users(non root users):
 - normal users can only create/edit/delete any files only in thier respective
home directories.
   that is in /home/ec2-user, /home/nithin
   by default they cannot do any install/uninstall softwares & other
administration activities

How to switch from ec2-user to root user?
-----------------------------------------
sudo su -
How to add a new user to linux machines?
-----------------------------------------
  switch as root user first
  syntax: useradd <username>
  eg:
  [root@ip-172-31-93-35 ~]# useradd viratkohli

How to verify new user created / not ?
----------------------------------------
- Go to /home directory & verify directory is created in newly created users
name
                           (or)
  check for entry in file --> cat /etc/passwd | tail -1
    [ec2-user@ip-172-31-93-35 ~]$ cd /home
    [ec2-user@ip-172-31-93-35 home]$ ls -lrt
    total 0
    drwx------ 4 ec2-user   ec2-user   142 Jul  6 03:23 ec2-user
    drwx------ 2 viratkohli viratkohli  83 Jul  6 05:53 viratkohli
    [ec2-user@ip-172-31-93-35 home]$
Note:
----
- whenever we create user, a new group will also gets
created automatically with the created users name .
- whenever we create user, Inside /home automatically
new folder will gets created with the created users name

command to switch from one user to another user?
-----------------------------------------------
su - <username>
```

```
 How to set password for newly created user?
  syntax: passwd <username_created>
  Note: password you are typing will not be visible in your screen
  eg:
```

```
  [root@ip-172-31-93-35 ~]# passwd viratkohli
  Changing password for user viratkohli.
  New password:
  Retype new password:
  passwd: all authentication tokens updated successfully.
  [root@ip-172-31-93-35 ~]#
```

SUDO users:
----
we should not permitt all users to access ROOT user due to security purpose.

Sudo stands for “super user do”

if a user has sudo permissions, then he can run commands like root user ,
if we add any user to SUDOERS list, we can run commands like root user,
without logging as root user.

How to add user to sudo group?

   Step1: login as root user (because only root user can give sudo permissions
to any user)
```
          sudo su -
```
   Step2: edit vi /etc/sudoers file will gets opened now go to last line of file
add block
```
               <Your_userName> ALL=(ALL)       NOPASSWD: ALL
          example: devops_user ALL=(ALL)       NOPASSWD: ALL
                 save & quit the file ( :wq! )
                 after saving the file run below command
```

   step3: Run below command to apply changes done in previous steps
```
          service sshd restart
```

   now user got added to sudoers list, switch to created user using
su - <user_name>, check by running commands with sudo prefix.
Note:

- to delete user command   ==> $userdel <user-name>
- to create group manually ==> $groupadd <group-name>

Assignment:

- command to check size of afile

- create multiple user in your names, add the created user
to sudo group & verify sudo priviliges.
++
Note:
-----

1. Absolute path & relative path:

   -absolute path means complete path to your file
    it will start from / ==> root directory
   - Every single file and directory starts from the root(/) directory

```
[ec2-user@ip-172-31-82-40 ~]$ ls d1/d2/d3/census_data_linkfile
```

```
d1/d2/d3/census_data_linkfile
[ec2-user@ip-172-31-82-40 ~]$
[ec2-user@ip-172-31-82-40 ~]$
[ec2-user@ip-172-31-82-40 ~]$
[ec2-user@ip-172-31-82-40 ~]$ ls /home/ec2-user/d1/d2/d3/census_data_linkfile
/home/ec2-user/d1/d2/d3/census_data_linkfile
[ec2-user@ip-172-31-82-40 ~]$
```

What is the difference between absolute path and relative path?

```
    An absolute path begins with the root (/) directory ==>
/home/ec2-user/dir1/myfile1
    A relative path starts from your present working directory(pwd),
if i am in /home/ec2-user
directory then relative path is dir1/myfile1
```

2. man (manual pages): If you don't know the command you
can use man and find the details of the command

```
   syntax: man <any_command>
   Eg: man ls
```

3. --help: to see how you can use command effectively with other options

```
   synatx: <any_command> --help
```

++

File Permissions

Permissions to files / directories are applied in 3 level
1. user level
2. group level
3. other users level

Total 9 permissions. First 3 are user permissions, next 3 are group permissions and
next 3 are others permissions

