LINUX

-Linux is widely used opensource operating system similar to windows mac and android

-it shares similarities with unix another operating system known for its commercial use

-unix and linux have comparable elements including the kernel shell and programs

-many commands in unix and linux exhibit similar behaviours and syntax

-learning the basics of linux helps you understand how this powerful OS works why its widely used in various fields and how it differs from windows with its widely used in various fields and how it differs from windows with its open source nature better security and flexibility in customizations

Linux commands Functions

1. Ls commad = displaying info about files in current directory
2. Pwd command = displays the current working directory
3. Mkdir command = creates a directory
4. Cd command = to navigate between different folders
5. Rmdir command = removes empty directory from the directory files
6. Cp command = copy files from one directory to another n
7. Mv command = rename and replace the file
8. Rm command = delete files
9. Uname command = command to get basic information about os
10. Locate command = find a file in db
11. Touch command = create empty files
12. Ln command = create shortcuts to other files
13. Cat command = display file content on terminal
14. Clear command = clear terminal
15. Ps command = display the process in terminal
16. Man command = access manual for all linux commands
17. Grep command = search for a specific string in an output
18. Echo command = print string or text to the terminal
19. Wget command = download files from internet
20. Whomai command = displays the current user name
21. Sort command =sort the file content
22. Cal command = view calendar in terminal
23. Whereis command =view the exact location of any command typed after this command
24. Df command = check the details of the file system
25. Wc command = check the lines word count and characters in a file using different options

Linux file hierarchy structure

- In the FHS, all files and directories appear under the root directory /, even if they are stored on different physical or virtual devices.

- Some of these directories only exist on a particular system if certain subsystems, such as the X Window System, are installed.
- Most of these directories exist in all UNIX operating systems and are generally used in much the same way; however, the descriptions here are those used specifically for the FHS and are not considered authoritative for platforms other than Linux.

/root directory

1./bin/ - essential user command binaries

2./boot/-static files of the boot loader

3./dev/-device files

4./home/ - user home directories

5./lib/-shared libraries

6. /media/-removable media

7. /mnt/-mounted ecosystem

8./opt/-add on application software package

9./srv/-data for service from system

10./tmp/-temporary files

11. /usr/-ser utilities and applications

12./proe/-process information

1./(Root)

Primary hierarchy root and root directory the entire file system hierarchy

-every single file and directory start from the root directory

-the only root user has the right to write under this directory

-/root is the root users's home directory which is not the same as /

2./bin/-

Essential command binaries that need to be available in single user mode for all users eg cat,ls,cp

-contains binary executables

-common linux commands you need to use in single user modes are located under this directory

-commands used by all the users of the system are located here ps ping ,grep,cp,ls

/boot:

Boot loader files eg-kernel initrd

-kernel initrf vmlinux grub files are located under /boot

-example initrd img-2.6.32-24 generic vmlinuz-2.6.32-24-generic

4./dev;

Essential device files eg. /dev/null

-these include terminal devices usb,or any device attached to the system

-example dev/tty1,/dev/usbmon0

5./etc

Host-specific wide configurations files

-contains configurations files required by all programs

-this also contains startup and shutdown shell scripts used to start/stop individual programs

-example ;/etc /resolv.conf /etc/logrotate.conf.

6./home

User home directories containing saved files personal settings etc

-home directories for all users to store their personal files

-example : /home/kishlay/, home/kv

7./lib:

Libraries essential for the binaries in /bin/ and /sbin/

-library filesnames are either id* or lib*.so.*

-example:Id-2.11.1.so libncurses.so.5.7

8./media :

Mount points for removable media such as cd -rom

-temporary mount directory for removable devices

-examples ,media/cdrom/ for cd-ROM; ?media/floppy drives;/media/cdrecorder for cd writer

9. /mnt:
Temporarily mounted filesystems

-temporary mount directory where sysadmins can mount filesystems.

10. ./opt:

Optional application software packages

-contains add on applications from individual vendors

-add on applications should be installed under either /opt/or /opt/sub/directory

11. /sbin :

Essential system binaries, e.g., fsck, init, route.

