ROOT = Only one user as ROOT, can we manage linux system
NORMAL = Others are Normal Users
we can nominate few normal users as sudoers without compromising the security.

ROOT -> can control any users on the system

sudoers -> User Management permissions [he cannot add another user, cannot change ROOT user]
- software installations
- process management
- resource management

sudoers
super user commands /sbin with sudo privilige

How to promote a Normal User as Sudoer?
There is a configuration file /etc/suoders in this you can add Normal Users to promote as sudoers.
There are 2ways to do this.

sudoers =Group

sudoers
username

group
sudoers

joe
1. add joe in /etc/sudoers
2. assign joe to sudoers group

sriman$ sudo su - (or) sudo su root
sriman$ sudo su joe
joe$ sudo su sriman

1. ROOT = There can be only one root user can exists in Linux System.
2. NORMAL

What is the different between a ROOT AND SUDOERS?
A ROOT user can perform any operation on the linux machine like adding a user or removing/ disabling the user , but a sudoer can only perform super user commands only cannot change any other user on linux system.

There is a configuration file /etc/sudoers to promote a NORMAL user as a sudoers.
1. add the user or a group into the above file
2. add the user to the sudoers group.

he can execute any command use sudo.
sudo = super user do - Run a command as a super user previlige

sudo su - = root user = A sudoer can execute this command and can switch as root user or not?

Linux permits a user to login/access the computer from remote machine as well using some technics like SSH.

joe   ALL=(ALL:ALL) ALL
1st ALL = indicates joe can access the machine and can grab sudo from any host.
2nd ALL = indicates joe can run any commands on behalf of any user in the system.
3rd ALL = indicates joe can perform any commands on any group of users.
4th ALL = he can perform all the above on all the commands.

How to add users?

useradd username
sudo adduser --uid 2002 --home /home/rockhome rock

sudo adduser --gid 2003 bob = this will directly adds the bob to hr group rather than creating an bob group.

userdel username = deletes a user

How to add groups

groupadd groupName
groupadd -gid 1002 groupName

groupdel groupName = deletes the group.

Want to see a user in which groups?
groups username = this shows a user is in which groups

I want to see the details of a user.
id username = shows the information of a user

passwd = is to reset the current system password of the user after login.

How to modify a user
usermod -aG sudoers rock

How to see all users
cat /etc/passwd

How to see all groups
cat /etc/group

File Permissions

adduser username
adduser --uid id username
adduser --uid id --gid gid username
userdel username
usermod -aG groupName userName

id username
groups username
cat /etc/passwd

groupadd groupName
groupadd --gid id groupName
groupdel groupName
cat /etc/group

passwd
sudo = to run the commands with super user previlige

Files and Folder permissions

Linux is an Multi-User operation system where various different users can parallely can use the machine and underlying resources. It is the basic requirement of Linux to built-up some security features in handling and protecting the underlying resources of the users when multiple users are using the machine.

One of such system resources the users will use in linux is Files and Folders.
A File or Folder that has been created by a user should be protected to be accessed/modified by any other user of the Linux System. This can be achieved by Using Files and Folder Permissions provided by Linux.

Files and Folders Permissions.

Linux has comeup with 3 levels of managing the File permissions.
1. Owner = Creator of the File/Folder
2. Group = The group to which the file belongs, those user permissions.
3. Others = who is not a user and does belongs to the file group is called others.

Permissions
1.read
2.write
3.execute

The files/folders we create in linux by default we dont need to assign permissions, Linux takes care of granting permissions by default.

File:-
1. read = we can see the contents of the file, cat, grep, wc, vi
2. write = we can change the contents of the file or can delete as well.
3. execute = if the file contains programming instructions then we can run it.

Folder:-
1. read = we can list or see the contents of folder (ls). But we cannot change the folder contents
2. write = we can add/remove files and folders inside that folder
3. execute = we can cd into that folder

touch story.txt
ls -l story.txt

-rwx-rwx-rwx bob hr created date last modified size story.txt

typeofthefile
d=directory
c=character
b=blockfile
-=regularfile

When we create File/Folder in Linux by default it creates the File with owner as user who created and group as group of the user with default permissions.

How can i see the owner, group to which it belongs and permissions of a file/folder
ls -l filename/foldername

FileType Permissions links ownername groupName size last modified date filename/foldername
[owner-group-others]

Can we change the default permissions to the files/folders of linux?
chmod = is used for change mod of the file/folder

There are 2 notations we can use in modifying the permissions of a file/folder.
1. octal notation
2. symbolic notation

1. octal notation = For every permission there is an numeric number assigned to identify the permission you want grant. Now denote the number or sum of these to give permissions to three levels like owner/group/others.
read = 4
write = 2
execute = 1

0 - 7 = 8 = octal notation
rwx rw rw

chmod 764 filename

chmod [owner][group][other] file
chmod [owner][group][other] folder = only applies to that folder not sub files/directories
chmod -R [owner][group][other] folder = recursively apply these permissions to sub files/folders including that folder.

octal notation will not permits to change permissions to the specific level,we have to always assign permissions to all the 3 levels of users.

2. Symbolic Notation = Instead of using numbers to grant permissions there are predefined symbols using which we can grant permissions to owner/group/others. You can only reset permissions of a specific level also.

permission symbols
---
read = r
write= w
execute=x

level symbols
---
owner = u
group = g
others = o

granting symbols

+ = add
- = remove
= = reset

chmod o+x stories.txt
chmod g-wx stories.txt
chmod u+rwx stories.txt
chmod u+rwx g+rw o+x stories.txt
chmod o=r stories.txt = remove existing and only assign read permission

touch stories.txt

There are 2 notations are available for chaning the file/folder permissions using CHMOD command
1. octal notation
2. symbolic notation

1. octal notation:- In octal notation we assign a weighted value for each permission from a number ranging between 0 - 7 where zero indicate no access and 7 being the highest granting full access. As 0-7 has 8 number it is being called octal notation.
read = 4
write = 2
execute = 1

For each group [user/group/others] of users assign an appropriate weighted number (or sum of these numbers) depends on the permission you want to grant.

chmod [user][group][other] file
Note:- if you are the owner of the file then you can change the permissions. You cannot change other users files unless you are a sudoer/root

chmod -R [user][group][other] folder = -R stands for recursively apply the permissions

In octal notation we cannot add/remove a permission for a individual levels always we reset permissions for all the three levels of users.

Symbolic Notation:- There are predefined symbols using which we can grant access to the files/ folders.
r = read
w = write
x = execute

u = owner/user
g = group
o = others
+ = adding permission
- = removing permission
= = resetting the permissions

chmod [u/g/o] [+/-/=] [rwx] file
chmod u+rwx,g+r,o+r secret.txt

chgrp groupName filename = to change group of the file
chown ownername fileName = to change owner of the file.

owner = sriman, joe, bob
group = hr

sriman, joe = hr
bob
others

chmod [user][group][others] fileName
chmod -R [user][group][others] folderName

r = read, w= write,x =execute
u = owner, g = group, o = other
+ = add, - = remove, = = reset

chmod u+rwx fileName

chown [user] fileName
chgrp [group] fileName

chown -R [user] folder
chgrp -R [group] folder

chown [user]:[group] fileName
---
umask = is a variable available per linux system user based on which his files/folder permissions will be computed and assigned by default.
The default umask value is 002. By using this value linux will compute the permissions of a file or folder while creating by the user.

Max UMask of a File = 666
Max UMask of a Folder = 777

666 - 002 = 664 = permission of the file
777 - 002 = 775 = permission of the folder rwx-rwx-rx
---
user/system commands

Current User of the Machine?
whoami = gives you current user of the machine
logname = current user of the machine.

Who are logged in to the Linux System
who = shows all the users currently accessing the machine including order/logged-in time
users = only gives space separated usernames currently logged in.

uname = operating system name
uname -a = gives full details of the operating system.

hostname = gives name of the computer
hostname -i = ip address

ls -l | more = displays by page the list of files.

man command-name = man stands for manual and can get information about any linux command.
---
networking commands
---

sudo apt-get install -y net-tools = install net-tools package for using ifconfig command
ifconfig = displays the ip address and network adapter information.

ping serverName = checks the communication line between client and the server and rate of transmission.

free -h

umask = it is variable in linux per user through which the default file/folder permission will be calculated.
0002=default value
max umask for a file = 666
max umask for a folder = 777
---
whoami = current logged in user information
logname
users
who = both commands returns currently logged in users of the linux
hostname = name of the computer
hostname -i = ip address
ifconfig = net-tools package should be installed
uname -a = platform
ping = to reach the destination server
free -h = ram space
top = show all the current running process of the machine.
---
cat /proc/cpuinfo = file contains the information about the processor of your machine.
lshw =list hardware

reboot = reboot computer
reboot -f = fast reboot

init 0 =power off
init 6 = restart
---
df -k
du -sh filename
mount = mount points
---
process related commands

free -h [human understandable]
du -sh fileName = disk usage of the file.
df -h = disk free
cat /proc/cpuinfo = shows the cpu information
lshw = list hardware

reboot
reboot -f
init 0 = shutdown
init 6 = reboot

---

process related commands

top = to see the process running on the linux system.
PID = process id
USER = owner
PR
NI = Nice Value
VIRT = virtual memory
RES = reserved memory
SHR = shared memory
S = status
%CPU
%MEM
TIME+ COMMAND = command used to run the process

status can be of 5 values
D = uniterruptable program
S = sleeping
R =running
T = traced/terminated
I = idle
z =zoombie

You should press q to stop the Top

---

ps = process status
ps -ux = current user running process including background process
ps -ef =shows all the user process running on the system.
ps -f pid = show only that process information

if i want to launch a process with a specific nice value use. The default nice value is 0
nice [nicevalue] command

how to change the nice value of a running program
renice [nicevalue] pid

How to kill a running process
kill [PID]
kill -9 [PID] = forceful termination

There are 2 types of process are there in linux
1. foreground process = is process that blocks the user until termination. we can terminate a foreground process by pressing ctrl+c

2. background process = shell will not be blocked when we launch the process as background.
How to run a process as background.
processname &
nohup processname

What is a process?
Program under execution is called a process.

How does linux will keep track of the information about the running programs?
For every program under execution linux will keep the information about the program and the resources it is consuming in a process file that is stored in /proc directory. For every process to identify it assigns PID for the process.

Who has to create a process or how to create a process?
When we launch/run a program in Linux, operation system takes care of creating a process representing the program we are running. We dont need to manual create a process.

How do i know which process are running in my Linux machine?
There are 2 ways.
1. top
2. ps

top = top of execution/stack [realtime]
It shows the current running programs on the linux machine. displays the below information as a table.

USER PID PPID PR NICE VRM RESMEM SHR S %CPU %MEMORY TIME+COMMAND

USER = Owner of the process
PID = Process ID
PPID = Parent Process ID
PR = Priority
NICE = CPU Scheduling
VRM = Virtual Memory
RESMEM = Reserver Memory
SHRM = Shared Memory
S = process status
%CPU% = percentage of cpu used
%MEMORY% = percentage of memory used
TIME+COMMAND = Shows running time + command used for launching the program.

STATUS

D = UNITERUPTED PROGRAM
R = RUNNING
S = SLEEPING
T = TRACE, TERMINATED
Z = ZOOMBIE
I = IDLE

NICE = Its an numeric number with in the range of -20 - 19 based on which the cpu will be allocated to the process.
The lowest the nice value is the high in priority in executing the program. The default nice value is zero.

nice -n -20 command
renice -n 9 pid

---
priority =PR
Based on the priority of the process the system cpu will be allocated. User cannot assign priority for a process directly.

The priority of a process ranges from 1 - 139
The system space takes 1 - 99 priority [reserver priority]
The user space process can take from 100 - 139

The priority of a process is computed through nice value.
PR = 20 + [-20 - 0 + 19]
PR = 100 + 0

PR = 20 [0]
PR = 100 + 20 = 120

PR = 20 + [19]
PR = 100 + 39 = 139
---
ps = process status

ps = that shows the process running in the current terminal window.
ps -au = -a stands for all the process of all TTY, -u = user name
ps -x = current running process of the user
ps -ef = -e =everyones or all user process and BSD Format

ps -eo pid,ppid,command,time = -o output only these columns

---
how to kill a running process.

kill pid
kill -9 pid

How many types of process are there in Linux?
There are 2 types of process are there
1. foreground process
2. background process

There are 2 types of processes are there in linux
1. Foreground process = upon running a program on a shell by default the process executes as a foreground process,this means it blocks the shell and allows the user to interact with the program. We can terminate these programs by pressing ctrl + c

In a foreground process if we close the terminal window linux will send an hangup signal to the program asking to terminate so that program execution will be stopped and quit.

2. Background process = Background process returns the terminal window of the user and doesnt allow the user to interact with the program. The only way to monitor the program is using top/ps and to stop execution we need to use kill.
There are 2 ways in runing a program as a background program.
1. command & = this makes the program run as background

2. nohup command = nohup stands for no hangup. So even we terminate the shell the nohup will not passes the hangup singal to the command making it run for long time.

The difference between nohup and & is
bash calc.sh & = the output will be redirected to standout and if we close the console the output will be lost.
nohup bash calc.sh & =nohup automatically redirects the output of the background process to the default file called nohup.out.so that we see the output of the program.

nohup will not runs the program in background it helps only in redirecting the output so the best practise of running background program is nohup command &

---
package managers in ubuntu linux.

there are series of commands we need to execute to install s/w

package managers

1. How does the software applications developed by a developer is distributed to the end-user?
It should be packaged as a single file distribution by binding metadata about the software application containing instructions/information related to that software
1. author
2. organization
3. licensing
4. dependencies packages required.
5. system requirements
6. instructions on how to run the software.

