Linux is the OS of choice for server environment due to its stability and reliability

Linux based server could run nonstop without a reboot for year on end.

Each Linux distribution build for its own purpose to meet the demands of it target users.

An operating system based on the Linux kernel is called a Distribution or Distro.

There are hundreds of distributions available some of which are designed to accomplish a sole purpose like running servers, acts as network switches etc.

Naming the best Linux Distribution is difficult as they are made for different needs.

Files are store in root directory and its sub directorires. >> ROOT/

Directory files

Regular User >> Regular accounts are called standard account in Ubuntu desktop

Root User >> Super User >> Can access restricted files, install software and has administrative privileges

Services User >> Service account in Ubuntu Server edition

Unix/Linux uses a tree like hierarchical file system.

Peripherals like hard drives, cd rom, printer are also considered files in Linux/Unix.

Linux file naming convention is case sensitive.

For every user /home/<username> directory is created which is called his home directory.

Command line interface(CLI)- Terminal === Graphical User interface(GUI)- File Manager

1. Commands are flexible and offer more options.
2. Perform more with Just one command.
3. Work on multiple files at a time.
4. CLI is fast, Use less RAM

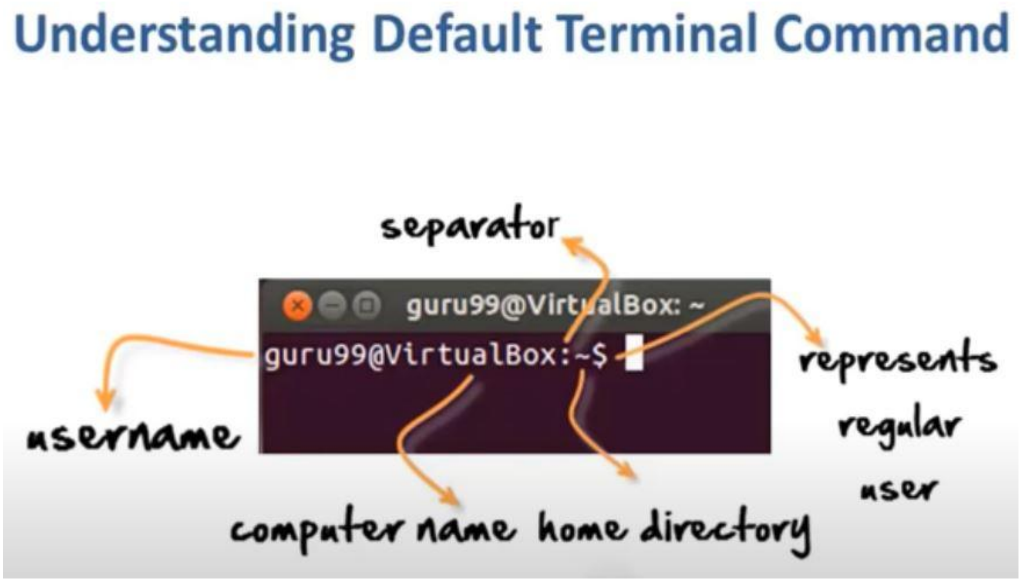
GUI importance >> Performance Graphs, Edit Images & Video, creating sketches, CAD, CAM = Graphics intensive tasks

Ctrl+Alt+T >> Terminal window also from dashboard.

cd / = for root dir

cd ~ = for home dir

cd directory1/directory2 >> navigating through multiple directories

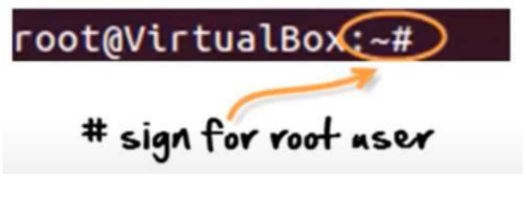


```
guru99@VirtualBox:~$
```

cd .. = moving up one directory level

A path in computing is the address for file and folder.