```
d rwx   rw-   r-x 2 ec2-user ec2-user 6 Nov  7 03:06 dir1
d rwx   rw-   r-x
  ---   ---   ---
   |     |     |
  user   |     |
       group   |
             others
  rwx   rw-   r-x
 ---   ---   ---
r=read     ==> use commands such as cat, more, less, head / tail for a file
w=write    ==> use commands such vi(to modifiy), rm (to delete) for a file
x=execute  ==> you can run any command, scripts / run a file as a programm
```

r=read=4
w=write=2
x=execute=1

```
current        desired
- rw- rw- r-- ==> rwx     rwx      rw-
  420 420 400     421     421      421
                 4+2+1   4+2+1    4+2+1
   6   6   4       7       7        7

 to give only read permissions to all

 - r-- r-- r--
   400 400 400
    4   4   4

 to give write permisions to user
 - rwx r-x r--
   421 401 400
    7   5   4
---------------------------------------------------------
chmod (change mode)
--------------------------------------------------------------------------------
-----
chmod command is used to change the permissions of file / directory
syntax:
  for file--->    chmod <Permission_in_Number> <FileName>
                     eg: 1. chmod 777 <FileName> ==> it will give all
permissions(rwx) to everyone

  for directory:  chmod -R <Permission_in_Number> <DirectoryName>
                     -R ==> will change/applies mentioned permissions of all
directories & sub directories
examples:
    list of some common settings, numerical values and their meanings:

   -rw------- (600) -- Only the user has read and write permissions.

   -rw-r--r-- (644) -- Only user has read and write permissions; the group and
others can read only.

   -rwx------ (700) -- Only the user has read, write and execute permissions.

   -rwxr-xr-x (755) -- The user has read, write and execute permissions; the
group and others
can only read and execute.

   -rwx--x--x (711) -- The user has read, write and execute permissions; the
group and others
can only execute.

   -rw-rw-rw- (666) -- Everyone can read and write to the file. Bad idea.

   -rwxrwxrwx (777) -- Everyone can read, write and execute. Another bad idea.
```

---

chown(change ownership)

```
-----------------------
chown command is used to change ownership of file / directory

syntax: chown <usersName>:<groupName> fileName
        chown -R <usersName>:<groupName> directoryName

Note:
-R is mandatory for directories
for running chown / chmod may require sudo permissions
```

```
Software management in Linux:
-----------------------------
- we do need to install/uninstall third party software into linux OS
- we do update our softwares & os when required
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Note:
-----
1. package == software
   android ==> playstore ==> search ==> install
   windows ==> microsoft/internet ==> search ==> install
   Linux ==> yum (or) apt ==> search ==> install

   - in redhat based systems software is managed by using yum tool
   - in debian based systems(ubuntu) software is managed by using apt tool
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
yum :
---------
- Yum is package manager using which we can install any packages (softwares)

  we can search, install, remove a software from linux os

syntax: yum search <software/package name>
        yum search tree
        yum search git

For Adding / Installing a software:
--------------------------------
- using install keyword along with apt/yum
  syntax: yum install <software/package name>

Note: while installing / unintslaling linux will ask for confirmation using y/n
==> select y

For uninstalling a software:
--------------------------------
- using remove keyword along with apt/yum
   syntax: yum remove <software/package name>

updating Linux system:
---------------------
- Updating software / app / os is very important as updates brings added
```

security
and add new features to our software
- in linux, updates won't reach us automatically as in window
we have to manually update our system:

- in linux os we do have two kinds of updates
   update & upgrade
i) update:
- it updates the list of software, applications that are ready to get update.
  Eg: sudo yum update
ii)upgrade
  - it actually updates every software that has updates
  - it takes may take hours/days for upgrade
++

Sleep command:
----
it will pause your terminal/shell for certain time