2. Why we need to package and provide metadata information?
package = easy distribution of the software
metadata = since the end-user dont know how to run the application we bind the information in the software package so that operation system can read this information and can execute/run the application.

3. How can the end-user can run the software package?
The end-user cannot directly run the software package.
Running it directly might take more diskspace in extracting the package and takes more time in validating and executing by reading metadata.
Thats where it should be installed. Operating systems provider package managers responsible for extracting the package and copying to the disk location. They validate whether software meets the requirement of the system and registers the metadata of the software into system registry.
So that everytime in order to run the software it gets executed out of registry using the copied contents from the disklocation.

The software package and package managers differs from operation system to another.
For e.g.. in windows the software package is ".msi" and windows installer is the package manager who extracts the contents and registers.

kernel = interacts with the hardware of the computer. in kernel there is not concept of software packages and package mangers. How to run the software applications then?
You have to manual copy the software programs we want to run and need to directly pass instructions to the kernal running it. If he is a developer then he finds it easy to run manually.

To reduce complexity of running applications on kernal by end-user Linux distros added software packages and package managers.
1. Debain
Software package standard :- ".deb"
Package Managers:- "apt" and "dpkg"

2. Redhat
Software package standard :- ".rpm"
Package Managers:- "yum" and "rpm"

In Linux there are different packaging standards and package managers exists specific to the distro's, why?

Anything that comes out of Kernel becomes common across all the distro's. Unfortunately the kernel has not defined any packaging standard or package managers to install/run a software application.
The developer/end-user has to manually perform all the operations inorder to use/run the software application directly on the Linux Kernel. This may not makes the Linux end-user friendly operation system.

The Linux distro's identified this gap and has comeup with their own packaging standard and package managers for facilitating the installation of the software packages on their distro's

Ubuntu distro
package standard = ".deb"
package manager = apt and dpkg

Redhat distro
package standard = ".rpm"
package manager = yum and rpm

There are 4 types of Software Repositories provided by the Distro
Ubuntu/Redhat/Slackware/Arc

Main – Canonical-supported free and open-source software.
Universe – Community-maintained free and open-source software.
Restricted – Proprietary drivers for devices.
Multiverse – Software restricted by copyright or legal issues.

Linux Registries/Repositories can be classified into 4 types
1. Main = Cannonical andOpensource Official Software of Distro
2. Universe = Community Driven Software Packages of open source
3. Restricted = Device Driver software
4. MultiVerse = Licensed

How to install software package from repository?
1. goto Repository
2. search for the software package you are looking for
3. download the software package onto your local system
4. Install using Installer.

End-User should know the software distribution model of Linux Operation System.
Every Distro provided Package Managers of their own.
Ubuntu = apt
Redhat = yum
arc = pacman
fedora = dnf

apt = advanced packaging tool
we goto "apt" package manager asking for installing a software package.
1. It goes to repositories searching for software package
2.download it into local system, checks for dependency packages of this software
3.install it by calling dpkg (debian package installer).

apt-get = downloading software packages and installing
apt-cache update

These 2 tools has plenty of options which a developer find it too difficult or overwhelmed = These 2 tools are wrapped inside one tool called "apt"

apt = both the essential options of apt-get and apt-cache are being added into apt tool.
/etc/apt/sources.list
---
sudo apt update = used for cache the remote repository software index locally
sudo apt upgrade = rarely required unless you want to upgrade software packages of your system
sudo apt install packageName
sudo apt search packageName
sudo apt remove packageName
sudo apt list --upgradable = shows list of packages upgradable

jdk1.8 -installed /uninstall
jdk 1.9 = latest = install

apt-get = dowload and install software package
apt-cache = creating local cache of the repository and search, update functionalities

apt = simple tool with handful of options in managing software packages.

sudo apt update
sudo apt upgrade
sudo apt --list upgrade
sudo apt install packageName
sudo apt search packageName
sudo apt show packageName
sudo apt remove packageName

---
apt add-ppa .ppa

There are 2 reasons why we use PPA.
1. If a higher version of software package has been released and is not available in current version of linux repository we can add that higher version using PPA
2. private repository of the software vendor can be used through PPA.

sudo apt-add-repository ppa:ppaname
sudo apt-remove-repository ppa:name
sudo apt-get install purge ppa:name

#1 how to install open source software packages distributed through linux distro repositories
#2 proprietary softwares distributed by vendors directly
#3 how to add custom repository and install software.

While installing through dpkg if there is a failure due to dependent libraries missing we can fix it using the below command.

sudo apt --fix-broken install

deadsnakes.list
ppa
- repository
- repository

/etc/apt
|-sources.list
-- add entry into this file about your ppa repository

|-sources.list.d
|-add an file into it containing information of your repository
---

There are two types of software distribution are there
1. packaged distribution = These are softwares packaged into single file with metadata attached that can be downloadable and installable.
1.1 Open Source [available through repositories]
1.1.1. use apt tool
1.1.2 add ppa repositories

1.2 Licensed [direct downloadable] = dpkg

2. binary distribution = These are software applications being shipped as single compressed files as (.tar,.tar.gz) files which doesnt have metadata and cannot be installed.

There are two types of software distributions are there.
1. Packages Software distribution = is an installable software. = apt
1.1 Open Source/Community driven softwares
1.1.1 Ubuntu/Community Distributed
1.1.2 Private Repositories through PPA

2 Licensed = Installable by directly downloadable = dpkg

2. Distributable Software distribution
Binaries compiled to platform independent and packaged and distributed as .zip, .tar.gz

Environment Variables

In linux at shell level or bash level we can create variables with name and value and will be retained till the end of the terminal.

What is a variable?
It holds value with a name so that we can access the value by using name.

literal value = direct value
10 = Integer
20
"Christopher" = String

A="Christopher"
to a value we are giving name like "A" above. Now we can use instead of value Name of the variable to usethe value.
Variables are placeholder for storing values.

We can define variables in SHELL,how to define.
export VARIABLE_NAME=VALUE

echo $VARIABLE_NAME= $indicates what next to me is not a text rather it is a variable try to get the value and print.

There are two types of environment values.
1. User Defined Variables = are created by user and will be removed at the exit of the terminal
2. System Defined Variables = These are created by operation system by default during the startup of the linux.

env = shows all the environment variables

In Linux there is an special variable called "PATH",gets created by default by linux. The value of this variable is special.
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin

When we execute linux commands, for every command the corresponding code is stored in a file respectively and located in /bin.
cp = cp file is there in /bin
cat = cat file is there in /bin

So when we run a linux command at the command-prompt linux quickly goes to /bin directory searching for the file and executes.
The /bin and /sbin are the default directories into which linux looks for executing the commands.

What are environment variables?
Environment variables are shell or bash variables that are local to the bash terminal and are used to store values inside them.

How to create an environment variables?
export VARIABLE_NM=VARIABLE_VAL
The VARIABLE_NM will be defined with capital letters as a standard convention.

How to access the value of variable.
You need to use $notation of accessing the value $VARIABLE_NM

There are 2 types of Environment Variables are there
1. System Variable = pre-configured variables that comes out of linux install and will be created during bootup of the linux operating system. These variable and values are used specially by Linux

2. User-Defined Variable = The variables created by the Linux user are called User-Defined variables.

Out of the System variables created by Linux there is one special variable called "PATH". PATH variable carries the value as location of directories. by default PATH variable points to the following locations.
PATH=
/usr/local/sbin:
/usr/local/bin:
/usr/sbin:
/usr/bin:
/sbin:
/bin:

In the above PATH variable is not having value rather has directory locations as values what is the significance of this?

cp, cat, mv, mkdir, rm = is an executable file stored on the filesystem location of the Linux. Linux inorder to execute these files/commands will look for their directory locations by using the PATH variable values.

When we issue a command linux has to execute a file, but how does linux knows the directory location of this file?
It uses PATH variable values.

distributable software
~/apache-tomcat-9.0
|-bin
|-startup.sh - For starting the tomcat server.

cd ~/apache-tomcat-9.0/bin
$./startup.sh

cd ~/books
startup.sh

by default linux looks for an executable file under current directory. If not located the goes to

PATH variable.

Now to run startup.sh you have two options.
1. always navigate to ~/apache-tomcat-9.0/bin
2. add the ~/apache-tomcat-9.0/bin directory location to PATH variable.

export PATH=~/apache-tomcat-9.0/bin [this overwrites and leaves the path in-correct]
always append
export PATH=$PATH:~/apache-tomcat-9.0/bin

---

bash profiles

A environment variable is always local to the shell/bash. If we want this to be available for multiple bash sessions we need to create it manually in each bash session.
This is by nature will be applicable for SYSTEM / USER-DEFINED Environment variables as well.

But we wanted these environment variables to be persisted for multiple bash sessions how is it possible?
Linux has offered solution for this using bash profiles.

Linux throws set of files to us asking to configure the variables with values you want bash shell to be created.
There are multiple files are available serving different purposes.

/etc/profile = admin level
~/.profile = graphical user interface
~/.bash_profile = interactive shell terminal
~/.bashrc = non-interactive shell
~/.bash_logout = when exit from terminal

symbolic links = These are pointers to the original files using which we can access the contents of the original file.

How many types of symlinks are there?
There 2 types:
1. softlink
2. hardlink

softlinks
1. we can create softlink for both files and directories
2. softlinks can be created across the file system.
3. INode address of the original and softlink file are different
4. Permission for soflink file is different from original file

softlink file has its own permissions -> hardlink file has its own permissions
softlink if you have access , you cannot original file

permissions are softlink means can you see it or not
operations on the softlink depends on original file permissions.

write ->
ln -s source ryhmes.txt

5. Softlinks are broken we the original file has been moved/renamed

hardlinks
---
1. hardlinks can be created for only files but not directories.
2. Hardlinks can be created for the files which are on local file system only.
3. Original and Hardlink file has same INode address
4. File permissions are same for both
5. Those are not broken even original file has been moved/renamed/deleted.

ln -s sourceFile lnkName
alias names to the original files so that application can access the original files through symlinks.

/home/sriman/
|-stories
|-kids
|-infant
|-ryhmes.txt
|-story1

/home/sriman
directory -> kids -> /home/sriman/stories/kids

cd kids
pwd -> /home/sriman/kids
ls ->contents of original directory /home/sriman/stories/kids

export VARIABLE_NM=VARIABLE_VALUE
2 Types
1. User Defined Variables
2. System Defined Variables

PATH = Holds the directory locations of the applications/command files.
export PATH=$PATH:directory

export ZIP_LOC=03930

How to make environment settings/values permanent?
Linux has thrown set of files in which we can define the environment settings, so that it executes all of the settings configured in these files while opening the terminal/bash shell.
The files into which these settings goes are called BASH PROFILE.

#1 /etc/profile [admin level] = globally applied to all the users of the linux
#2 ~/.profile = Graphical User Interface these will be executed
#3 ~/.bash_profile = Interactive Terminal these settings will be configured
#4 ~/.bashrc = non-interactive termainl
#5 ~/.bash_logout = exit and logout from terminal

---
symbolic links/sym links in linux

These are the pointers to the file we create referring the original file.
There are 2 types of symbolic links are there.
1. softlinks
2. hardlinks

touch ~/stories/books.txt
~/library$ ln -s articles.txt ~/stories/books.txt
ls -l
articles.txt --> ~/stories/books.txt
if we dont use "-s" option it becomes hardlink

What is the difference between softlink and hardlink.

Softlink
1. Softlink can be created for files and directories
2. Softlinks can be created across the filesystem.
3. INode address of the original file and symlink file is different
4. File permissions are different from original to symlnk
5. soft symlinks are broken if you delete or move the original file location

HardLinks
1. Hardlinks are only allowed to files but cannot be created for directories.
2. Hardlinks should be pointed to the local filesystem
3. INode address of original file and Hardlink file with be same.
4. File permissions will be same for both.
5. Hardlinks are not broken if you delete original file or move original file. Because HardLink is a copy of original file.

How to do text processing or data extraction on Text Files in Linux?
Linux has provided few shell commands/utilities to handle this.
1. cut
2. paste
3. awk
4. sed

How to process text content in a File using Linux commands.
There are text processing commands are available in Linux, those are
1. cut
2. paste
3. awk
4. sed

1. cut
cut is command that takes the File as an input or standard STDIN and will perform operation. We need pass the delimiter to cut asking to break the data into columns/fields based on delimiter and helps us in accessing the data from the file.

syntax:
cut -d "delimeter" -f number fileName [FILE INPUT]
cat filename | cut -d "delimiter" -f fieldNo [STDIN]

2. paste
paste is used for merging the contents of 2 files line by line based on delimeter.

books.txt
isbn102,practicle approach to devops,200
isbn938,linux kernal,1200
isbn039,shell scripting,900

books_stock.txt
isbn102,2
isbn938,10
isbn039,12

paste -d ":" books.txt books_stock.txt

This combines both the files line by line by using "," between the two lines.
the result.

isbn102,practicle approach to devops,200,isbn102,2
isbn938,linux kernal,1200,isbn938,10
isbn039,shell scripting,900,isbn039,12

3. awk
cut only accepts delimeter but in awk we can use pattern to break the data into fields. The default pattern used by awk is "space"

products.txt
1,led 32inch tv,7000
2,refrigerator,12000
3,washing machine,22000

awk '{print}' products.txt = prints by default all the lines one by one
awk -F "," '{print $1 $2}' products.txt
Here -F stands for field separator and using print $1 or $2 you are extracting the specific fields of data by using Field Separator and printing.

cut -d "," -f 1,2 products.txt

1, led 32inch tv
2,refrigerator
3,washing machine

awk -F "," '{print $1 $2}'
1led 32inch tv
2refrigerator
3washing machine

employees.txt
100,smith,12000,new jersey
101,susan,12200,new jersey
102,rock,8000,atlanta
103,james,6000,mineapolis
104,mathew,9000,hyderabad

awk -F "," '{print $2}' employees.txt

awk '/new jersey/ {print}' employees.txt = print all lines matching with word new jersey
awk -F "," '/[7-9]000/ {print $2}' employees.txt = print 2 field of every line matching between salary 7000-9000
awk -F "," '/s.*/ {print $2}' employees.txt = print 2 field of every line based on Field Separator and contains "S" letter

What is an operating system?
#its a layer between the end-user and the hardware of the computer which helps us in interacting with the hardware of the computer.