Absolute Path >> full path to reach a directory. Complete address of a file

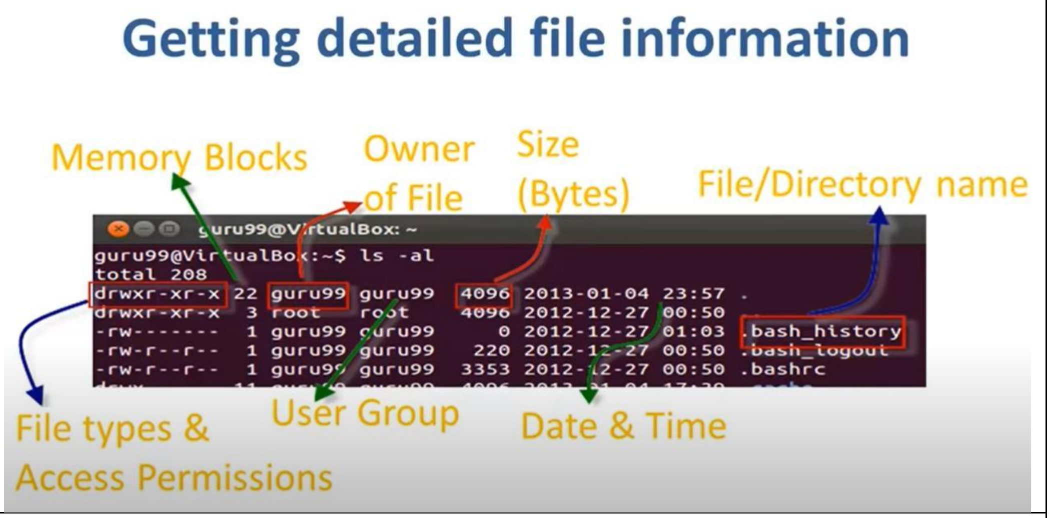


```
root@VirtualBox:~#
```

or directory.

Relative path >> just folder Name path if in the same directory = path to reach already open directory. Relative location of a file of directory with respect to current directory.

COMMANDS	DESCRIPTION
ls	Lists all files and directories in the present working directory.
ls -R	Lists files in sub-directories as well

ls -al	Lists files and directories with detailed information  Getting detailed file information
	Memory Blocks
	Owner of File
	Size (Bytes)
	File/Directory name
	File types & Access Permissions
	User Group
	Date & Time
	guru99@VirtualBox: ~ `guru99@VirtualBox:~$ ls -al`
	`total 208`
	`drwxr-xr-x 22 guru99 guru99  4096 2013-01-04 23:57 .`
	`drwxr-xr-x  3 root   root    4096 2012-12-27 00:50 ..`
	`-rw-------  1 guru99 guru99     0 2012-12-27 01:03 .bash_history`
	`-rw-r--r--  1 guru99 guru99   220 2012-12-27 00:50 .bash_logout`
	`-rw-r--r--  1 guru99 guru99  3353 2012-12-27 00:50 .bashrc`

ls -a	Lists hidden files start with '.' period symbol.
cat > filename	create & write = CTRL + D to exit from file = Crate a new file.
cat filename	read file = Display's file content
cat file1 file2 > file3	file3 = combine 2 files in 1 file = Join 2 files and stores output in a new file.
rm filename	Deletes a file
mv filename newfilelocation	Moving file to new location.
sudo mv filename newfilelocation	sudo = Allows regular users to run programs as superuser or root command contains password for 15 minutes per terminal.
mv filename Newfilename	renaming the file to new filename
mkdir directoryName	Creating new directory in the present working directory
mkdir path/dirName	Creating New directory at the specified path location
mkdir dir1 dir2 dir3	Creating Multiple directories
rmdir directoryName	Removing or Deletes a directory
mv directory Newdirectory	Renames a directory
man man <command>	manual similar to Helpfile >> reference book for Linux = Gives help information on a command
history	Gives list of all past commands typed in current terminal session
clear	clears the terminal

paste on terminal >> Ctrl+Shift+V or Shift+Insert or Edit-Paste option

Authorization Levels

1. Ownership
2. Permission

Permission system in linux >> User, Group, ALL >> Read Write Execute

r = read, w = write, x = execute, - =no permission

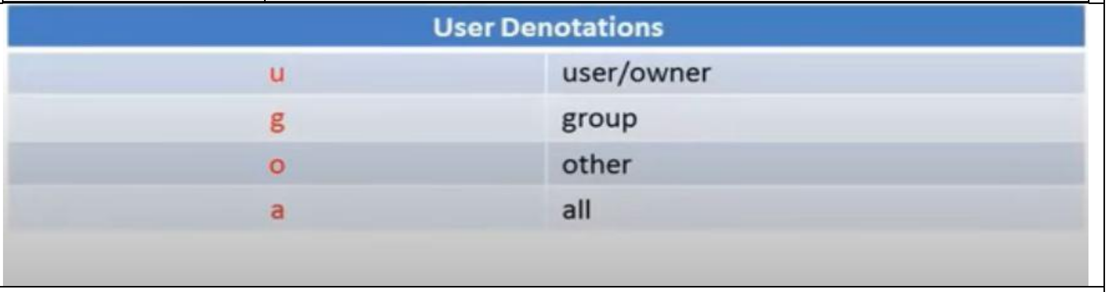
chmod	changing file/ directory permission >> permission on a file can be change which can be further divided into Absolute and Symbolic mode. changing ownership and group