```
syntax: sleep <time_in_seconds>
        sleep 30
```

ps

It is used to display information about running processes.
which stands for "process status."

ps -aux:
----

is used to list all process running in linux.
ps -ux ==> displays only processes started by current user
ps -aux ==> displays all processes started by all user

When you run `ps aux`, the command will display a list of processes in
 a tabular format. Here's a breakdown of the columns you might see:

- USER: The username of the owner of the process.
- PID: The unique Process ID that identifies each running process.
- %CPU: The percentage of CPU (processor) usage by the process.
- %MEM: The percentage of RAM (memory) usage by the process.
- START: The start time of the process.
- TIME: The total accumulated CPU time used by the process.
- COMMAND: which commans created the process.

Here's an example output of `ps aux`:

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
john     12345  5.0  2.0 200000 100000 ?       R    Sep10  10:30
/usr/bin/example
mary     23456  0.0  0.5  50000  25000 pts/1   S    Sep11   0:45
/usr/bin/another
root     34567  1.0  1.5 150000  75000 tty1     Ss   Sep09   2:00 /sbin/init
```

```
--------------------------------------------------------------------------------
---
kill
--------
kill command is used to stop/ terminate the running process (ps -ef)
syntax: kill -9 <ProcessID>
        9==> signal
SED:
--------------------------------------------------------------------------------
-----
 sed command is used to replace a string/pattern without opening file.

 syntax:

 sed -i 's/<old_name>/<new_name>/g' filename


   [ec2-user@ip-172-31-16-223 ~]$ cat samplefile2
   Hello all,
   we are learning unix
   [ec2-user@ip-172-31-16-223 ~]$

   to replace name unix with Linux in above example ==>
   sed -i 's/unix/Linux/g' samplefile2
--------------------------------------------------------------------------------
--------
Find:
-----
find command is used to search for a file/folder.

Find provides multiple options for searching a file eg: name, size, group,
 user, date, permissions, etc..,
Note:- Some times you are trying to find some files you got permission denied
errors, On that case use sudo before the
command

syntax: find <path> -name <filename_tobe_searched>

find /home/ec2-user -iname  <filename_tobe_searched>

i ==> ignore case-insensitive

To search only a directory
find <path> -type d  -name <directoryname_tobe_searched>

To search only a file
find <path> -type f  -name <filename_tobe_searched>

Assignment:
-----------
find command to list files size more than 1GB ==> du sh-
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SSH (Secure Shell):
```

```
-------------------
using ssh we can login to linux server
SSH is the most commonly used way to login secuerly to any linux servers

In SSH we have two type of logging in:
--------------------------------------------------------------------------------
-
    1. using private key file (.pem file ) ( more secure)

           so far we have logged in to ec2 by this type
(using .pem file or private key file)

                syntax: to login using pem file
                ssh -i <path_to_private_key_file>  <username>@<ipaddress>
                ssh -i "Sept_Key_pair.pem"
ec2-user@ec2-54-224-83-60.compute-1.amazonaws.com

                Note: .pem file is called as private key file....

    2. using password  ( less secure)

                syntax: to login using password
                ssh <username>@<ipaddress>

                ssh yash@54.224.83.60

Note: by default aws / linux blocks password based authentication, so to change
that

How to login server from users which we have created using passwords?
-------------------------------------------------------------------
1. In linux servers by default, password authentication is set no(disabled),
(login as root) enable in the /etc/ssh/sshd_config  file and restart sshd
service
     (PasswordAuthentication no ) change to (PasswordAuthentication yes)

     service sshd restart

2.In password based authentication you need to provide password mandatorily
while logging in

Note:
syntax to set password for user
passwd <usersName>

uptime
--------------------------------------------------------------------------------
is used to check from how long the system is up & running
  (or)
how long since server is started

who
--------------------------------------------------------------------------------
---
```

```
command is used to check how many user logged in to system

Hostname
--------------------------------------------------------------------------------
---
command is used to print your server name
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