Why i cannot directly program the hardware why i really need an operating system?
The hardware instruction set provided by the vendors (assembly level language) is different from one vendor to another, so an end-user should program the instructions to the hardware and cannot change it. Instead use operating system, it provides system routines that can be used for programming the instructions, so that operating system takes care of translating the system routine instructions into hardware instructions dependents on the vendor. This makes our instructions portable across mulitple hardware.

What are high level programming languages,why do we need to use them?
While we write programs using System Routines of the operating system that program only works on that operating system and cannot be used on another since System Routines are different from one operating system to another.
To achieve operating system/platform portability programming languages comes into picture.

Programming languages provides their own english like language instruction set.programmers writes their instructions using high level language instructions. Operating Systems can only understand System Routines,so these high level language instructions should be translated into operation system routines that where programming languages provides compilers/interpreters. They provider compilers or interpreters for every platform (windows/linux/mac).

Hardware portability = operating system
Platform portability = programming language

Linux is an operating system = it is helpful for achieving hardware portability, and interacts with the underlying hardware of the machine.
Linux has a key component called "Kernel" that defines the operating system.

UNIX = commercial source/runtime
LINUX = free distributed and freely modifiable (open source) = "Kernel" = Command-Line Interaction

There are few folks understood it is not user-friendly to Linux Kernal and started a project called GNU Linux
open source software utilities (GUI), (Network) (System Software)(Software Utilities)
In GNU Linux the started producing/distributing software components for Linux Kernal like GUI, NETWORKING, SYSTEM SOFTWARES AND SOFTWARE UTILITIES

- Linux Kernal [install]
- download software built as part of GNU Linux and install on Kernal

Linux Operating System (CUI) (difficult) -> Hardware (difficult)
GNU Linux (any one in the world can produce software for kernel)
- GUI

End-User (Programmer)
Install Linux Kernal
Download GNU Linux Software Utilities and install them on Kernal

There are few software manufacturing companies came forward and started distributing linux as

a single piece.
Linux kernal + GNU Linux Software utilities and provided together. Linux Distro

www.leawy.com/drive

What is an Operating System?
- acts as an interface between end-user and hardware of the computer along with that it helps us in achieving the hardware portability.

What is Programming Language?
- programming languages provides english like instruction set that can be used for programming the computer easily. They provide compilers for every platform so that the instructions can be translated into platform executable instructions and can be ran. So that we can run/execute a high level language instruction set on multiple platforms by compiling with platform compilers.

What is Linux Operating System?
General Purpose computer operating system
Release - 1991
Linus Trivolds

Kernel? = core component of linux operating system that interacts with the hardware of the computer.

Linux is defined by "Kernel", and it is not user-friendly and provides CUI for interacting with the Hardware of the computer. To use Linux one should know the commands and looks like he should be "Programmer". This makes Linux difficult to use and not categorized as a General Purpose operating system.

Now the world began to think about how to make Linux easy to use as a End-User Operating System, so an GNU Linux Initiative has begun. GNU (General Public License). Any software that is manufactured under GNU Linux can be freely distributed and can modified.

Lot of developers around the world started producing different software utilities that can be combined with Kernel that makes it easy to use.

How to use Linux?
Install Linux Kernel and Install GNU Software utilities to make it easy to use. Forget about using Linux first ofall installation itself is difficult. to make linux easy to install and use Linux Distro has been released.

Linux Distro?
There are few Software Organizations came forward and started building Linux with GNU Software Utilities and provided as a Linux Operation System. Once we install Linux Distro we get all the things making it easy to use.

Linux Distro = Linux distro comes with Linux kernal packaged with Software Utitilies, GUI and package managers that makes linux easy to use.

how to install software in windows operating system.
.exe file and run it
Windows Installation Manager = Installation Wizard

Linux Distro = Linux kernel sourrounded with set of GNU Software utilities and package managers with different configuration files.

Package Manager is similar to an Installer in windows that is shipped as part of the operating system. It facilities in extracting and installing the software application packages that are distributed. Software application packaging depends on the linux distro because the package manager differs from one linux distro to the another.
For e.g..
Redhat Enterprise Linux = rpm (redhat package manager), packaging = .rpm
Debain Linux = dpkg (debain package manager) , packaging = .deb
---

There are 4 major/popular Linux Distro are available in the market
1. Arc
2. Debain
3. Redhat
4. Slackware

The 2 most popular Linux Distros are
Debain
1. ubuntu (we use)
2. linux mint
3. kali linux
4. elemenary os

Redhat
1. CentOs
2. Fedora
3. Redhat Enterprise Linux

Features of Linux Operating System.
1. Portable = Linux works with various different hardware configurations of multiple different vendors.
2. Multi-User / Multi-Programming = Multiple Users can parallely use the Linux Machine by logging in and supports running Mulitple programs parallely/simultaneously dependes on Hardware configuration.
3. Open Source = Free distributed and Freely modifiable.
4. Hierarchial File System = Due to hierarchial file systems storage and retrieval of files is faster and efficient.
5. Shell = Interpreter program that helps us in interacting with kernel of linux operating system.
6. Security = Linux has been built from its roots keeping in view of Security. It has built-in authentication and authorization support that helps us in controlling the access to the machine at various different levels.

What is Linux Distro?
The kernel sourrounded with various different GNU Sofware Utilities, package manager and system configurations together is called "Linux Distro"

There are 4 major linux distributors are available
1. Arch
2. Debain
3. Redhat
4. Slackware

2 popular distributors are
1. Debain
1. Linux Mint
2. Ubuntu
3. Kali Linux
4. Elementary OS

2. Redhat
1. CentOS
2. Fedora
3. Redhat Enterprise Linux

What are the features of Linux Operating System?
1. Portable
2. Multi-User and Multi-Programming
3. Open Source
4. Hierarchial FileSystem
5. Shell = using which we can directly interact with Kernel of Linux
6. Security

What is a File System?
to know what is it, through assumption.

Linux File System
What is a File System, what is the purpose of it?
Storing and organizing the data on the physical hardware of the computer.

How does the data is stored on the storage devices of the computer?
We cannot store directly the character/symbol representations on the Storage device of the computer. Storage devices are mechanical/magnetic devices that can only store zero's and one's. So we need to convert character representation of data into binary format using charset [Character set encoding standards].

There are lot of character set encoding standards are available one of them is ascii.
ascii charset has provided decimal/binary representation for 256 characters from 0 - 255. It supports only english language characters only. A character in ascii takes maximum 8 bits for storing on storage device.
All the characters on the storage device will be stored in 8 bit length only.

What is File System, what is the purpose of it?
File system is a techinic or a way of storing and organizing the data on the physical storage devices of a computer like
1. Harddisk
2. Pendrive
3. CD Rom

How does the actual data would gets persisted on these storage devices?
The storage devices are electro magnetic or mechanical magnetic devices and they can only carry 0's and 1's. so, the data would be stored in binary format only.

Human data is not binary format it will be in character/symbol representation, then how the symbols can be stored in binary format on the physical devices?
The human characters/symbol representations has to be converted into decimal and an binary equivalent of the decimal should be stored on the physical devices. This technic is called encoding. For this there are standard charset encoding techinics are available to ensure all the characters are encoded with the same numeric values so that the information can be carried across.

What are the various charset encoding standards available?
1. ASCII
2. UNICODE
3. UTF-8
4. UTF-32
5. ISO-9001

Why we need several charset encodings what is the difference between them?
Different charset encodings supports different languages, so we need multiple charset encodings.

What is ASCII how many languages it supports?
ASCII supports english language only characters with some special symbols. Total it supports 256 characters. The largest number for representing a characters is 255 and requires 8 bits. So to encode and decode the data all the characters from 0- 255 are stored in 8 bit length on the storage device.

If every character has not stored with 8 bit size we cannot interpret the data while reading from the physical location since we dont know which character is of what length bits.

Why cant we have one charset representing all the languages instead of many charset standards?
As we learnt from earlier if a charset supports lot of languages it needs more bits for representing the character on the pysical device.this eventually leads to wastage of memory when we are using only a specific language.

What is file system?
File system are the mechanism / techinic used for storing and organizing the data on the physical storage device of a computer.

File systems organizes the data on the storage devices interms of files and folders.
What is File?
File is a data structure in which we store the address locations of the actual data where does it resides on the storage device.
(or)
Its a pointer to the actual locations of the data on storage device.

If we have several data representing their locations on the storage we need multiple files.To identify which data is represented by a file we need to define a unique name for a file.
We can access the actual data on the storage by referring the unique name of the file.

If we store the address locations of the data in a file, we endup wasting lot of memory locations in keeping the address of the data locations. This is not efficent.

File is an datastructure in which we hold the physical address location of the actual data resides on the storage device.
If we hold address locations of the memory in a file directly we endup in storing address in memory than actual data and storage space will not be used efficiently.

To avoid the above problem the physical storage has been divided into blocks of sequential memory locations and block are assigned to the files and block no's are stored in file to track the information.

Each block size is fixed so that we can keep block no in a file as a pointer and can save memory for storing the blocks information. For a given file multiple blocks can be assigned depends on the information/data we want to store.

A file always holds array of block no's pointing to memory locations of the data.

There are multiple File Systems are available in storing and organizing the data on storage devices. The techinic/mechanism or algorithmic approach of organizin and storing the data on the storage device will be different from one filesystem to the another.

We can say File System is an integral part of an operating system. Since an operating system should be able to store the data on the physical storage device based on the File System Technic.
Usually we pick a File System while installing the Operating System because of the above reason.

We can say indirectly Operating System Vendors comesup with FileSystem Technics as those are integral part.

Windows Operating System
- FAT32 (File Allocation Table)
- NTFS (Network File System)

MAC
- HFS (Hierarchial File System)

Linux
- ext - 1st File System [very old not being used doesnt support huge storage space]
- ext2 - 2nd File System - performance [more number of smaller size files make it in-efficient to access]
- ext3 - recovery is not supported
- ext4 - Journal concept for recovery of the file system [current generation]
- btrs
- jfs
-xfs

File System = File System is a technic / mechansim through which we can store and manage the data on a physical storage device of a computer.
File Systems enables us in storing the data interms of files/folders so we can keep track of the information and can access it easily from the Physical storage device of the computer.

Physical storage memory is partitioned into two areas basically for keeping track of the data.
1. File System memory = Here the information about the files and their attributes will be tracked. Again we can divide File System Memory into 2 parts.
1.1 File Blocks = For every data we store on the storage device, we need to track the block no's of the memory where the data has been stored. To maintain these block addresses We create a File block in this memory area and store information about block nos of data along with File attributes (owner, created date, last modified, charset)

1.2 File Index / File Table = It is a Map like datastructure in which we hold every file with respective filename and file block no of that file.

2. Data Memory=The actual data is persisted in this memory. (User usage memory)

Who works with the File System?
Always the physical storage devices of the computer are managed through operation system. So the operating systems come up with a FileSystem mechanism to store and manage the data on the Physical storage device.
From operating system to another the File System technic would differ as every vendor want to produce and comeup with their own algorithms and strategies of optimizing in storing the data.

Across all the operating systems does the data is portable or not?
We cannot copy or transfer the data from one operating system to another since their file systems are different. Now a days there are system software built on which this portablity can be achieved to some extent.

Windows File System.
FAT32 - desktop
NTFS - server grade systems (security)

Mac File System
HFS - Hierarchial File System.

Linux File Systems
1. ext = First generation File System available, no more used right now.
2. ext2 = 2nd generation File System, supports upto 1 tb of data to organize
3. ext3 = 3rd generation File System, it works for large files but crashes oftenly and cannot be recoverable.
4. ext4 = current generation File System that comes as part linux installation.It solved the problem of recovery using Journal.

--- distros no more available in the market.
5. jfs =
6.xfs =
7.btrfs = oracle and not efficient in handling small files

How does ext4 File System organizes the data on the storage.
ext4 is an hierarchial techinic for oraganizing and storing the data.

hierarchy = expressing something interms of parent and child relationship.
In a Flat filesystem we cannot easily identify and access all of the files we stored on the computer, as classification is not there.
In hierarchial system as data/files are grouped in parents/child directories it would be quick to find the files by navigating to that directories. In addition we can enforce security or change of file attributes to the parent and child through inheritance.

https://www.leawy.com/drive/login.htm
www.leawy.com = web site

What are the File Systems types available/supported by the Linux operation system?
1. ext
2. ext2
3. ext3
4. ext4
5. btrfs
6. jfs
7. xfs

Linux File System is Hierarchial File System
- Parent and Child relationship is called "Hierarchial"
advantages of it:-
1. Traversing the Files/Folders of the storage device is easy.
2. Easy to organize and group the data/files
3. We can perform operations on a group/parent level like delete all the files of a parent or apply security permissions for these files under this parent.

Linux operation system as part of its installation creates some default files/directories on the storage device. It is important that we understand the directory structure that would gets created out of box.