Absolute (Numberic) Mode

Number	Permission Type	Symbol
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute + Write	-wx
4	Read	r--
5	Read + Execute	r-x
6	Read + Write	rw-
7	Read + Write + Execute	rwx

Symbolic Mode

Operator	Description
+	Adds a permission to a file or directory
-	Removes the permission
=	Sets the permission and overrides the permissions set earlier



User Denotations	
u	user/owner
g	group
o	other
a	all

chown user <filename>	can change the ownership of a file/directory chown user:group <filename>
chgrp group filename	can change the group ownership
	/etc/group = all group users
groups	
newgrp	

2 groups can not own the same file.

Linux being a multi-user system uses permission and ownerships for security.

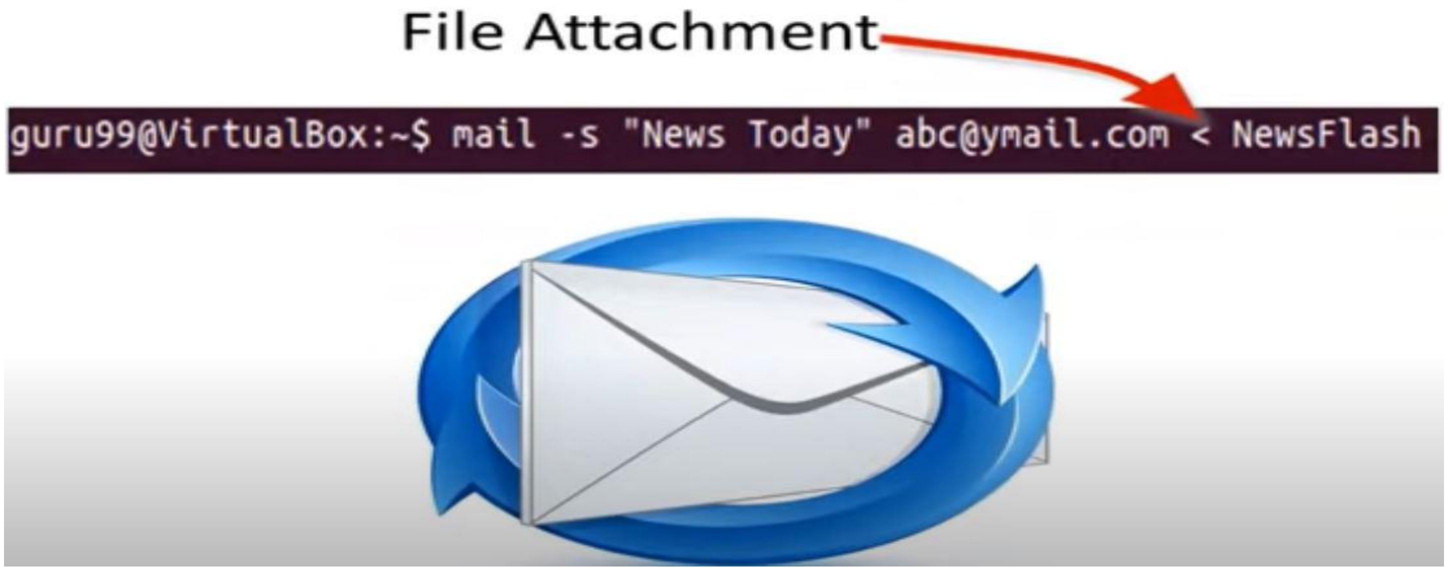
pr	print command can format and print directly from the terminal. The formatting you do on the file does not affect the file contents.