Ports:
------
Port numbers are used as Point of communication

linux system has 65,000 port numbers available

0-1000 ==> reserved by linux
ssh - port 22
http - port 80
https - port 443
telnet - port 23

netstat (network statsics)
-------------------------
command is used to list all listnening/active ports

syntax: netstat -lntp
--------------------------------
Linux OS:
---------
linux operating system has 2 major faimilies
1. RHEL-Family:
   -----------
   linux falvors in Redhat family are Redhat, Centos and Amazon Linux...

2. Debian family:
   --------------
   linux falvors in Debian family is ubuntu
Note:
in Redhat linux family---> yum install <software_name>
in debian/ubuntu family -> apt-get install <software_name>
--------------------------------------------------------------------------------
------
how to check OS related info
cat /etc/os-release
--------------------------------------------------------------------------------
top[top]
-------
Command is used to check which process utilizing what % of CPU / % of
Memory(RAM)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++
Note:
-----
SERVER == COMPUTERS == MACHINES === VIRTUAL MACHINE === EC2 INSTANCES ====> All
are same
```

```
what all compoenents does  a server have?
--> HARDware resources ( RAM + HDD + CPU ) + OS

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++

Networking :
=============
Two or more systems connected each other ==>> networking

systems ==>>>nothing but servers.

Two servers are in same network ==>>> minimum requirements..

Two servers must be cabled with other.
================================================================================
=====

Networking advantages..

1. files transfer ==>>> from one server to another server.

2. Remoteuserly login ==>> from one server to another server.
--------------------------------------------------------------------------------
scp (secure copy)
------------------------------------------------------------------------------
used to transfer files/directory from one server to another(remote) server

syntax:
--------
For File:
scp <file_tobe_copied>
<username_of_target>@<ipaddress_of_target>:<path_in_targetserver>

scp census_data_tobecopied  hari3@13.49.240.55:/home/hari3

scp sample1 soni@54.162.242.161:/home/soni/
54.162.242.161

For Directory-->
scp -r <file_tobe_copied>
<username_of_targetserver>@<ip_of_targetseer>:<path_in_targetsver>

-r ==> mandatory while copying directories

scp -r test soni@54.162.242.161:/home/soni/

Note:
-----
- create user with password
- ensure password authentication is enabled
--------------------------------------------------------------------------------
-------
```

```
ping
--------------------------------------------------------------------------------
----
is used to check any server is up & running

syntax:
ping <ip_of_remote_server>
--------------------------------------------------------------------------------
-------
how to check size of a file or directory?
-----------------------------------------
du -sh <filename/directoryname>
--------------------------
cut command
-------------------
is used to cut the content of a file column wise
syntax: cut -d " " -f<columnNumber> <filename>

d=delimeter=separator
f= field

vi cut_example
----------------------------
AndhraPradesh AP hyderabad
ArunachaPradesh AC Itanagar
Assam AS guwahati
Bihar BH patna
Chhattisgarh CG Raipur
karanataka KA benagluru
uttarparadesh UP lucknow
jammuKashmir JK srinagar
-------------------------

vi cut_example2
----------------------------
AndhraPradesh|AP|hyderabad
ArunachalPradesh|AC|Itanagar
Assam|AS|guwahati
Bihar|BH|patna
Chhattisgarh|CG|Raipur
karanataka|KA|benagluru
uttarparadesh|UP|lucknow
jammuKashmir|JK|srinagar
-------------------------
logs:

[ec2-user@ip-172-31-93-35 ~]$ cut -d " " -f1 cut_example
AndhraPradesh
ArunachalPradesh
Assam
Bihar
Chhattisgarh
karanataka
uttarparadesh
```

```
jammuKashmir
[ec2-user@ip-172-31-93-35 ~]$ cut -d " " -f2 cut_example
AP
AC
AS
BH
CG
KA
UP
JK
[ec2-user@ip-172-31-93-35 ~]$ cut -d " " -f3 cut_example
hyderabad
Itanagar
guwahati
patna
Raipur
benagluru
lucknow
srinagar
[ec2-user@ip-172-31-93-35 ~]$
[ec2-user@ip-172-31-93-35 ~]$
[ec2-user@ip-172-31-93-35 ~]$ cut -d " " -f1,3 cut_example
AndhraPradesh hyderabad
ArunachalPradesh Itanagar
Assam guwahati
Bihar patna
Chhattisgarh Raipur
karanataka benagluru
uttarparadesh lucknow
jammuKashmir srinagar
```

```
awk_example_file
------------------
AndhraPradesh    AP  hyderabad
ArunachaPradesh  AC  Itanagar
Assam            AS  guwahati
Bihar            BH  patna
Chhattisgarh     CG  Raipur
karanataka       KA  benagluru
uttarparadesh    UP  lucknow
jammuKashmir     JK  srinaga
```