/ [root]
|-/bin [binary / executable] = the linux system commands/bash commands are placed in this directory
|-/boot = [boot loader files in starting the linux os]
|-/dev = [device all the system devices are mounted on /dev] [/dev/sdv0] [/dev/sdv1]
|-/mnt = [all the remote system directories are mounted into this directory]
|-/media = [headphones / music player]
|-/usr = programs/softwares under this directory are shared to all users of linux
|-/proc = process under execution and their information is kept as files in this directory.
|-/lib = shared libraries across the softwares will be placed here
|-/tmp = temporary system files
|-/var = linux system logs and very large files
|-/etc = linux system configurations files
|-/opt = software packages will be kept here
|-sbin = super user bash commands
|-/home = per user one home directory gets created
|-sriman

File System Types
- ext
- ext2
- ext3
- ext4
- btrfs
- jfs
-xfs

Hierarchial FileSystem
Expressed interm of parent and child relationship is called Hierarchial. Hierarchial data structures resembles trees and has only one parent at the top.
1. Traverse the files & folders is easy
2. enforcing the security on group of files and folders
3. managing the file operations on the group.

/
|-/bin = bash commands
|-/boot
|-/dev
|-/etc
|-/opt
|-/media
|-/mnt
|-/tmp
|-/var
|-/home
|-rick
|-/proc
|-/usr = shared programs
|-/sbin
|-/lib = shared libraries that are used by the programs

Linux System Architecture
Refers what components of the operation system are available to interact with the underlying hardware of the computer.
Layers of the operating system enabling us in communicating with the hardware of the computer.

Shell = Shell is a software program / Utility Program that provides bunch of commands using which we can interact with kernal of the Linux Operating System. This helps us in ease the interations with Linux.

Basically there are 2 types shells are there
Bourne/Bash Shell
C Shell

Bourne Shell
- Bourne (SH)
- Bourne Again Shell(BASH) (BASH)
- Korn Shell

- POSIX Shell

C Shell
- C Shell
- TCSH (Tenex Shell)

Linux Terminal Window [CUI]

Shell = Software Program/Utility helps us in interacting with kernal of the linux operating system.

At high level we have 2 types of Shells.

Bourne Shell
1. Bourne Shell (SH)
2. Bourne Again Shell (BASH) = its a better version of Bourne Shell (SH) (Shell Scripting)
3. Korn Shell
4. POSIX [Portable operating system interface]
C Shell
1. C Shell - C programming
2. TENEXT C (TCSH) = extension C Shell + History, up arrow, down arrow, inline editing

How do we use these Shell in Linux.
Linux has provided a Terminal Window using which we can run the shell software. Its a Character User Interface (CUI).

debain distribution = ubuntu
ubuntu-20.04-desktop-amd64.iso
download url - https://ubuntu.com/download/desktop

Virtualization Softwares = we can create and run mutiple isolated environments on the same physical system
-Oracle Virtual box
-VM Ware
-Hyper V

Virtual box download - https://download.virtualbox.org/virtualbox/6.1.10/VirtualBox-6.1.10-138449-Win.exe
Extension pack - https://download.virtualbox.org/virtualbox/6.1.10/Oracle_VM_VirtualBox_Extension_Pack-6.1.10.vbox-extpack
- extension pack few features will be enabled.
1. Syn folders
2. Network Configurations / Adapters
3. Mount USB Devices

- Virtual box is not 2nd computer, using this we can create another machine. open Oracle VM Virtual box from program menu, will virtual box manager.

www.leawy.com/drive [register using your email address and login to download]

To register for Java Realtime Tools [9:30 am - 10:30 am] send an email. [!Important = you need java basics]
subject = ODWA1-REALTIMETOOLS
Drop an email to durgasoftonlinetraining@gmail.com with cc="tech.sriman@gmail.com"

Shell = Software utility or a program that helps us in interacting with the kernal of the operating system.
There are 2 types of shells.
Bourne Shell
1. Bourne Shell (SH)
2. Bourne Again Shell (BASH)
3. Korn Shell
4. POSIX Shell

C Shell
1. C Shell
2. TCSH

In Virtual box after starting the machine
Goto view menu -> select aspect ratio dependents on your machine and choose 250% autoscaled for adjusting your screen.

[Ctrl = right side on the keyboard]
Ctrl + C = Scaled Mode (Menu Hide/Show)
Ctrl + F = Fullscreen mode

Linux Terminal
1. ctrl + c = kills the current running process.
2. ctrl + l = clear the screen
3. ctrl + a = moves the cursor to the begining of the line
4. ctrl + e = moves the cursor to the end of the line
5. ctrl + u = deletes the characters from the current cursor position to the begining of the line
6. ctrl + k = deletes the characters from the current cursor position to the end of the line
8. ctrl + r = search in the history of commands
9. up arrow =brings previously typed commands
10. down arrow = will navigates back to the current commands.
11. ctrl + left arrow = moves one word to the left of current cursor position
12. ctrl + right arrow = moves one word to the right of current cursor position.
13. ctrl + shift + c = copy text
14. ctrl + shift +v = paste text

Bash command structure
---
command [options] [inputs]

options = customizing the way command works or tweaking the output. Is a way through which we can pass additional instruction asking the command to perform the operation.
command [option]
-character
for every command options are optional to be passed.
for a command we can pass multiple options as well.
command -option1 -option2 -option3
command -option1option2option3

inputs = acts as input for the command on which it has to perform the operation.
inputs are optional and may not exists for few commands in linux.

File and Folder

File = It is used for storing sequence of set of characters or data on a disk or storage device. We always organize and store the data as part of a file so that we can access it collectively/ easily.

Open Terminal Window we will be under a default prompt directory called User home.
/home/sriman/

cd directory/path = change directory
cd .. = moves you to the parent directory.
pwd = print working directory.

touch is command used for creating an emtpy file.
syntax:-
touch filename = creates an empty file in the present working directory.

echo is used for displaying the text on the terminal
echo "Good Morning" = it prints "Good Morning" to terminal.

cat filename = displays the contents of the file.

create an empty file and write content into it.
touch filename
echo "text" > filename

excersise
1. explore the default directory of a terminal
2. change to a parent directory
3. create an empty file and see the size of the file
4. create an empty file and write data into the file
5. create an file with data directly.

Storage Management commands
[File and Directory commands]

1. pwd = print working directory = to find the directory location in which we are in.
2. touch filename = creates an empty file under the current directory location.
3. cat filename = shows the contents of the file
4. echo "text" = will writes the text on to the console
5. echo "text" > filename = With this we can write some text content into an existing file. Here ">" is a redirection operator to write the output to different location. Without creating the file using touch command we can directly say echo "text" > filename, if the file doesnt exists it creates a new file with that text content.
6. cd directoryName= change directory
7. ls = list files and folders under the current directory.

---

ls -R directoryName = displays recursively all the directores and sub-directory contents

What is a directory/folder?
Directory is a special type of File. In a File we store data, instead in a Directory File rather than storing data we hold information/pointers to the actual files contained in it.

Why we need directories/folders?
Its a way through which we can organize or group related files and can locate them easily on the storage device out of lot of files.
Always a directory contains files/folders inside it.

How to create a directory?
mkdir directoryName = This creates an directory with the specified name under current directory.
ls directoryName = to see the contents of a directory.

cd directoryName = to change to the newly created directory.

"." = refers to current location/directory
".." = refers to parent of the current directory

cd .. = takes you to the parent directory of the current directory.
cd = without any input takes you to the home directory of the user. [For e.g.. /home/username]
cd ~ [tilda = below the escape key on keyboard] = takes you to the home directory. Always "~" refers to home directory location of the user.
cd /dir1/dir2 = we can navigate through multiple directories using "/"
cd - = will takes you back to the directory where you came from.

/dir1/dir2
|-tales1/tales2
|-bulletin1/bulletin2

cd /dir1/dir2/tales1/tales2

2 ways we can navigate:-
cd /dir1/dir2/bulletin1
cd ../../bulletin1

2 ways

cd ../tales1/tales2
cd /dir1/dir2/tales1/tales2
cd - = takes you back to the previous directory where you came from.

You can view structure of a directory by using tree command. Tree is a utility that will not available by default as part ubuntu distro. We need to install by using below commands.

sudo apt-get update
sudo apt-get install tree

you can use tree directoryName
stories/
├── action
├── kids
│   ├── infant
│   │   └── rhymes.txt
│   └── toddler
└── scifi

mkdir directory1
mkdir -p /directory1/directory2/directory3 = create then entire path of directories if they dont exist

cat filename = cat command takes the data from somewhere and print it
cat filename = it takes input from file and displays on to the console
cat = takes the input from input device/keyboard/console we have to type ctrl + d to terminate the cat.

cat >> filename = this takes the data from input/keyboard and redirects/writes the output to filename.
echo "hi" > filename
echo "end" >> filename
">" =redirect the output on to the file
">>"=append the output onto the file

head =is used for displaying the contents of the file from top to by default 10 lines
if we want to see n lines from top use
head -n filename

tail filename = displays the last/bottom 10 lines of the file by default
tail -n filename=display n lines from the bottom
note:- tail is especially used to see log files of an application.
tail -10f filename = file stands for float on the file. display the flow of output from the file.

In linux files are not identified by their type using extension. Linux will not categorize a file based on extension. rather it categories based on the content stored inside it.

file filename = shows the type of the file
stat filename = gives statics about the file like name,size, blocks, created and modified.

cat > file1
cat takes the input from a file and writes to the console, but in the above command we have not given the filename to read from, so it waits for the user input from the keyboard, once we enter data as we use redirection operator instead of writing data on to the console it writes to the file.

head -5 filename =shows top 5 lines of a file [default = 10]
tail -5 filename = shows bottom 5 lines [default = 10]
tail -f filename = keeps hold of the and shows the flow of data.

How do we identify the file types in linux?
Linux doesnt uses extension of a file to determine the type of a file [audio, video, text] rather it looks for the content of file to classify it. So end user by looking at the filename cannot identify the file type so linux has provided a shell command handy to identify it "file".
file filename = tell the type of the file.

stat filename = gives statistics information about the file.
Blocks, inode, created date, accessed date, last modified, owner, permissions.

mkdir -p /dir1/dir2 = creates the directories including the parents.

---
ls = shows files/directories with color
ls --color=none = displays files/directories with no colors.

expression we can use as part of the commands
* = multiple character match
{} = multi inputs
? = single character match

ls a* = shows the contents of the directories which starts with a or gives names of the files start with a.
ls {*.txt,*.png} = shows the files with any of those two extensions.
ls ??? = shows contents of the directories of 3 characters in length

rm filename = for removing a file
rm -f filename = force remove file
rmdir directory =removes the directory if it is empty

rm directory = we cannot remove directory
rm -r directory = here -r stands for recursively remove so if you directory as an input it removes files/directories inside it including that directory.

how to create hidden files in linux.
In linux if the filename starts with "." it is considered as hidden file
touch .sec
ls -a = then it shows hidden files [-a stands for all]

1. regular expression we can use
2. create directories including the parent
3. Interactively how to create a file
4. removing directories and files.
5. Hidden Files

mkdir -p dir1/dir2

3 regular expression are there
* = matches any no of any characters
? = matches any single character
{} = range of inputs

cat > filename
rm filename
rm -f filename = f (force)

rmdir directory [given the directory is empty]
rm -r directory = [recursively deletes contents of directory along with directory]
---
cp file1 path/filename
~/accounting/finance.txt
~/accounting$ cp finance.txt finance_final.txt [source/destination directories are same so we told the filename into which it should copied. So, cp creates a new file with destinationName and copies the contents]

~/accounting$ cp finance.txt ~/accoutings/backup [destination is different so the current file will be copied]
~/accounting$ cp finance.txt ~/accoutings/backup/financebkup.txt

cp -R dir1 destinationPath = to copy contents of a directory including that directory
cp file1 file2 file3 destination
---
mv [move] =is used for moving files/folders from one location to another. It can also be used for rename when source/destination are same.

~/accounting$ finance
~/balance_sheet

~$ mv accouting/finance balance_sheet
~$ mv accouting/finance accounting/finance-2020 //rename
~$ mv file1 file2 file3 destination/
~$ mv director1 destination/
---
tr = is used for deleting,replacing or changing to upper or lower letters of a given character
delete or replace works on only one character
transformation applies to whole string.

echo "Good Morning" | tr "[:lower:]" "[:upper:]" (or) echo "Good Morning" | tr [a-z] [A-Z]
echo "Good Morning" | tr M E

```
echo "Good Morning" | tr -d M
echo "Good Morning" | tr [:space:] &
echo "Time is 7'oclock" | tr -d [:digit:]
```

cp file/folders copy
cp -R dir1 dir2
---
cp file1 dir2
cp file1 file2
cp file1 dir1/file2
---
mv files/folders to a different location or can be used as a rename

mv file1 dir1
mv file1 file2 (rename)
mv dir1 dir2
---
tr = replace or delete a character of string sequence or transformation of sequence of characters.

tr h l
tr -d l
tr [:upper:] [:lower:]
tr [:digit:] n =every digit should be replaced with 'n' char
tr [:space:] x = every space character is replaced with 'x' char
---

echo "This is Devops Course" > file1

wc = word count can be used
-l = lines
-c=characters
-w-words

cat file1 | wc -l (or) wc -l filename
cat file1 | wc -c (or) wc -c filename
cat file1 | wc -w (or) wc -w filename
cat file1 | wc -lcw

---

find = find is command used for finding the files on your linux filesystem based on name of the file.

syntax:-
find directory -name "[pattern]" [by default -print is enabled]
find directory -name "pattern" -print0 [displays the contents in one single line]
find directory -name "pattern" -empty
find directory -name "pattern" -exec command {}\;
find directory -name "pattern" -o -name "pattern" = -o stands for "or"

find directory -name "pattern" [-print enabled]
find directory -name "pattern" -print0 [everything in one line]
find directory -name "pattern" -empty [shows only empty files]
find . -name "*.log" -o -name "*.out" -o -name "*.tmp"
find directory -name "pattern1" -o -name "pattern2"

find directory -name "pattern" -exec wc -l {} \;
\; = termination of the command

find . -name "*.log"
find . -name "*.tmp"
find . -name "*.out"

wc = stands for word count
If i want to count the characters, words and lines in a given file i can use wc
Following are the options we have
-l = lines
-w = words
-c = characters
and we can use any of the options in combination as well.
Syntax:-
wc -lcw fileName

---

find = we want to search for a file on linux filesystem use find command.
How can we use it?

syntax:-
find directory -name "pattern" -print
There are plenty of options available in find.