- Just like /bin, /sbin also contains binary executables.
- The linux commands located under this directory are used typically by system administrators, for system maintenance purposes.
- Example: iptables, reboot, fdisk, ifconfig, swapon

12. /srv :

Site-specific data served by this system, such as data and scripts for web servers, data offered by FTP servers, and repositories for version control systems.

- srv stands for service.
- Contains server specific services related data.
- Example, /srv/cvs contains CVS related data.

13. /tmp :

Temporary files. Often not preserved between system reboots and may be severely size restricted.

- Directory that contains temporary files created by system and users.
- Files under this directory are deleted when the system is rebooted.

14. /usr :

Secondary hierarchy for read-only user data; contains the majority of (multi-)user utilities and applications.

- Contains binaries, libraries, documentation, and source-code for second level programs.
- /usr/bin contains binary files for user programs. If you can't find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp

- /usr/sbin contains binary files for system administrators. If you can't find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel
- /usr/lib contains libraries for /usr/bin and /usr/sbin
- /usr/local contains user's programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2
- /usr/src holds the Linux kernel sources, header-files and documentation.

15. /proc:

Virtual filesystem providing process and kernel information as files. In Linux, it corresponds to a procs amount. Generally, automatically generated and populated by the system, on the fly.

- Contains information about system process.
- This is a pseudo filesystem that contains information about running processes. For example: /proc/{pid} directory contains information about the process with that particular pid.
- This is a virtual filesystem with text information about system resources. For example: /proc/uptime

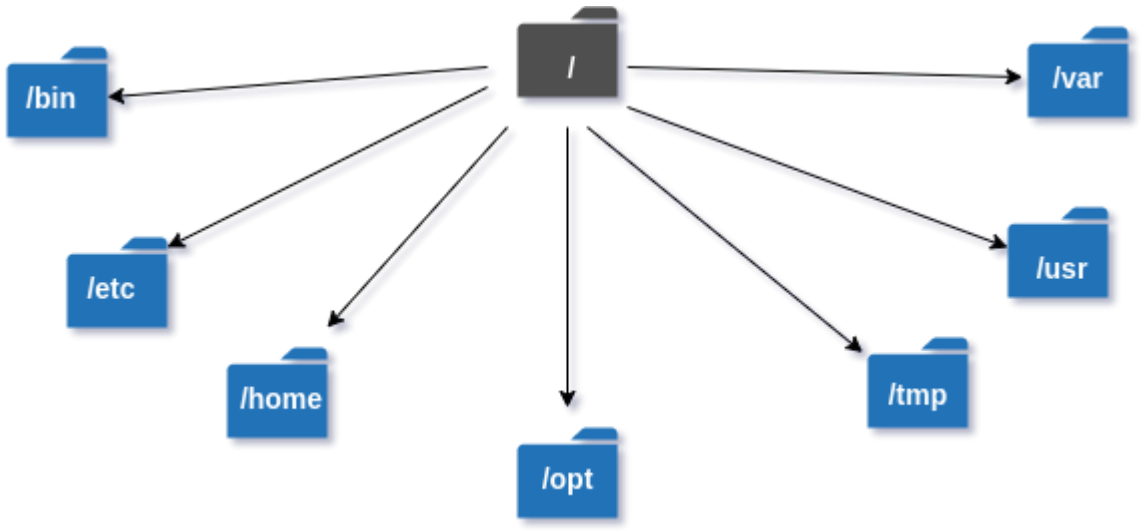
LINUX DIRECTORY STRUCTURE

In Linux/Unix operating system everything is a file even directories are files, files are files, and devices like mouse, keyboard, printer, etc are also files. Here we are going to see the Directory Structure in Linux.

Types of files in the Linux system.