----
awk

----
awk command is used to cut file column wise & row wise

syntax:

```
awk -F " " '{print$1}' <filename>
```

-F ==field separator

(i)to print/cut first column of a file

```
awk -F " " '{print$1}' <filename>

(ii)to print/cut third column of a file

awk -F " " '{print$3}' <filename>

(iii) to print/cut last column of a file
   awk -F " " '{print$NF}' disk_util_file

(iv) to print/cut second last column of a file
   awk -F " " '{print$(NF-1)}' disk_util_file
```

```
[root@ip-172-31-10-87 dirA]# cat awk_example_file
AndhraPradesh    AP  hyderabad
ArunachaPradesh  AC  Itanagar
Assam            AS  guwahati
Bihar            BH  patna
Chhattisgarh     CG  Raipur
karanataka       KA  benagluru
uttarparadesh    UP  lucknow
jammuKashmir     JK  srinagar
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]# awk -F " " '{print$1}' awk_example_file
AndhraPradesh
ArunachaPradesh
Assam
Bihar
Chhattisgarh
karanataka
uttarparadesh
jammuKashmir
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]# awk -F " " '{print$2}' awk_example_file
AP
AC
AS
BH
CG
KA
UP
JK
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]# awk -F " " '{print$3}' awk_example_file
hyderabad
Itanagar
guwahati
patna
Raipur
benagluru
```

```
lucknow
srinagar
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]# df -kh
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        4.0M     0  4.0M   0% /dev
tmpfs           475M     0  475M   0% /dev/shm
tmpfs           190M  2.8M  188M   2% /run
/dev/xvda1      8.0G  1.6G  6.5G  20% /
tmpfs           475M     0  475M   0% /tmp
tmpfs            95M     0   95M   0% /run/user/1000
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]# df -kh | head -5
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        4.0M     0  4.0M   0% /dev
tmpfs           475M     0  475M   0% /dev/shm
tmpfs           190M  2.8M  188M   2% /run
/dev/xvda1      8.0G  1.6G  6.5G  20% /
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]# df -kh | head -5 | tail -1
/dev/xvda1      8.0G  1.6G  6.5G  20% /
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]#
[root@ip-172-31-10-87 dirA]# df -kh | head -5 | tail -1 | awk -F " " '{print$5}'
20%
[root@ip-172-31-10-87 dirA]#
```

++

```
x=10
y=20
```

variables:

Variables are memory location which can store some values.

```
syntax: <variable_name>=<variable_value>
        eg: MYNAME=devops_user
```

+++

startup file (.bashrc):
----
startup file will get executed automatically, eachtime you login to your server.

.bashrc file location ==> every user's home directory ==> /home/<user>

each user will have his own .bashrc / startup file

Observation on startup file:
vi .bashrc ===> go to last line & add some echo statements & save comeout

close your session (ctrl+D) & login again , you will see the message putup in
.bashrc file printed as soon as you logsin.
++
+++++++
telnet :

Telnet is used to connect to remote systems.
it is oldest network protocol.
it is not secure.
It connects to servers and network equipment over port 23.

syntax:
```
telnet <ip_of_remote_server> <port_number_tobe_checked>
```

note:
```
yum install telnet -y
```
how do you check a port number in another server is listening or not, from your
server?


```
telnet <ip_address_remote_server> <port_number_tobe_checked>
```

if the port in remote server is opened(listening) ==> we get connected to that
server
if the port in remote server is not opend(non listening) ==> we get connection
failed error

ssh-keys:
----
ssh keys help secure your communication with other servers.
ssh-keygen creates keypair(two files).
   1. public key (like a lock) ==> can be share with any servers / anyone
   2. private key (like a key) ==> keep it secret (dont share with anyone).

syntax: ssh-keygen
observation:
 - after running above command, use enter (keyboard) 3 times
 - all ssh files get stored in the user's home directory ==>
/home/<users_name>/.ssh
 - in .ssh directory
```
    ~/.ssh/id_rsa.pub ==> publickey
    ~/.ssh/id_rsa     ==> private key
```