1. find directory -name "pattern" -print = Here we want find to look for a file under specified directory of that pattern. Print prints the complete path of the file from root location and by default print is enabled even dont pass.

2. find directory -name "pattern" -print0 = -print0 will not output the full path of the file it only lists the filenames found in a single line

3. find directory -name "pattern" -empty = prints only the files which are empty

4. find directory -name "pattern" -o -name "pattern" = -o stands for or means either matching first pattern or matching second pattern

5. find directory -name "pattern" -exec command {} \; = -exec indicates execute the command supplied here by passing all the files matching with that criteria.

find logs tmp -name "*.log" -exec rm {} \; -o -name "*.out" -exec rm {} \; -o -name "*.tmp" -exec rm {} \;

---

grep = grep is command for searching a text in a file or find the matching content files.

syntax and options of the grep:-

grep text fileName = it searches for the given text in the specified file and prints matching lines.
for e.g..
grep "joy" stories/kids/infant/story1
Remainder recommend engrossed who eat she defective applauded departure joy.

grep -R "text" directory = It searches recursively into all the files and subdirectories with that matching content and displays along with matching textline the path of the file also.

grep = text search in a file or group files can be done using grep.

grep "text" filename = it displays all the matching lines of text within that file.
grep -R "text" directoryName = it displays all the matching lines of text along with file path
grep -n "text" filename = along with matching lines it displays line number
grep -iR "text" directory = ignore case comparision
grep -oR "text" directory = output the matches words only
grep -w "text" filename=matches the whole word
grep -c "text" filename = count of occurences
grep -digit "text" filename = display the no of lines around the matching lines.

---

cmp = for comparing the byte by byte of text information of a file and outputs the first difference of the file.
cat >menu1.txt
menu1.txt
idly
vada
poori

cat >menu2.txt
menu2.txt
idly
bonda
poha
vada pav

cmp menu1.txt menu2.txt
output:-
menu1.txt menu2.txt differ: byte 6, line 2

---

diff = it is used for comparting two text files and displays all the mis-matches in the files. In addition it displays special symbols helping us in understanding what we should do to make those two files identical.

a = add
c = change the text of the file
d = delete line

In addition to this it displays the mis-match lines with notation.
> = indicates the right side file
< = indicates suggestion should be applied to left side file
2,5c3 < =in the left file from 2 - 5 lines should be replaced with 3 line of the right file contents.

diff stations1.txt stations2.txt
2c2
< Mumbai

> Vijayawada

2nd line of left file should be changed with 2nd line of the right side file.

diff artists1.txt artists2.txt
0a1
> Mahesh babu
0th line of left file should be added with 1st of right side file.

0th line leftside add what 1 st line of right side file "Mahesh babu"

http://www.leawy.com/drive

Mostly Linux operating system is used in server-based install [headless]
1. Less computing resources
2. Highly secured

Text Editor (CUI)
1. VIM [VI]
2. Nano

VI = Default Text Editor available in most of the Linux Distros

There are 3 modes of VI Editor
1. Command Mode
In command mode we cannot edit/input the text into the file/editor. Here any keypress we do will be taken as a command by editor and will perform a relavant operation.
To enter into command mode we need use [ESC] key. by default

2. Input Mode
Input mode any key press will be taken as an input and will writes on to the file
To enter into input mode you should press [i] key on the keyboard

3. Exit Mode
In this we are going to quit from the editor.
First we should be back to command mode by [ESC] and should type :q [quit]

vi filename [existing file open its contents][new file shows as blank]

with the default installation of ubuntu vi older version will be present.
Now we can upgrade latest VIM editor by doing the below.
sudo apt-get update
sudo apt-get install vim

What are the commands we use in VI in command mode?
A = append the contents to the end of the current line
a = appends the contents from the current cursor position

I=insert contents from the begining of the line.
i=insert contents from the current cursor position

R=Replace characters from current cusor position
r=replace single character
S=replace the current line

O =open a new line above the current line
o = open a new line below the current line

yy = yank (copy the line)
2yy = copy 2 lines
p = paste

x=delete a character
dw=delete a word

dd=delete line
2dd=delete 2 lines

vi/vim editor

command mode
everything that we type is a command to the editor will perform some operation.
by default the editor opens in command mode. anytime to return back to the command mode we need to press [ESC]

input mode
any key we enter will be written on the underlying file. by presing "i" we can enter into the (insert) mode.

exit mode
to quit from editor we should go back to command mode and use :q

i = insert text from the current cusor position
I = insert the text from begining of the line
a = append the text from current cusor position
A = append the text at the end of the line
r = replace a character
R = replace from the current cursor position
S = replace the entire line
YY = yank (copy)
2YY = copy 2 lines
p = paste
x = to delete a character
dw = delete word
dd = delete line
2 dd = delete 2 lines
o = open a new line below the cursor position
O = open a new line above the cusor position
b = move cursor to the begining of the previous word
w = move cursor to the begining of the next word
e = move cursor to the end of the current word
u = undo
[ESC] :1 = to goto the begining of the first line
Shift + G = End of the file
:q = quit
:w= write
:q! = quit without saving
:wq = write and quit
[ESC] / = searching a word then press enter
if we press "N" next word "B" previous word matching [ESC]
ctrl+f =page forward
ctrl+b=page backward

---
nano editor = popular
nano filename
ctrl+o = save file
ctrl+x = quit

---
file packaging and compression

packaging = we can package multiple files of a directory into one single file by using packaging. Packaging doesnt compress the contents file/folder.The only purpose is make into one single file so that it can be easily distributed.
linux has provided a utility called "tar" stands for tarball.

tar -cvf books.tar books
-c = create
-v = verbose
-f = output filename
always the tar files has the extension ".tar"

how to untar the files
tar -xvf books.tar
-x = extract
-v = verbose
-f = input filename

tar -xvf books.tar -C directory = untar the file into the specified directory.

compression

packaging = group of files in a folder can be packaged into a single file using packaging. So that it can distributed to others over then network.

Linux has provided a software utility "tar" = tarball

packaging= tar -cvf target.tar sourcedirectory
un-pack = tar -xvf target.tar
un-pack to a different directory = tar -xvf target.tar -C directory

compression = The group of files of a folder cannot be compressed directly unless those are packaged into a single tar file. The compression technic will reduce the size of the original files so that it can be shared/distribtued quickly over the network.

Linux has provided a software utility "gzip and gunzip"
gzip = compression
gunzip = uncompression

~/$target.tar
~/$gzip target.tar = it creates target.tar.gz file with zipped contents in the same directory,now here you wont find .tar file.
".gz" = indicates its an linux zip extension

gunzip target.tar.gz = this will unzip the .gz into .tar back

to know the size of a file/directory contents we can use below command
du -h filename/directoryName

User Management
Linux is a Multi-User operating system, multiple users can login and access the system resources at the same time.

In Linux we have two types of users.
1. ROOT user = There can be only 1 ROOT user of a Linux Install. He has total control of the entire system resources.
1.1 CREATE OTHER USERS
1.2 MODIFY OTHER USERS
1.3 CAN MANAGE ANY FILES ON THE DEVICE
1.4 CAN MANAGE PROCESS

2. Normal user = Other than the one ROOT any user we add into linux becomes Normal User. The Normal user can only see his/her files under the user home directory but cannot have access to any system resources.

packaging and compression

if i want to package the contents of a directory or group of files together into one single file then use packaging.
tar = tarball
-x = extracting
-v = verbose
-f = archive name
-t = table of contents
-c = create
--delete = to delete a file inside the tar
--add-file=to add a file into existing tar

for e.g.. tar --delete -f images.tar index.jpeg

gzip filename.tar
gunzip filename.tar.gz

User Management in Linux

There are two types of users are there in linux
1. Root User = There is only one root user for a Linux System and would gets created by default through installation.
The root user has total control of the Linux System like
1. Managing all the system resources
2. Managing the processes
3. User Management
4. File Management
5. Network Configurations
6. System Configurations
7. Software Installations

Never loose the root user of the linux if so you lost the access to the system.

2. Normal User = Any other user added by the ROOT will becomes Normal User. He has access to his home directory only can manage files and process of his own.

Groups:- User in linux can't be managed individually as it takes lot of time in tracking which user has access to what and issuing permissions individually also takes lot of time.
That is where Linux comes with groups. We can create groups and can associate users to groups so that collectively we can manage those group of users by granting permissions/ revoking to the group.

What are basic attributes of the user in linux?
Every user we add in linux is identified by uid of the user.
- username
- uid
- password

For every group also we have gid (group id) through which a group is identified

How a user is added to the linux system?

The user is added to the linux system by creating an User Account
1. While adding the user account for a user it generates the uid of the user automatically by incrementing the last uid.
2. Every user must and should associate to a group in linux, so when we add a user linux will automatically creates a group with the username and associate the user to his group.
3. For every user account being added a default userhome directory will be created and associated for that user.

and stores all this information about the user in a file called /etc/passwd

useradd username
useradd -u uid -g gid -d /home/directory -s shell

sudo su - = will switch the user into root prompt.
By default the terminal will not be opened with root privilige to protected accidental damage to the system resources.

/sbin = commands

1. group will be created withthat user with a gid
2. user will be created by generating the uid and associate the user to the group.
3. create home directory and associate with user

before perform any user management operations we should login as a root user in terminal/ bash.
so to switch to root user we should run below command
sudo su -

For all the commands in /sbin directory can be only accessed by super user of the linux. By default linux terminal logins in the user as normal. In order to execute any of the /sbin commands we need to switch as root.
Instead sudo is a command helps us in executing /sbin commands with super user priviliges without switch the prompt.
sudo = super user do = do/execute command with super user privilige.

sudo can be used by only root as of now.

How to add a user in ubuntu
adduser username

What are the Text Processing Commands available in Linux?
There are 4 commands are available.
1. cut
2. paste
3. awk
4. sed

cut -d "char" -f fieldNo fileName

paste = is used for merging 2 files line by line.
paste -d "delimeter" file1 file2

awk = extracting the portion of a text file based on delimeter/pattern or a combination of them.

awk [options] 'pattern' fileName

awk '{print}' fileName = prints the contents of the file
awk -F "delimeter" '{print $1}' fileName = prints the first field

awk 'regularExpression {print}' fileName
awk '/developers/ {print}' fileName

awk -F "delimeter" '/developers/ {print $1}' fileName

---
sed = streaming editor = process the text line by line.
sed is used in various ways, like searching the text, deleting, and substitution

print
delete
substitute

stories.txt

India Today News breaks the most important stories in and from India and abroad in six sections - India News, Business News, Cinema News, Sports News, World News and Lifestyle News. India News keeps tab of every development in all parts of India. Business News has the latest business updates from India and abroad. Cinema News tracks the latest from Bollywood, Hollywood and the South film industries and TV channels. Sports News has all the sports from India and abroad with a special focus on cricket. Lifestyle News presents developments that impact one's lifestyle. World News makes sense of news across the world and its impact on India

sed can take STDIN as an input or file as an input on which it apply text processing. after processing the text it output the text by default on STDOUT

How to use sed.
sed ' ' news.txt = by default reads the content of file and output.
cat news.txt | sed ' ' = both are same.

syntax:-
sed [options] commands [File Input]

#1 Printing Lines
---
sed 'p' news.txt = p stands for print command for sed and it prints twice line by line because default option is also print.

sed -n 'p' news.txt = supress the default printing option.

#1.1 Using address ranges
sed -n '1p' players.txt
sed -n '1,3p' players.txt = print lines from 1 - 3
sed -n '2,+2p' players.txt = print from line 2 next 2 lines
sed -n '2~3p' players.txt = print from 2nd line every other line after 3 line.

#2 deleting text
d stands for delete command.
sed will not operate on the file contents rather it reads and apply commands in the memory and output the STDOUT.
sed '1d' players.txt
sed '1, 2d' players.txt

if you want to change the contents of file and write it you can use redirection operator but should not apply on same file
sed '1d' players.txt > players1.txt

but i want to modify on the same file.

sed -i '1d' players.txt = inline modification. read the line apply processing and write on the same line.

while using -i original file will be modified so we should very careful while using -i instead sed has provided backup feature of original before applying changes using below command.
sed -i.bak '1d' players.txt

#3 substituting text

's' = stands for substitute command

sed 's/dhoni/suresh raina/' players.txt = line by line searches dhoni and subtitutes with suresh raina
by default substitute works line by line and replaces only first occurence of the word in each line.
if we want to replace all the occurences of the word in each line we need to use 'g'.

sed 's/Dohni/Suresh Raina/g' players.txt

if we want to replace only 2nd occurence use '2'
sed 's/Dohni/Suresh Raina/2' players.txt

sed -n 's/Dohni/virat/p' players.txt = print only the substituted lines only.
sed by default searches for the matching text based on care,if we want sed to ignore case while comparing we need to use 'i'
sed 's/dohni/virat/ig' players.txt = i = here for ignore case, g = here for global replace

cat ui-config.properties
color=red

background-color=white
font-size=12
font-family=consolas

I want to replace color property with blue.
sed 's/^color=.*/color=blue' ui-config.properties = this replaces all the lines which are starting with color=
sed 's/.*color=.*/color=blue' ui-config.properties = this replaces all the lines which has a word color=

sed 's/.*color=.*/&,blue/' ui-config.properties = we are fetching matched content with & and appending to it the output of this command is as below.

cat ui-config.properties
color=red, blue
background-color=white,blue
font-size=12
font-family=consolas

How to write a shell script and execute it?
You can write any file with shell commands and usually saved with an extension (.sh).
Every shell script should start with shebang #!/bin/bash pointing the directory of the shell.
Shebang tell here to the Linux this shell script should be executed with which Shell Interpreter.