1. **General Files** – It is also called ordinary files. It may be an image, video, program, or simple text file. These types of files can be in ASCII or Binary format. It is the most commonly used file in the Linux system.
2. **Directory Files** – These types of files are a warehouse for other file types. It may be a directory file within a directory (subdirectory).
3. **Device Files –** In a Windows-like operating system, devices like CD-ROM, and hard drives are represented as drive letters like F: G: H whereas in the Linux system devices are represented as files. As for example, /dev/sda1, /dev/sda2, and so on.

   We know that in a Windows-like operating system, files are stored in different folders on different data drives like C: D: E: whereas in the Linux/Unix operating system files are stored in a tree-like structure starting with the root directory as shown in the below diagram.



These are the common top-level directories associated with the root directory:

Directories	Description
/bin	binary or executable programs.
/etc	system configuration files.
/home	home directory. It is the default current directory.
/opt	optional or third-party software.
/tmp	temporary space, typically cleared on reboot.
/usr	User related programs.
/var	log files.

Some other directories in the Linux system:

Directories	Description
/boot	It contains all the boot-related information files and folders such as conf, grub, etc.
/dev	It is the location of the device files such as dev/sda1, dev/sda2, etc.
/lib	It contains kernel modules and a shared library.
/lost+found	It is used to find recovered bits of corrupted files.
/media	It contains subdirectories where removal media devices are inserted.
/mnt	It contains temporary mount directories for mounting the file system.
/proc	It is a virtual and pseudo-file system to contains info about the running processes with a specific process ID or PID.
/run	It stores volatile runtime data.
/sbin	binary executable programs for an administrator.
/srv	It contains server-specific and server-related files.
/sys	It is a virtual file system for modern Linux distributions to store and allows modification of the devices connected to the system.

Exploring directories and their usability:

We know that Linux is a very complex system that requires an efficient way to start, stop, maintain and reboot a system, unlike Windows operating system. In the Linux system some well-defined configuration files, binaries, main pages information files are available for every process.

Linux Kernel File:

- **/boot/vmlinux –** The Linux kernel file.

Device Files:

- **/dev/hda –** Device file for the first IDE HDD.
- **/dev/hdc –** A pseudo-device that output garbage output is redirected to /dev/null.

System Configuration Files:

Configuration Files	Description
/etc/bashrc	It is used by bash shell that contains system defaults and aliases.
/etc/crontab	A shell script to run specified commands on a predefined time interval.
/etc/exports	It contains information on the file system available on the network.
/etc/fstab	Information of the Disk Drive and their mount point.
/etc/group	It is a text file to define Information of Security Group.
/etc/grub.conf	It is the grub bootloader configuration file.
/etc/init.d	Service startup Script.
/etc/lilo.conf	It contains lilo bootloader configuration file.
/etc/hosts	Information of IP and corresponding hostnames
/etc/hosts.allow	It contains a list of hosts allowed accessing services on the local machine.
/etc/host.deny	List of hosts denied accessing services on the local machine.
/etc/inittab	INIT process and their interaction at the various run levels.
/etc/issue	Allows editing the pre-login message.
/etc/modules.conf	It contains the configuration files for the system modules.

Configuration Files	Description
/etc/motd	It contains the message of the day.
/etc/mtab	Currently mounted blocks information.
/etc/passwd	It contains username, password of the system, users in a shadow file.
/etc/printcap	It contains printer Information.
/etc/profile	Bash shell defaults.
/etc/profile.d	It contains other scripts like application scripts, executed after login.
/etc/rc.d	It avoids script duplication.
/etc/rc.d/init.d	Run Level Initialisation Script.
/etc/resolv.conf	DNS being used by System.
/etc/security	It contains the name of terminals where root login is possible.
/etc/skel	Script that initiates new user home directory.
/etc/termcap	An ASCII file that defines the behavior of different types of the terminal.
/etc/X11	Directory tree contains all the conf files for the X-window System.

User related files

User Related Files	Descriptions
/usr/bin	It contains most of the executable files.
/usr/bin/X11	Symbolic link of /usr/bin.
/usr/include	It contains standard files used by C program.
/usr/share	It contains architecture independent shareable text files.
/usr/lib	It contains object files and libraries.
/usr/sbin	It contains commands for Super User, for System Administration.