Option	Function
-x	Divides the data into 'x' columns
-h "header"	Assign "header" value as the report header
-t	Does not print the header and top/bottom margins
-d	Double spaces the output file
-n	Denotes all line with numbers
-l page length	Defines the lines (page length) in a page. Default is 56
-o margin	Formats the page in accordance with the margin number

pr -x	Divides the file into x columns
pr -h	Assigns a header to the file
pr -n	Denotes the file with Line Numbers
lp -nc <FileName> lpr c <FileName> Prints	"c" copies of the file.

lp -d<printername> < FileName > lp -P<printername> < FileName >	Specifies name of the printer
apt-get	Command used to install and update packages sudo apt-get install software name
Sudo apt-get install mailx	Install package mail
mailx address body	to send the mail

Unix/Linux software is installed in form of packages. A package contains the program itself. Any dependent component needs to be downloaded separately.



```
guru99@VirtualBox:~$ mail -s "News Today" abc@ymail.com < NewsFlash
```

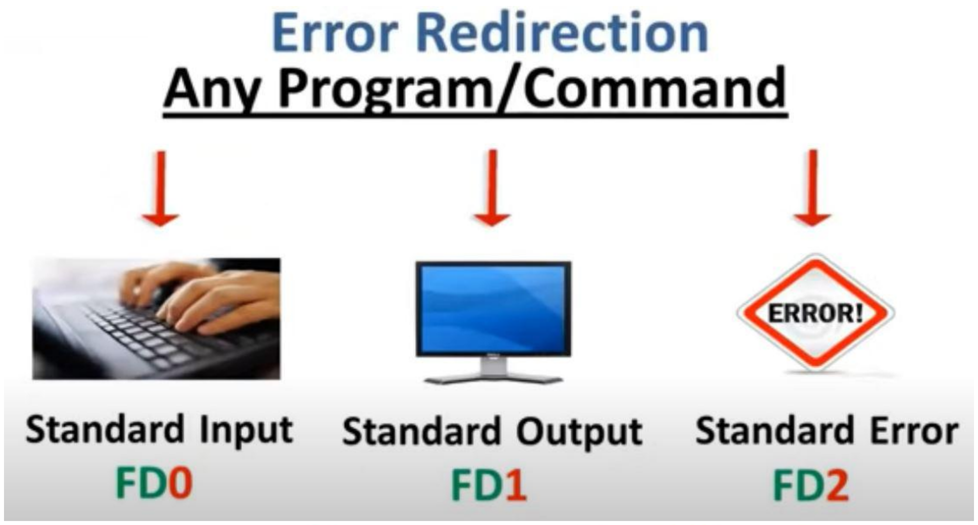
ls -al > listings

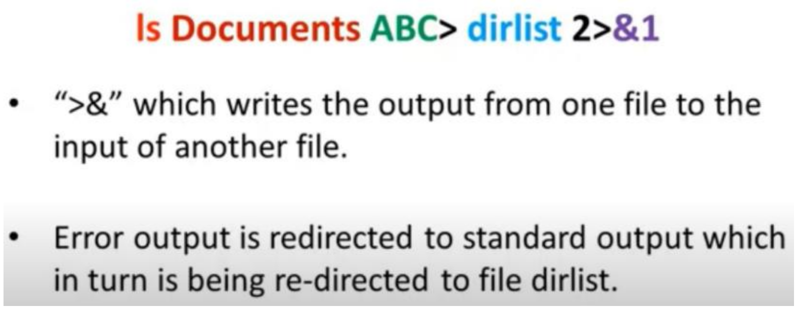
Every file has a number called FD

Each file in Linux has a corresponding file descriptor associated with it.

Error redirection

Ex. telnet localhost 2> errorfile





ls Documents ABC> dirlist 2>&1

- “>&” which writes the output from one file to the input of another file.
- Error output is redirected to standard output which in turn is being re-directed to file dirlist.

" > " is the output redirection operator.

" >> " appends output to an existing file.

" < " is the input redirection operator.

" >& " redirects output of one file to another.

Can re-direct error using its corresponding File Discripter2.

'|' denotes a PIPE >> to run two commands consecutively. Helps in creating powerful commands. Ex. cat filename | less , cat filename | more, cat filename | pg

A filter in a pipe is an output of one command which serves as input to the next. less, pg and more commands are used for dividing a long file into readable bits.

grep = Scan a document & Present the result in a format you want.

grep <search_string> = cat filename | grep wordSearch >> command can be used to find strings and values in text document.

-v	Shows all the lines that do not match the searched string
-c	Display only the count of matching lines
-n	Shows the matching line and its number
-i	Match both (upper & lower) case
-l	Shows just the name of the file with the string

sort filename = command sorts out the content of a file alphabetically.