Shebang is optional and if not specified Linux uses the default shell of the user which we specified at the time of adding the user.

What is the difference between a file and a program?
In general a File holds the data or a text of information that is used for reading or modifying.
A program is also a File stored on File System. But it is a special file, A program file rather than having data inside it it has executable instructions which can be executed by the underlying operation system / interpreter program.

lsdirs.sh
#!/bin/bash
ls -l | grep "^d"

lsdirs.sh -rw-rw-r-- = a file by default will be saved without execute permission.
chmod u+x lsdirs.sh
./lsdirs.sh = will execute the shell file

without execute permission we can run if we have read permission
sh lsdirs.sh = read the shell instructions and pass to sh interpreter so that it executes.

PATH=
/bin:/sbin:/usr/bin:/usr/sbin

users
export PATH=everyone
bashrc

root user can makethe command availabe to all the users of linux
symlink = /usr/bin

What is shell scripting?
File containing shell commands and shell instruction set that is saved with usually (.sh) extension and is executed.

lsdirs.sh
[shebang]
#!/bin/bash
ls -l | grep "^d.*"

chmod u+x lsdirs.sh
./lsdirs.sh

no execution permission but with read permission we can execute
sh lsdirs.sh

To run the above shell script from any directory location of the system we have 2 approaches.
#1 approach
1. export PATH=$PAH:DIRECTORY_LOCATION_OF_SHELLFILE
repeatedly we need to set it on the terminal instead other option.

2. ~/.bashrc
export PATH=$PAH:DIRECTORY_LOCATION_OF_SHELLFILE

#2 approach
If we want to distribute the shell file to all the users of linux machine then first give execute permission on the shell file to all the users.
chmod uog+x lsdirs.sh
Then create a symlink pointing to the file with lsdirs under /usr/bin as /usr/bin is a default directory added in PATH of every user the symlink will become accessible globally.
ln -s /common/sh/lsdirs.sh /usr/bin/lsdirs

What are variables and how to use them in shell script?
Variables are the names assigned to the literal values in program
(or)
Variables are the place holder of values.
Variables are named memory locations in which we can store values.

Using Variables we can save maintainance of the application, in the event of change in value we can simply change the value we assigned to the variable rather than changing all the lines of code where we used literals.

How to declare variables in SHELL Script. In SHELL its a common practise of declaring variable name with CAPITAL letters.
VARIABLE_NM=VARIABLE_VALUE

How to use value of a variable?
We need to use $VARIABLE_NM to access its value of its memory location.

#!/bin/bash
A=10
B=20

echo $A
echo $B

Reading data interactively in SHELL SCRIPT
How can we take the inputs from USER at the time of running SHELL SCRIPT?
read A

add.sh
#!/bin/bash
echo "Enter A:"
read A
echo "Enter B:"
read B
echo "A : $A and B: $B"

---

ls
wc filename
grep "text" filename

ls = it will list all the files and folders of the current directory
ls /stories = non-interactive inputs or command-line inputs

Most of the time we design shell script programs to accept non-interactive inputs through command line.
./add.sh
./add.sh 10 20
./add.sh 20 40

ls /stories

How to pass the data as an input to the shell program non-interactively?
There are two ways of passing the data as an input to the shell program.
1. interactive = Here the program during execution will wait for the user to enter the data as an input to continue execution with user input data.

2 non-interactive = Here while running the program itself the user is going to pass the input values with which he wanted the program to perform operation, this can be done by passing additional arguments along with program while running as below.
./program.sh 10 20 30
In the above we are running a shell program name "program.sh" and passing three arguments.10 20 30

non-interactive inputs are more preferred way of passing data to the shell programs as we can achieve seemless automation with them without user interaction.

./program.sh 10 20

program.sh
#!/bin/bash
A =$1
B =$2

How to work with performing operations in shell program?
In general we can perform mathematical operations on two numberic inputs which can be either integer/floating point. But we cannot perform mathematical operations on "Character data / String".

by default in shell program every value we refer/use are treated as "Character data/String" and hence we cannot perform mathematical operations.

A=10
B=20

How ways we can send the data as an input to the shell program?
There are 2 ways we can send the data as an input to the shell program
1. interactive
2. non-interactive

non-interactive inputs can be passed as arugments to the program while running the program. How can the program can read those inputs passed as arguments inside it.

Bash takes those arguments and pass them as inputs to the shell script as special variables so that our program can refer those variables in reading the data.

$0 ... $NArgs
$0 - shell script file name or command name we are running
$1..$2... - arguments we passed
$# = no of arguments
$* = all the arguments

There are two statements available for printing the output to the console.
1. echo = echo always outputs the new line at the end of the output.
2. printf = it allows format specifiers on how to format the data while rendering on to the console output.
Formatters are: %s, %d, %f = string, integer and float respectively

---

By default all the data we use as part of Shell Script will be refered/treated as "Characters/String Format", so we cannot apply any arthematic operations on the data.

There are two ways we can perform the arthematic operations.
1. expr = is an shell utility/program that takes 2 strings as an input with operator and converts them into "Integer" and perform operation. expr is used for evaluate the expression asking him to compute.

```
#!/bin/bash
A=10
B=20
SUM=`expr $A + $B`
echo $SUM
```

The expr is an old fashion of writing the arthematic operations and there is a an different way of doing this.

```
#2 approach
#!/bin/bash
A=10
B=20
SUM=$((A + B)) #arthematic expression evaluated by has.
echo $SUM
```

#3 short-end format of wrting arthematic expression
instead $(()) = we can use $[]

Command Substitution.

In general when we run a shell command those exibhit the output on to the console. If we want to capture the output of a shell command in a script variable then we need to use command substitution.
syntax:-
VARIABLE=$(shellcommand)

$(()) = expression / $[]
$() = command substitution

expr can also be applied on String Types

We can find length of a string as below
MESSAGE="Good Morning"
LENGTH=`expr length  "$MESSAGE"`

We can extract a sub string of a String.
AWBNO="AWB0029203981"
AWB10Numbers

REFNO=`expr substr $AWBNO 4 13`
echo $REFNO

here 1st number is position to start in the string
2nd number is length of characters to extract

MESSAGE1="Good Morning"
MESSAGE2="Good Evening"
MATCHES=`expr "$MESSAGE1"  :  "$MESSAGE2"`

$(()) = is used for performing arthematic operations
$[] = alternate to $(())

---

command substituion
The output of a linux command if we want to capture in an variable to perform further operations then we use command substitution.
VARIABLE=$(command)

What are operators?
Operators are pre-defined Symbols provided by Shell Utility like BASH with predefined meaning attached to them.
Upon using those operators in our shell script BASH will perform relevant operation.

How many types of operators are provided by bash.
There are 5 types of operators are available.
1. Arthematic operators
2. Relational operators
3. Logical operators
4. String operators
5. File operators

---

1. Arthematic operators = Used for performing mathematical calculations.
+ = add
- = substraction
* = multiplication
/ = division
% = modulus = remainder

2. Relational operators = To compare any two given inputs and derive the relation ship between

them
The given two inputs are same or one is greater than other or less than
These operators are only applicable on integers only.

-gt = greater than
-lt = less than
-ge = greater than or equal to
-le = less than or equal
-eq = equal to
-ne = not equal to

What are operators?
Operators are predefined symbols attached with special meaning by the bash interpreter. So that we can use them as part of shell script to let the bash perform operations.

How many types of operators are there?
There are 5 types of operators are available.
1. Arthematic operators
2. Relational operators
3. logical operators
4. String operators
5. File operators

1. Arthematic operators = these are used for performing mathematical operations
+ = addition
- = substraction
* = multiplication
/ = division
% = modulus = remainer out of division

2. Relational operators = for comparing 2 different inputs we use relational operators and these operators can be applied only on integers. The output of these operators are either true/false
-gt = greater than
-lt = less than
-ge = greater then equal
-le = less than equal
-ne = not equal
-eq = equal to

3. logical operators = used for combining multiple relational operators and evaluate together.
-a = and
-o = or
! = not

4. String operators = used for comparing strings
> = greater than
< = less than
>= / <= = greater/less than equal to
-z = is zero length
-n = non-zero

5. File operators = these works on file and directories
-e = is file/directory exists
-s = size of the file is greater than 0
-c = character special file
-f = regular file
-d =directory
-r = read permission
-w = writer permission
-x = execute permission
-o = owner file
-of =older than
-nf =new than f1 and f2

Control Statements

operators
special symbols defined with meaning up on using them in shell script will perform an operation.
- arthematic operators
- relational operators
- logical operators
- string operators
- file operators

Control Statements
2 Types of statements are there
1. Conditional Control Statements
2. Loop Control Statements

1. Conditional Control Statements - based on the outcome of evaluating an expression we wanted bash to execute the next set of lines. Then we need to use Conditional Control Statements.
1.1 IF STATEMENT
1.2 CASE STATEMENT

IF STATEMENT = IF the condition has been passed then only execute the block of code sourrounded in the IF condition.
syntax:-

```
IF [ CONDITION ]
THEN
STATEMENT1
STATEMENT2
STATEMENT3
FI
```

WAP to find biggest of 3 numbers by taking non-interactive input

```
#!/bin/bash
N_ARGS=$#

IF [ $N_ARGS -ne 3 ]
THEN
echo "ERROR! Required 3 Arguments as Input"
exit
FI

A=$1
B=$2
C=$3

IF [ $A -gt $B -a $A -gt $C ]
THEN
printf "%d is bigger" $A
FI
IF [ $B -gt $A -a $B -gt $C ]
THEN
echo "$B is bigger"
FI
```

```
IF [ $C -gt $A -a $C -gt $B ]
THEN
echo "$C is bigger"
FI

IF [ $A -eq $ B -a $ A -eq $ C ]
THEN
echo "all three numbers are same"
FI
```

LOGICAL OPERATORS = Used for joining expressions and evaluate together

AND	OR	NOT
T T = T	T T = T	T = F
T F = F	T F = T	F = T
F T = F	F T = T	
F F = F	F F = F	

wap to print wether the given number is even or odd
any number that is divisible by 2 is an even number

10/2 = 0
11/2 = 1
12/2 = 0
13 /2 =1

12/2 = 6
12%2=0

N
N%2= REMAINER = ANY NUMBER DIVIDED BY 2 GIVES ONLY 2 POSIBILITIES
EITHER 0 OR 1
2)13(6
12

1

```
#!/bin/bash
N_ARGS=$#

if [ $N_ARGS -ne 1 ]
then
echo "Input a number"
exit
fi

N=$1
REM=$((N%2))

if [ $REM -eq 0 ]
then
echo "$N is even"
else
echo "$N is odd"
fi
```

expr only works with integer inputs in performing mathematical operation.To apply on floating point numbers there is an another shell utility "bc" standards for basic calculator.
- numbers
- floating
- relational

if we pass an expression as an input it performs operation and gives the result.

```
N1=10.4
N2=11.3
M=$(echo "$N1 * $N2"|bc)
```

HOTEL - BILLING PROGRAM

1. IDLY = 12.5
2. DOSA = 20.5
3.CHAPATHI = 18
4.POHA = 22

ITEM=1
QUANTITY=2

BILL_AMOUNT=12.5*2 = 25

```
#!/bin/bash
echo "1. IDLY"
echo "2.DOSA"
echo "3.CHAPATI"
echo "4.POHA"

printf "select 1 item from the above"
READ ITEM

if [ $ITEM -lt 1 -o $ITEM -gt 4 ]
then
echo "ERROR: invalid selection, please select items from 1 to 4 options"
exit
fi


printf "quantity:"
READ QUANTITY

if [ $ITEM -eq 1 ] #idly
then
BILL_AMOUNT=$(echo "12.4*$QUANTITY"|bc)
elif [ $ITEM -eq 2 ]
then
BILL_AMOUNT=$(echo "20.5*$QUANTITY"|bc)
elif [ $ITEM -eq 3 ]
BILL_AMOUNT=$[18 * QUANTITY]
else
BILL_AMOUNT=$[22 * QUANTITY]
fi

echo "BILL AMOUNT: $BILL_AMT"
```

if-else = When there are only 2 possible outcomes of an expression rather than writing 2 IF conditions for executing the statements we can use if-else-fi structure.

```
N=$1
REM=$[N%2]
```

in the above lines of code the REM can have only either 0 or 1 and if it is zero we want to print even number if it is 1 we want to print odd number. So the lines can be written as below.

```
if [ $REM -eq 0 ]
then
echo "even"
fi
if [ $REM -eq 1 ]
then
echo "odd"
fi
```

here the second if statement is unnecessary because if it is not zero then the REM will be one only as there is no other possibility. In such a case writing second if condition wastes the additional cpu comparisions and kills performance instead use else as shown below.

```
if [ $REM -eq 0 ]
then
echo "even"
else
echo "odd"
fi
```