Virtual and Pseudo Process Related Files	Descriptions
/proc/cpuinfo	CPU Information
/proc/filesystems	It keeps useful info about the processes that are currently running.
/proc/interrupts	it keeps the information about the number of interrupts per IRQ.
/proc/ioports	Contains all the Input and Output addresses used by devices on the server
/proc/meminfo	It reports the memory usage information.
/proc/modules	Currently using kernel module.
/proc/mount	Mounted File-system Information.
/proc/stat	It displays the detailed statistics of the current system.
/proc/swaps	It contains swap file information.

Version information file :
./version – it display the linux version information

Log files

Log Files	Descriptions
/var/log/lastlog	It stores user's last login info.
/var/log/messages	It has all the global system messages
/var/log/wtmp	It keeps a history of login and logout information.

SHELL SCRIPTING

If we are using any major operating system, we are indirectly interacting with the **shell**. While running Ubuntu, Linux Mint, or any other Linux distribution, we are interacting with the shell by using the terminal. In this article we will discuss Linux shells and shell scripting so before understanding shell scripting we have to get familiar with the following terminologies:

- Kernel
- Shell
- Terminal

Table of Content

What is Kernel?

The kernel is a computer program that is the core of a computer operating system with complete control over everything in system it manages the following resources of the linux system

-file management

-process management

-I/O management

-memory management

-device management

It is often mistaken that linus tovalds has developed linux os but actually he is only responsible for development of the linux kernel

What is shell?

A shell is a special program that provides an interface for the user to use operating system services shell accepts human readable commands from users and converts them into something which the kernel can understand.it is a command language interpreter that executes commands read from input device such as keyboards or from files. The shell gets started when the user logs in or starts the terminal.

Shell is broadly classified into two categories

-command line shell

-graphical shell

COMMAND LINE SHELL

Shell can be accessed by users using a command line interface A special program called terminal in linux/macos or command prompt in window os is provided to type in the human readable commands such as “cat”,”ls” etc then it is being executed.The result is then displayed on the terminal in ubuntu 16.4 system looks like this –

In the “ls” command with “-l” option will list all the files in the current working directory in a long listing format. Working with a command line shell is a bit difficult for beginners because its hard to memorize so many commands.

It is very powerful it allows users to store commands in a file and execute them together this way any repeatitive task can be easily automated. These files are usually called batch files in windows and shell scripts in linux/macos systems.

GRAPHICAL SHELL:

Graphical shell provides means for manipulating programs based on the graphical user interface by allowing for operations such as opening closing moving resizing windows as well as switching focus between windows

Windows os or ubuntu os can be considered as a good example which provides gui to the user for interacting with the program

Users do not need to type in commands for every action a typical Gui in the ubuntu system

There are several shells are available for linux system like

-BASH(Bourne Again Shell)-it is the most widely used shell in linux system it is used as default login shell in linux systems and in macos it can also be installed in windows Os

-CSH(C shell)- The C shell syntax and its usage are very similar to the C programming language

-KSH(Korn shell)- The korn shell was also the base the posix shell standard specifications

Each shell does the same job but understands different commands and provides different built in functions

WHAT IS TERMINAL

A program which is responsible for providing an interface to a user so he/she can access the shell

It basically allows users to enter commands and see the output of those commands in text based interface

Large scripts that are written to automate and perform complex tasks are executed in the terminal

To access the terminal simply search in search box “terminal” and double click it

SHELL SCRIPTING

As a shell can take commands as input from file we can write these commands in a file and can execute them in shell to avoid this repetitive work these files are called shell scripts or shell programs shell scripts are similar to the batch file in ms-dos each shell scripts is saved with .sh file extension eg myscript.sh

A shell script has syntax just like any other programming language

A shell script compromise the following command

Shell keywords - if else break etc

Shell commands -cd,ls,echo,pwd,touch,etc

Functions

Control flow-if then else case and shell loops etc

Why do we need shell scripts

-There are many reasons to write shell scripts:

-to avoid repetitive work and automation

-system admins use shell scripting for routine

-system monitoring

-Adding new functionality to the shell etc.