Note:
-----
in linux home directory is also called as ~

```
Assignment:
-----------
Establish passwordless connection between servers.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++
variables:
--------------------------------------------------------------------------------
-
Variables are memory location which can store some values.

syntax: <variable_name>=<variable_value>
        eg: MYNAME=devops_user

How to print / check / substitute / call values of a variable?
--------------------------------------------------------------

$ (dollar) is mandatory while substituting varaible ==> $<variable_name>

echo $<variable_name>


ex: PLAYER=viratKohli
    echo " my favorite sportsmen is $PLAYER "

Variable types:
--------------------------------------------------------------------
1. Local variables / user defined variables.
   variables created by user


2. system / predefined variables.
   variables created by system
   env ==> command to check system defined variables

user defined variables are only valid till your terminal(session)
 exists or your servers running.

to store variables irrespective of terminal running or restarts

How to set variable permanately in all shell sessions & also server restarts ?
------------------------------------------------------------------------------
Answer: By adding the variable inside startup script ( ~/.bashrc ).

**********IMPORTANT***********************

EXIT CODE
-------------
every command /script generates an auomatic code which signifies,
previous command execution is succesfull or not
syntax: echo $?

if output is 0 zero ==> command executed succesfully
if output is  non-zero ==> Command not executed succesfully
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++

SHELL:
------
1  Shell is responsible to read command provided by user.
2. Shell will check whether the command is valid or not.
3. Shell interpretes (converts) that command into kernel understandable form
& Kernel execute that command with the help of hardware.

Shell + kernel = Linux OS.

Types of shell:
---------------
1. sh    ==> basic shell, earlier / olden days is used in unix   ==> /bin/sh
2. bash  ==> adavnced shell, which is used in linux    ==> /bin/bash
others: c-shell, k-shell etc

Note: echo $SHELL ==> Command to check which shell we are using

Steps to create & execute shell script:
----------------------------------------
step1. shell script will have extension called <script-name>.sh
        vi <script-name>.sh
step2. in the script first line we are going to write is called as shebang
#!/bin/bash
step3. you add your commands (save the script file)
step4.  execute the script
       sh <script-name>.sh  (or) bash <script-name>.sh (or) ./<script-name>.sh

what is shebang line, why its sused?
------------------------------------
shebang line is used to specify interpreter (or) shell to be used for execution
of script

can we run script without shebang line?
------------------------------------------
yes, in that case whatever default shell is there, that will get used.

script1. write a shell script to print date & time.
--------------------------------------------------------------------------------
----------
vi date-script.sh
--------------------------------------------------------------------------------
----------
#!/bin/bash
echo "Current date and time...."
date
save & quit
execute script
sh date-script.sh

script2. write a shell script to CHECK Storage(HDD) of server.
--------------------------------------------------------------
```

```
vi STORAGE-script.sh

#!/bin/bash
echo "Current storage used is ...."
df -kh | head -5 | tail -1 | awk -F " " '{print$5}'

save & quit

execute script
sh STORAGE-script.sh
script3: write a script to create a variable (sportsman=viratkohli)
& verify the variable creation using echo command:
-----------------------------------------------------------------------
[ec2-user@ip-172-31-6-171 ~]$ cat vars_check.sh
#!/bin/bash
sportsman=viratkohli
echo "my fav sportsamen is $sportsman"

script4: write a script to have employee data coming as variables
& verify the variable creation using echo command:
-------------------------------------------------------------------
[ec2-user@ip-172-31-93-35 ~]$ cat employee_data.sh

#!/bin/bash
EMP_NAME=alice
EMP_ID=54234
EMP_LOC=bangalore
EMP_STATE=karnataka
echo "Display employee name $EMP_NAME "
echo "Display emloyee id $EMP_ID "
echo "Display employee location $EMP_LOC "
echo "Display employee state $EMP_STATE "
script5: script to check harware resources:
------------------------------------------
[ec2-user@ip-172-31-6-171 ~]$ cat ram_usage_check.sh

#!/bin/bash
echo "Current free ram available is..."