---

bc = basic calculator
To perform arithematic operations we have expr but it works only for integers, in case if we want to perform operations on floating point numbers the we need to use bc
bc takes expression as an input and evaluates and writes the output to a console.

```
A=10
B=20
SUM=$(echo "$A+$B" | bc)
```

---

if-elif-elif-else-fi = is called if-else-if ladder

If there are multiple possibilities out of a expression, to match one of these possiblities and perform the operation we have to write multiple if conditions.

```
printf "select one item"
read ITEM
1.
2.
3.
4.

if [ $ITEM -eq 1 ]
then
// do this
```

```
fi
if [ $ITEM -eq 2 ]
then
//do this
fi
if [ $ITEM -eq 3 ]
then
//do this
fi
if [ $ITEM -eq 4 ]
then
//do this
fi
```

Here when ITEM matches with one of the condition then there is no chance of matching with other conditions,so we can say all these if conditions are mutual exclusive. So writing multiple conditions will evaluates all of them but only matches to one which wastes CPU capacity and kill performance.
To rescue us from this if-elif-elif-else-fi comes into picture.

```
if [ $ITEM -eq 1 ]
then
//do this
elif [ $ITEM -eq 2 ]
then
//do this
elif [ $ITEM -eq 3 ]
// do this
elif [ $ITEM -eq 4 ]
// do this
else
// invalid option
fi
```

In the above the expression starts from top and evaluates until it matches one of the condition and skips the rest when one matched. This saves performance

PROPERTY QUOTATION GENERATOR

FLAT TYPE: 2 BHK, 2.5 BHK, 3 BHK, 4 BHK
2 BHK = 33 lacs
2.5 BHK = 41 lacs
3 BHK = 55 lacs
4 BHK = 68 lacs

FACING
1. EAST
2. NON-EAST
= BASE PRICE + 10% BASE PRICE
33 + 3.3 = 36 lacs

FLOORS = 22 Flooors
1 - 5 = No charges
6 - 15 = 2% base price

16 - 22 = 3% base price

PARKINGS = 1.5 lacs
MAX = 2

TOTAL FLAT COST

electricity bill

UNITS_CONSUMED=100
CONNECTION_TYPE=COMMERCIAL/DOMESTIC

DOMESTIC
0 - 100 = 1 UNIT - 1.50
101 - 250 = 1 UNIT - 2.30
> 250 = 3.75

COMMERCIAL
0 - 100 = 1 UNIT 5.30
101 - 250 = 1 UNIT - 8.20
> 250 = 9.50

120 - DOMESTIC
100 * 1.50 + (20 * 2.30)

```
#!/bin/bash
printf "Units Consumed :"
read UNITS_CONSUMED
printf "Connection Type (COMMERCIAL/DOMESTIC) : "
read CON_TYPE

if [ $CON_TYPE = "DOMESTIC"] # if it is domestic connection execute this block of code
then
if [ $UNITS_CONSUMED -le 100 ]
then
AMT=$(echo "$UNITS_CONSUMED * 1.5" | bc)
echo "UNITS CONSUMED IS $UNITS_CONSUMED AND BILL AMOUNT : $AMT"
elif [ $UNITS_CONSUMED -gt 100 -a $UNITS_CONSUMED -le 250 ]
then
AMT = $(echo "(100 * 5.30) + (($UNITS_CONSUMED-100)*8.2)" | bc)
echo "UNITS CONSUMED : $UNITS_CONSUMED and BILL AMOUNT: $AMT"
else
AMT = $(echo "(100 * 5.3) + (150*8.2) + (($UNITS_CONSUMED-250)*9.5)" | bc)
echo "UNITS CONSUMED : $UNITS_CONSUMED and BILL AMOUNT: $AMT"
fi
elif [ $CON_TYPE = "COMMERCIAL"] # if it is commercial execute this block of code
then
if [ $UNITS_CONSUMED -le 100 ]
then
AMT=$(echo "$UNITS_CONSUMED * 5.30" | bc)
echo "UNITS CONSUMED IS $UNITS_CONSUMED AND BILL AMOUNT : $AMT"
elif [ $UNITS_CONSUMED -gt 100 -a $UNITS_CONSUMED -le 250 ]
then
AMT = $(echo "(100 * 1.5) + (($UNITS_CONSUMED-100)*2.3)" | bc)
echo "UNITS CONSUMED : $UNITS_CONSUMED and BILL AMOUNT: $AMT"
else
AMT = $(echo "(100 * 1.5) + (150*2.3) + (($UNITS_CONSUMED-250)*3.75)" | bc)
echo "UNITS CONSUMED : $UNITS_CONSUMED and BILL AMOUNT: $AMT"
fi
fi
```

String operators = are used for string comparisions.
=
\>
\<
-z
-n
$str

control statements
- conditional control statements
2 types are there
if
case

if statement
if else fi
if elif else fi ladder
if - if-else- else fi nested if-else

string operators
string operators are used for comparing strings
= : 2 strings are same or not
\> (or) \>= : 2 given string which is bigger
\< (or) \<= : 2 given strings which is smaller
-n = non-empty
-z = zero / empty string
str = empty

File operators
File operators are used for performing conditional checks on files.

-e = is file or directory exists
-f = is it a file or not
-d=is it a directory or not
-r
-w
-x
-o
-s

search for a word in a file
location file
word

found/not found

text-search.sh

```
#!/bin/bash
N_ARGS=$#
if [ $N_ARGS -ne 2 ]; then
echo "required filename and word to search"
exit
fi

FILE_LOC=$1
WORD=$2

if [ ! -f $FILE_LOC ];then
echo "$FILE_LOC doesnt not exists"
```

```
exit
fi
if [ ! -r $FILE_LOC ]; then
echo "$FILE_LOC doesnt have permission"
exit
fi

N_MATCHES=$(grep -wo $FILE_LOC $WORD | wc -l)

if [ $N_MATCHES -gt 0 ]; then
echo "found"
else
echo "not found"
fi
```

---

case is also an conditional control statement similar to if condition. Based on the outcome out of an expression we want to execute a block of code then use case or if condition.

in case of if condition we can do any comparision for e.g.. we can use all relational operations like greater than, less than or any other. and both the sides of the operators can be expressions. but case will be used for picking up the exact match based on output, it only works for matching equal conditions only and cannot apply any conditional operators.

```
prepaid
postpaid
read -p "enter your choice" CHOICE

10 if elif else blocks = 10 comparisions
if [ "$CHOICE" = "PREPAID" ]; then
// do this
else if [ "$CHOICE" = "POSTPAID"]; then
// do this
fi

case "$CHOICE" in
"PREPAID")
// do this
"POSTPAID")
// do this
esac
```

Conditional Control Statements
if statement
case statement

case control statement
---
1. Based on fixed set of outcomes we want to match the outcome and want to perform the operation or execte a block code,then we go case control statement.
it should be used for equality matching only.

In case of fixed outcomes and based on the matching outcome if we want to execute a block of code case seems to much efficient ratherthan if condition.

```
if [ $CHOICE = "prepaid"] ; then

elif [ $CHOICE = "postpaid" ]; then

fi
```
in the above in worst case we need 2 conditional checks to be made to determine the block of code to be executed.

```
case $CHOICE in
"prepaid")
"postpaid")
esac
```
in case there is only 1 lookup required for identifying the relevant block of code to be executed.

syntax:-
4

```
case $INPUT in
1)
// lines of code
;;#takes the control out of esac
2)
// lines of code
;;
3)
// lines of code
;;
*) #global case
// lines of code
esac
// lines of code
```

Case Conditional Statement
There are fixed set of possible outcomes out of which we need match one of them to perform an operation then we can go for Case statement.For these type of conditional matching case is ideally suitable than if-else-fi conditions.

Syntax of Case:-

```
case INPUT in
OPTION1)
statements
;;
OPTION2)
statements
;;
*) [Global]
statements
esac
```

Loop control statements
Flow control statements

```
I=1
whille [ $I -le 10 ]
do
echo "$I"
I=$[I+1]
done
```

10
1 - 10 all even numbers to be printed

1 - div/2 = reminder check even or odd
2 - div/2 = reminder check even or odd
3- div/2 = reminder check even or odd

```
I=1
read N

while [ $I -le $N ]
do
REM=$[I%2]
if [ REM -eq 0 ]; then
print '%d' $I
fi
I=$[I+1]
done
```

Loop control statements

#how can we make sure the bunch of lines of code can be executed for certain number of times. If we want repeatitively execute the code for certain amout of time we need to use counter variable and build loop condition based on the number of times it has to be executed.

MULTIPLICATION TABLE
2nd = 1 - 10 = 2 table should be printed

2*1 = 2
2*2 = 4
2*3=6
2*4=8
2*5=10
2*6=12
2*7=14
2*8=16
2*9=18
2*10=20

10 times
n1*n2=ouput

```
#!/bin/bash
read -p "enter which table :" N
I=1

while [ $I -le 10 ]
do
MUL=$[I*N]
printf "%d * %d = %d\n" $N $I $MUL
I=$[I+1]
done
```

*
* *
* * *
* * * *
* * * * *

- we need to print 5 lines repeatedly means i need one loop
- for each line i want to println repeatedly * = one more loop = the no of starts to be printed in every line depends on line we are printing

1 st
1- stars
2nd
2 stars
3rd
3 stars

```
#!/bin/bash
read -p "no of lines to be printed" N
I=1
```

```
while [ $I -le $N ]; do
J=1
while [ $J -le $I ]; do
printf "* "
J=$[J+1]
done
echo ""
I=$[I+1]
done
```

N=13
if a number is divisible by 1 or ifself then it is called prime number.

12/1 =0
13/13=0

12 = is divisible by only numbers which are falling into 2 and half of the number
2 - 6 = cannot divde then 13 is prime

12/2 =0
12/3=1
13/4=1
13/5=2
13/6=1

REM=N%COUNTER

```
#!/bin/bash
read -p "Enter Number you want to check :" N
ISPRIME=1
I=2
END=$[N/2]

while [ $I -le $END ]; do
REM=$[N%I]
if [ $REM -eq 0 ]; then
ISPRIME=0
break
fi
done

if [ $ISPRIME -eq 1 ]; then
echo "prime number"
else
echo "not a prime number"
fi
```

N=12
if it is divisible only by 1 or itself then it is called prime number

N has been given we should try dividing the number with numbers between 2 to half of the number N/2 and if it is not divisilble by any numbers then it is a prime number

```
#!/bin/bash
read -p "enter N :" N
I=2
END=N/2
ISPRIME=1 // consider the number N as prime number

15
2 = 7
I=2
END=7

while [ $I -le $END ]; do
REM=$[N%I]
if [ $REM -eq 0]; then
ISPRIME=0
break #break statement is used for stopping the loop and take the control outside
fi
//
I=$[I+1]
done

if [ $ISPRIME -eq 0 ]; then
echo "not prime"
else
echo "prime"
fi
```

FOR Loop

The purpose of WHILE and FOR are same as "If we want to execute a block of code repeatedly" then use Loops, but here we achieve it in a different type of syntax.

#1 For loop = For can iterate over fixed set of inputs. of we can iterate over fixed input items. Here we are not specifying the range like begin index and end index rather we are passing directly the inputs over which it should iterate

```
for i in 1 2 3 4 5 6
do
print $i
done
```

Here I first takes first item in second iteration takes 2nd item in the list and iterate.

wap that takes non-interactive input of 10 numbers and should sum and print

```
#!/bin/bash
SUM=0
```

```
for I in $*
do
SUM=$[I+SUM]
done

printf '%d' $SUM
```

---

directory -> list all the files of the directory

```
stories
|-chapter1.txt
|-chapter2.txt
|-chapter3.txt
|-introduction

read -p "enter directory you want to navigate :" DIR

for F in $DIR
do
if [ -f $F ]; then
echo $f
fi
done
```

For Loop:-
It we have some lines of code to be executed repeatedly at a certain place with in the shell script instead of duplicating the code, we can put them in a loop like for.

while loop always executed based on a condition, where as a for loop can be written to iterate based on fixed input of values as well.

```
for I in 1 2 3 4 5 ; do
println $I
done

for arg in $*; do
#iterate over all the shell script arguments
done

for f in dir/*; do
# iterate over all the files of the given directory
done
```

arguments
directories

```
dir1
|-file1
|-file2
|-subdir1
dir2
|-file3
|-subdir2
file4
```

yourShell.sh dir1 dir2 file4

```
#!/bin/bash
WORDCOUNT=0

#validation
for F in $*
do
if [ ! -e $F ] ; then
echo "ERROR: File or Directory not found $F"
exit
fi
done

for F in $*
do
if [ -f $F ]; then
C=$(wc -w $F)
WORDCOUNT=$[WORDCOUNT+C]
else if [ -d $F ]; then
for SF in $F/*
do
if [ -f $SF]; then
```

```
C=$(wc -w $SF)
WORDCOUNT=$[WORDCOUNT+C]
fi
done
fi
done


I=0
while [ $I -le 10]
do
I=$[I+1]
done

for ((initialize;condition;increment))
do
done

for ((I=0;$I -le 10; $((I++))))
do
print $I
done
```

Networking Concepts
- How does computer systems works over the network
- What are protocols why do we need to use them?
- Application Protocols
- Transport Protocols [OSI Model Layer - Highlevel]

Basics of Networking

Every computer inorder to communicate with other computer over the network it requires an Network Adapter called NIC Card "Network Interface Card". NIC Cards are plugged into the Motherboard of the Computer and these cards are Plugged in with RJ45 Socket Network Cables connecting computers.

A Computer can have multiple NIC Cards Plugged-In in order to inter-connect with multiple computers.