Some disadvantages of shell script

-prone to costly error a single mistake can change the command which might be harmful

-slow execution speed

-design flaws within the language syntax implementation

-not well suited for large and complex task

-provide minimal data structure unlike other sricpting language

Comparison operators

Interger comparison

Operator	Description
-eq	is equal to
-ne	is not equal to
-gt	is greater than
-ge	is greater than or equal to
-lt	is less than
-le	is less than or equal to
==	is equal to
!=	is not equal to
\<	is less than,in Ascii
/>	is greater than ,in Ascii

Conditional statement

If statement :-

It checks the condition and if it is conditioned true it executes the command

Syntax

If [condition]

Then

statement

fi

let see an example.

#!/bin/sh

x=10

y=11

if [$x -ne $y]

then

echo “Not equal”

fi

if-else statement

if an if -else statement you can specify a set of commands to run if the condition is not met

syntax

if [condition]

then

#set of statements if the condition is true

else

#set of statements if the condition Is false

fi

Lets see an example

#!/syntaxbin/sh

x=10

y=10

if [$x -ne $y]

then

echo “Not equal “

else

```
echo “They are equal “
fi
 $nano script.sh
$cat script.sh
#!/bin/sh
x=10
y=10
if [ $x -ne $y ]
then
echo “Not equal “
else
echo “They are equal “
fi
$./script.sh
```

Output:- They are equal

While loop

It starts running the specified commands if the condition is true and repeats them until the condition is false

Syntax

```
While [ condition ]
do
#set of statements
done
```

lets see an example

```
#!/bin/sh
X=2
While [  $x -lt 6  ]
do
echo  $x
x=` expr $x + 1`
```

done

$ nano script.sh

$ cat script.sh

#!/bin/sh

x=2

While [$x -lt 6]

do

echo $x

x=` expr $x + 1 `

done

./script.sh

Output:-

2

3

4

5

For loop

In a for loop the variable iterates over a list of values and ends when there are no more values to iterate over.

Syntax

for var in val1 val2 val3

do

#statements

done

Lets see an example .

#!/bin/sh

for var in 2 4 5 8

do

echo $var

done

POSTIONAL ARGUMENTS

Positional arguments are the arguments or values which we pass to the shell script while executing the script.They are accessed by variable $0,$2,....$9. Beyond that ,they are referenced by {$10} ,${11} and so on.$# stores the no of passed arguments and $0 stores the name of the script lets see an example to understand all this

#!/bin/sh

echo " No of arguments is $#"

echo "Name of the script is $0"
echo "First argument is $1 "

echo "Second argument is $2"
To pass the arguments just type them in the terminal after the script name as shown below

Storing the output of the commands

You can store the output of commands inside a variable in shell script there are two ways to do so

Syntax

Syntax 1

var =$(a valid linux command)

syntax 2-

var2=` a valid linux command `

lets see an example

#!/bin/sh

b=$(pwd)

c=`pwd`

echo $b

echo $c

d=$(ls /bin | grep bash)

echo $d

Exit codes of shell command

Whenever a command ends and returns the control of parent process it returns exot codes between 0 and 255 Exit code 0 means the command was successful and any other exit code means the command was unsuccessful you can view the exit code after running any command by accessing the $? Variable .

See the example below

You can mutually set an exit code for your shell script this can be used with conditional statements to convey if the script purpose was achieved or not.

Example

```
#!/bin/sh

read x

if [ $x  -ne  $10 ]

then

echo failed

exit 1

else

echo passed

exit 0

fi
```

BASH

BASH is an acronym for Bourne Again Shell, a punning name, which is a tribute to Bourne Shell (i.e., invented by Steven Bourne).

Bash is a shell program written by Brian Fox as an upgraded version of Bourne Shell program '**sh**'. It is an open source GNU project. It was released in 1989 as one of the most popular shell distribution of GNU/Linux operating systems. It provides functional improvements over Bourne Shell for both programming and interactive uses. It includes command line editing, key bindings, command history with unlimited size, etc.