free -m | grep Mem | awk -F " " '{print $4}'
#free -m

echo "current disk usage is..."
df -kh .

echo "number of cpu...."
nproc
=======================================================================
Note:
-----
- static content (variable key & value stored inside script /file )
= fixed content = Hardcoded == all are same
ex: employee_data.sh
```

```
[ec2-user@ip-172-31-93-35 ~]$ cat employee_data.sh
#!/bin/bash
set -x
EMP_NAME=alice
EMP_ID=54234
EMP_LOC=bangalore
EMP_STATE=karnataka
echo "Display employee name $EMP_NAME "
echo "Display emloyee id $EMP_ID "
echo "Display employee location $EMP_LOC "
echo "Display employee state $EMP_STATE "
Command line arguments
-----------------------------------------------------------------------
 in command line arguments we will give input to the script dynamically

 used to pass dynamic values to script while executing script

 syntax:  sh <scriptname>.sh <argument1> <argument2> <argument3>
 $1 --> 1st argument value
 $2 --> 2nd argument value
 $3 --> 3rd argument value
----------------------------------------------------------------

ex of Command line arguments

[ec2-user@ip-172-31-93-35 ~]$ cat employee_data_2.sh
echo "Display employee name $1 "
echo "Display emloyee id $2 "
echo "Display employee location $3 "
echo "Display employee state $4 "

sh /home/ec2-user/scripts/compute_resource_usage.sh

while executing pass arguments like below format
sh <scriptname>.sh <argument1> <argument2>    <argument3>   <argument4>

[ec2-user@ip-172-31-93-35 ~]$ sh employee_data_2.sh rohit 18976 mumbai
Maharashtra
Display employee name rohit
Display emloyee id 18976
Display employee location mumbai
Display employee state Maharashtra
observations from the above example:
 $1 ==>  rohit   --> 1st argument to script
 $2 ==>  1876         --> 2nd argument to script
 $3 ==>  mumbai       --> 3rd argument to script
 $4 ==>  Maharashtra  --> 4th argument to script
 ---------------------------------------------------------
arithamatic expressions
---------------------------------------------------------------------
we need to use expr keyword to do any arithamatic operations in linux

syntax: expr <value1> <arithamatic_operator> <value2>
```

```
arithamatic operators are
    + addition
    - substarction
    / division
    * multiplication

Note:
 \ (backward slash) is called as escape charecter in linux : it is used
to bypass some functionalities( its used in multiplying with *)
  expr $value1 \* $value2

arithamatic_operators
------------------------
[ec2-user@ip-172-31-93-35 ~]$ value1=20
[ec2-user@ip-172-31-93-35 ~]$ value2=5
[ec2-user@ip-172-31-93-35 ~]$
[ec2-user@ip-172-31-93-35 ~]$ expr $value1 + $value2
25
[ec2-user@ip-172-31-93-35 ~]$ expr $value1 - $value2
15
[ec2-user@ip-172-31-93-35 ~]$ expr $value1 * $value2
expr: syntax error
[ec2-user@ip-172-31-93-35 ~]$ expr $value1 \* $value2
100
[ec2-user@ip-172-31-93-35 ~]$ expr $value1 / $value2
4
[ec2-user@ip-172-31-93-35 ~]$ expr $value1 % $value2
0
[ec2-user@ip-172-31-93-35 ~]$
Note:
-------------------------------------------------------------
value1=25
value2=15

   to find sum of value1 & value2
     SUM=value1+value2
         SUM=25+15
         SUM=40

In all programming language / scripting first execution will always
be R.H.S(right hand side) like how it has been taught in our school
Note:
-----
1. ` ` --> back quoutes, these are used for mathematical opertions
          eg: SUM=`expr $value1 + $value2`
-------------------------------------------------------------------
Hardcoded format
-------------------------------------------------------------------
[ec2-user@ip-172-31-93-35 ~]$ cat arithmatic_op.sh
#!/bin/bash
#Author= bharath
value1=25
value2=15
#to find sum of value1 & value2
```

```
SUM=`expr $value1 + $value2`
echo " sum of value1 + value2 is $SUM "

 Dynamic parsing / Command line arguments parsing
 -------------------------------------------------------------
 [ec2-user@ip-172-31-93-35 ~]$ cat arithmatic_op_2.sh
#!/bin/bash
#Author= vijay
#to find sum of value1 & value2
SUM=`expr $1 + $2`
echo " sum of argument_value1 + argument_value2 is $SUM "
[ec2-user@ip-172-31-93-35 ~]$


[ec2-user@ip-172-31-93-35 ~]$ sh arithmatic_op_2.sh 70 20
echo " sum of argument_value1 + argument_value2 is 90 "
----------------------------------------------------------------------------
Other important dollar notations used in shell scripts:
-------------------------------------------------------
 $# --> To know number of arguments passed to script
 $$ --> To know PID(processID) of the script
 $* --> To print all arguments passed to script
 $0 --> To Print the name of script which i am executing
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```