How does a computer can communicate with another computer after inter-connecting them?
Every computer inorder identify itself on the network it requires an unique address called "ip address". Ip address is a logical network address that is available for every computer on a network. The ip address is unique within the network, so that a computer can communicate to another computer using the ip address of the others.

What is mac address?
"ip address" is a logical address that is generated whenever a computer is brought on to the network. Whereas the Network Interface Card has another address seeded inside it by the manufacturer that uniquely identifies your computer within others which is "Mac Address" or "Physical Address" of the computer.

Then why dont we use mac address for communicating?
These are not user friendly and cannot be memorized.

Who communicates over the network in a computer?
No computers will communicate with each other in exchanging the data over the network. Programs running in the computers wants to exchange data with another program running on another computer, so programs wants to communite over the network.

Here "ip address" allows you to identify Computer, but we need to pass the data to a specific program running inside the other computer. So a program by itself has to register with operating system wishing asking operating system to pass the data that it recieved over the network. So operating system allots one nuemeric number to the program called "port no" to identify itself in that computer.

The program looking for receiving data over the network only has to register a portNo with operating system,every program will not have portno in a computer.

In order to send the data across the network to another program we need to use ipaddress and port no also.

portNo is required only for the program which wants to receive the data from the network.

How does the data transfer takes place over the network connection?
Network/Physical hardware devices are not capable of trasmitting the data in symbolic represenation format. They are capable of only sending singals either as off or on. If some how some way if we can express the data interms of singals off/on then the information can be transferred over the n/w from one device to the another one.

How to represent information interms of singals as off/on?
We can express the data interms of 0's and 1's indicates 0 as off and 1 as on in a binary format. in order to transform the data into binary or 0/1 format we need to use character set encoding standards.

The charset encoding standards helps us in expressing the characters/symbols in a nuemric represenation so that it can converted into binary format and can transfered over the network. There are many charset encoding standards are available.
1. ASCII
2. utf-8
3. UNICODE
4. ISO-9001
5. utf-32

Both the senders and the receiver should use the same charset standard in exchaning the data to understand.

How many forms we can write a for loop?
There are 2 forms of for loop are there
1. In fixed items iterate
2. counter based for loop

1. In fixed items loop we pass the inputs over which the loop should iterated and every iteration we read an item from the input.

```
for i in 1 10 11 23 78; do
done
```

2. Counted based loop :- here we define initialization, condition and incrementation so that over fixed set of iterations can be achieved

```
for((initialization;condition;incrementation))
do
done
```

Until Loop:-
This is also same as while but until condition is false the loop iterates, once condition becomes true the loop exits.

```
while [ condition ]; do
done
```

In the above iterate set of statements between do and done till the condition becomes false.

```
until [ condition ]; do
done
```

In until loop iterate if condition false,if condition becomes true break;

```
#!/bin/bash
I=1
while [  $I -le 10 ];do
echo $I
done

I=1
until [ $I -gt 10 ]; do #until I becomes greater than 10 loop it
echo $I
done
```

What are Arrays?
Arrays are collection of items/elements/values.

```
NAME1='Paul'
NAME2='Joseph'
NAME3='Micheal'
NAME4='Bob'
NAME5='James'
```

In a variable we can store at max one value, if i have group of related values then we need to create multiple variables to hold them.

```
LOAN_AMOUNT=120000
12.25

INTEREST_AMOUNT=$(echo "($LOAN_AMOUNT * 12 * 12.25) / 100" | bc)

LOAN_AMOUNT1=120000
LOAN_AMOUNT2=150000
LOAN_AMOUNT3=175000
LOAN_AMOUNT4=225000
LOAN_AMOUNT5=100000
```

all the above numbers are similar type of values which are nothing but loan amount.

```
INTEREST_AMOUNT=$(echo "($LOAN_AMOUNT1 * 12 * 12.5 )/100" | bc)
echo "Interested To be paid : $INTEREST_AMOUNT For LOAN_AMOUNT: $LOAN_AMOUNT1"

INTEREST_AMOUNT=$(echo "($LOAN_AMOUNT2 * 12 * 12.5)/100" | bc)
echo "Interested to be paid : $INTEREST_AMOUNT For LOAN_AMOUNT: $LOAN_AMOUNT2"

INTEREST_AMOUNT=$(echo "($LOAN_AMOUNT3 * 12 * 12.5)/100" | bc)
echo "Interested to be paid : $INTEREST_AMOUNT For LOAN_AMOUNT: $LOAN_AMOUNT3"
```

In the above loan amounts are similar type of values for which we have same type of logic to be applied
if we declare these values in different variables then we need to duplicate the functionality for changing variables.

For similar type of values we apply same processing logic defining them in separate variables will duplicates the code rather use arrays.

In any array we can store multiple values.

```
LOAN_AMOUNTS = (120000 150000 175000 225000 100000)
```

How to access the data?
Array values are stored in sequential in adjacent memory locations with an index starting from '0', we can access the values of an array using index number say for e.g..

```
${ARRAY[0]} = zero element value
${ARRAY[1]} = first element value
```

How to modify the data of a location?

```
ARRAY[0]=value
```

An array can have any number of elements, then how to find how many elements are there?

```
${#ARRAY[*]}
```

How to access all the elements of the array?

```
${ARRAY[*]}
${ARRAY[@]}
```

How to iterate one element after another in an Array?

```
A=(10 20 30 40 50)
LEN=${#A[*]}
I=0

while [ $I -lt $LEN ]; do
echo ${A[I]}
I=$[I+1]
done

for ((I=0;I<LEN;I++))
do
echo ${A[I]}
done

for E in ${A[*]}
do
echo $E
done


APPLICANTS=("Joe" "Juli" "Smith" "Susan" "Bob" "Angila")
echo "${APPLICATIONS[*]}"

I=1
LEN=${#APPLICANTS[*]}

while [ $I -lt $LEN ]; do
echo ${APPLICANTS[I]}
I=$[I+2]
done
```

What are Arrays?
Arrays are collection values or group of values and usually used for applying some common processing logic on those elements.

If there are group of values on whom we want to apply some common logic if we store them in separate variables the logic should be duplicated for each of the variables. Instead store them in an Array so that we can apply common logic on those group of values by iterating.

How to create/declare Array?

#1
VARIABLE=(ELEM1 ELEM2 ELEM3)

#2
declare -a ARRAYNAME
In above it creates an empty ARRAY with no values
ARRAYNAME[index]=VALUE

#3
read -a ARRAYNAME
above line helps us in reading the elements of an array and create it interatively

Arrays stores the data sequentially in the adjacent memory locations and all the values can be accessed using index location. Index starts with zero and ends depends on the no of elements in the array.

How to find end index of the array (or) length of an array
${#ARRAY[*]}

In general arrays are called indexed collection, means we can access / store values using position. Linux bash arrays supports named collection (or) map collection also.
associated arrays

declare -a MARKS
MARKS[0]=90
MARKS[1]=100

MARKS['MATHS']=100
MARKS['SOCIAL']=90

#!/bin/bash

read -p "enter loan amounts to calculate interest amount : " -A LOAN_AMOUNTS
read -p "enter rate of interest :" RI
declare -A INTEREST_AMOUNT

LEN=${#LOAN_AMOUNTS[*]}
for((I=0;I<LEN;I++)); do
AMOUNT=$(echo "(${LOAN_AMOUNTS[$I]} * 12 * $RI)/100" | bc)
INTEREST_AMOUNT[$I]=$AMOUNT

```
done

echo ${INTEREST_AMOUNT[*]}
```

functions
What are functions why do we need to use them?
Functions are block of code that can be reused at various different places in the shell program so that we can avoid duplicating the code across the shell program and ease the maintainance of the program.

How to write functions in shell script?

```
#!/bin/bash
echo "1"
echo "2"
echo "3"
echo "4"
echo "5"
echo "6"
echo "7"
echo "2"
echo "3"
echo "4"
echo "8"
echo "9"
echo "10"
echo "2"
echo "3"
echo "4"
#!/bin/bash

#function block
function print() {
echo "2"
echo "3"
echo "4"
}

#main block
echo "1"
print
echo "5"
echo "6"
echo "7"
print
echo "8"
echo "9"
echo "10"
print
```

by the above code we can understand there are 2 parts in the program one is function part and another part is main block of program.

always the bash interpreter starts executing the main block of the program and functions by default will not be executed. function blocks are only executed when those are called from main block of code.

We can call a function in main block by just referring the name of the function.
As we stated about we have 2 parts being called as
- callee = the main block that calls the function
- caller = the function that is being called.

When a main block calls the function automatically the control of execution will be passed to function.
once the function block finished execution then the control comes back to the next line of statement in the main block after the function call.

```
#function block
function add() {
A=$1
B=$2

SUM=$[A+B]
echo $SUM
}

#main block
echo "1"
echo "2"
echo "3"
add 10 20 // passing parameters
echo "4"
echo "5"
echo "6"
add 20 30
.
.
.
add 50 60
```

In the above program whenever we call add it always adds only A=20, B=20 instead of this I want add function to be executed with different values taking as input and perform operations.
That is where we need to pass parameters to function.

What is a function and what is the purpose of it?
Block of code that can be executed at different places with in a shell program.

While working with functions our shell program contains 2 parts.
1. function blocks
2. Main block

The program execution starts from main block and executed from top-down. By default functions will not be executed unless we call them explicitly.

```
function doSomething() {}
#main block
doSomething
```

function parameters

```
function prime() {
N=$1

}
#main block
prime 13
```

---

scope of variables

```
#function blocks

function func() {
local K=10
J=20
echo $I
}

#main block
I=10 #global scope
func
echo $K
```

---

exit code of a program can be grabbed by using $?
bash returns '0' for successful termination and non-zero to indicate the failure.

---

wap to sum the even numbers in a given array.

```
// identifying the even numbers in the array
// creating another array with even numbers only
// adding all those even numbers of that array.

#function blocks

function isEvenNumber() {
local N=$1

if [ $N % 2 -ne 0 ]; then
```

```
F=0
fi
}

function extractEvenNumbers() {
F=1
LEN=${#NUMBERS[*]}

for((i=0;i<LEN;i++)); do
local N=${NUMBERS[i]}
F=1
isEvenNumber $N
if [ $F -eq 1 ]; then
EVEN_NUMBERS[$INDEX]=$N
INDEX=$[INDEX+1]
fi
done

return 0 #exit code of execution
#return ends the execution of the function and returns exit code to the callee.
}

function addEvenNumbers() {
for((i=0;i<${#EVEN_NUMBERS[*]};i++)){
SUM_EVEN_NUMBERS=$[SUM_EVEN_NUMBERS+EVEN_NUMBERS[I]]
}
}




#main block
declare -A NUMBERS #global variable
declare -A EVEN_NUMBERS
INDEX=0
SUM_EVEN_NUMBERS=0
read -p "enter numbers :" -a NUMBERS

// extract even numbers and create another array from it.
extractEvenNumbers // after extractEvenNumbers or function completed successfully?
STATUS=$?
if [ $STATUS -eq 0 ]; then
addEvenNumbers
echo "Sum of evens : $EVEN_NUMBERS"
else
echo "ERROR: extracting even numbers"
exit
fi
```

In shell functions a function cannot return computed output inside it to main block or other functions or to simply put it callee.

local / global variables scopes
exit codes of a program
how can we return returnValue from a function

There are 2 scopes at which we can declare variables.
1. global variable = We can access the variable throughout the shell program.
2. local variable = Local variables are defined inside the functions and are accessible within the function only.
How to declare local variables?
local VAR_NM=VAL
by default every variable in shell program is global unless we use local keyword in declaring them.

exit code of a program

It indicates the status of execution of a program. Whether the program completed its execution sucessfully or failed we can know by using exit code.
How to access the exit code of a program?
$? = helps us in getting the exit code of a last executed program.

0 = indicates success
non-zero = indicates failure

Can shell functions can return returnValue or not?
Functions in shell script cannot return returnValue to the other blocks of code, in order to return returnValue we need to use global variables.

Then what does an return statement indicates in shell functions?
To let the other block of code to understand whether the function completed execution successfully / failed we return exitCode using return statement.

wap a program that takes array of numbers and find and return another array with prime numbers within them.

NUMBERS = [10, 23, 17, 19, 15]

PRIME_NUMBERS= [23, 17]

findprimes.sh

```
#function blocks

function isPrime() {
local N=$1
local MAX=$[N/2]

for((INDEX=2;INDEX<=MAX;INDEX++)); do
if [ $[N%INDEX] -eq 0 ]; then
F=0
break;
fi
done
}
```

```
function findPrimes() {
local LEN=${#NUMBERS[*]}
F=1

for((i=0;i<LEN;i++)); do
local N=${NUMBERS[i]}
F=1
isPrime $N
if [ $F -eq 1 ]; then
PRIME_NUMBERS[$PRIME_INDEX]=$N
PRIME_INDEX=$[PRIME_INDEX+1]
fi
done
return 0 #to demonstrate what is a return statement in function
}

# main block
#!/bin/bash
declare -A NUMBERS #global variables so we dont need to pass parameters
delcare -A PRIME_NUMBERS
PRIME_INDEX=0
read -p "enter numbers: " -a NUMBERS
findPrimes
STATUS=$?
if [ $STATUS -eq 0 ]; then
echo PRIME_NUMBERS
else
echo "ERRROR! findprimes failed"
fi
```

---

ifconfig = shows the network adapters and ip address of the machine.
by default ifconfig is not available as part of ubuntu install you need to install a package called "nettools"
sudo apt install net-tools

ping ipaddress = to check whether we can access another machine of an ip address or not
scp = secure copy
scp sourcefile user@ipaddress:/destinationLocation

ssh = secure shell to access remote linux machine.
ssh user@ipaddress

Linux and Shell Scripting [developer] 1 month, 15 days [2 months]