In basic terms, Bash is a command line interpreter that typically runs in a text window where user can interpret commands to carry out various actions. The combination of these commands as a series within a file is known as a Shell Script. Bash can read and execute the commands from a Shell Script.

Bash is the default login shell for most Linux distributions

Shell : A unix shell is a program or command line interpreter that interprets the user command which are either entered the user directly or which can be read from a file and then pass them to the operating system for processing. It is important to note that

Shall scripts are interpreted and not compiled, as the computer system interprets them and there is not any need to compile Shell Scripts in order of execution.

There are different types of shells available in Linux Operating Systems. Some of which are as follows:

1. Bourne Shell
2. C shell
3. Korn Shell
4. GNU Bourne Shell

To know which shell types you operating system supports type the command into the terminal as given below

1.cat/etc/shells

And to know where bash is located in your os type the below command and you will get a specific location

1.which bash

1. Bash is **sh-compatible** as it derived from the original UNIX Bourne Shell. It is incorporated with the best and useful features of the Korn and C shell like directory manipulation, job control, aliases, etc.
2. Bash can be **invoked by** single-character command line options (**-a, -b, -c, -i, -l, -r, etc.**) as well as by multi-character command line options also like --debugger, --help, --login, etc.
3. Bash **Start-up files** are the scripts that Bash reads and executes when it starts. Each file has its specific use, and the collection of these files is used to help create an environment.
4. Bash consists of **Key bindings** by which one can set up customized editing key sequences.
5. Bash contains **one-dimensional arrays** using which you can easily reference and manipulate the lists of data.
6. Bash comprised of **Control Structures** like the **select construct** that specially used for menu generation.
7. Directory Stack in Bash specifies the history of recently-visited directories within a list. Example: **pushd** builtin is used to add the directory to the stack, **popd** is to remove directory from the stack and **dirs** builtin is to display content of the directory stack.
8. Bash also comprised of restricted mode for the environment security. A shell gets restricted if bash starts with name **rbash**, or the bash --restricted, or bash -r option passed at invocation.

Advantages of bash scripting

- **Automation**: Shell scripts allow you to automate repetitive tasks and processes, saving time and reducing the risk of errors that can occur with manual execution.
- **Portability**: Shell scripts can be run on various platforms and operating systems, including Unix, Linux, macOS, and even Windows through the use of emulators or virtual machines.
- **Flexibility**: Shell scripts are highly customizable and can be easily modified to suit specific requirements. They can also be combined with other programming languages or utilities to create more powerful scripts.
- **Accessibility**: Shell scripts are easy to write and don't require any special tools or software. They can be edited using any text editor, and most operating systems have a built-in shell interpreter.
- **Integration**: Shell scripts can be integrated with other tools and applications, such as databases, web servers, and cloud services, allowing for more complex automation and system management tasks.
- **Debugging**: Shell scripts are easy to debug, and most shells have built-in debugging and error-reporting tools that can help identify and fix issues quickly.

Bash script

Bash script is computer program written in bash programming language

How to create and run a Bash Script?

- To create an empty bash script, first, change the directory in which you want to save your script using **cd** command. Try to use text editor like ***gedit*** in which you want to type the shell commands.
- Use **touch** command to create the zero bytes sized script.

1. touch file_name

- To open the script in the text editor (eg., gedit), type

1. gedit file_name.sh

   Here, **.sh** is suffixed as an extension that you have to provide for execution.

- Type the shell commands for your bash script in the newly opened text window or the text editor. Before typing bash shell commands, first, look at the base of any bash script.

  Each Bash based Linux script starts by the line-

1. #! /bin/bash

   Where #! is referred to as the ***shebang*** and rest of the line is the path to the interpreter specifying the location of bash shell in our operating system.

   Bash use # to comment any line.

Bash use **echo** command to print the output.

At the end, execute the bash script prefixing with ./.