-r	Reverse's sorting
-n	Sort's numerically
-f	Case insensitive sorting

Ex. Cat file1 | grep -v a | sort -r

Regular Expressions >> are set of character used to check patterns in strings. 'regexp' & 'regex'

Symbol	Descriptions	Example
.	Replaces any character	
^	Matches start of string	cat file1 | grep ^a
$	Matches end of string	cat file1 | grep t$
*	Matches up zero or more times the preceding character	
\	Represent special character	
()	Matches up exactly one character	
?		

Interval Regular Expressions >> Number of occurrences of a character in a string.

{n}	Matches the preceding character appearing 'n' times exactly
{n,m}	Matches the preceding character appearing 'n' times but not more than m
{n, }	Matches the preceding character only when its appears 'n' times or more

Extended Regular Expressions >>

\+	Matches one or more occurrence of the previous character
\?	Matches zero or more occurnace of the previous character

Brace expansion >> is used to generate strings. It helps in creating multiple strings out of one.   echo {a..z} , echo {1..11} , echo a{0..9}b

Environment variables govern behavior of programs in your Operating system.

Variable is location for storing a value.

PATH	This variable contains a colon : - separated list of directories in which your system looks for executable files.
USER	The username
HOME	Default path to the user's home directory

EDITOR	Path to the program which edits the content of files
UID	User’s unique ID
TERM	Deafult terminal emulator
SHELL	Shell being used by the user
ENV	Display all the environment varibales

echo $VARIABLE	TO display value of a variable	echo $PATH
env	Display all environmental variables	
VARIABLE_NAME=variable_value	Create a new variable	Newvar
Unset VARIABLENAME	Remove a variable	
export Variable=value	To set value of an environment variable	

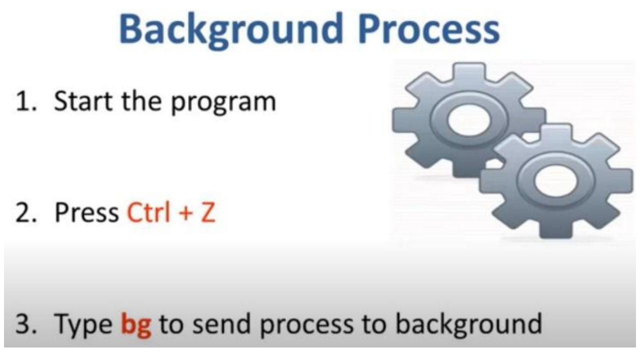
Communication in Linux

ping<ip-address or hostname>	Analyzing network and host connections. Tracking network performance and managing it. Testing hardware and software issues.
ftp<ip-address or hostname>	preferred protocol for sending and receiving large file. Logging in and establishing a connection with a remote host. Upload and download files. Navigating through directories. Browsing content of the directories.

dir	Display files in the current directory of remote computer
cd “dirname”	Change directory to “dirname” on remote computer
put file	Upload ‘file’ from local to remote computer
get file	Download ‘file’ from to local computer
quit	Logout

telnet<ip-address or hostname>	Connect to a remote Linux computer. Run programs remotely and conduct administration. Similar like Remote desktop found in windows machine.
SSH username@ip-address or hostname	SSH is a replacement for telnet and is used by system administrators to control remote Linux servers.

Managing Processes >> An instance of a program is called a Process. Any command that you give to your Linux machine start anew process. Any running program or a command given to a Linux system is called Process.



Background Process

1. Start the program
2. Press Ctrl + Z
3. Type **bg** to send process to background

fg<jobname>	Foreground processes
top	display all the running processes
ps ux == ps PID	PS utility pidof<processname> = PID
kill PID	Kill utility Terminating running processes
df or df -h	DF utility = Reports the free disk space
free -m free -g	Free = Shows free and uesd memeory (RAM) on the Linux system.

PID Status >> D = Uninterruptible sleep R=Running S=Sleeping T=Traced or Stopped Z-Zombie

Priority index of a process is called Nice in Linux.

Niceness 20 to -19 = Lower the niceness index, Higher would be the priority given to that Task.Default value of all the processes is 0 and it can vary between 20 to -19.

nice -n 'nice value' process name	Starts a process with given priority
sudo renice 'nice value' -p 'PID'	Change priority of an already running process

bg	To send a process to background
fg	To run a stopped process in foreground
top	Details on all Active processes
ps	Guve the status of processes running for a user
ps PID	Gives the status of a particular process
Pidof<processname>	Gives the process ID (PID) of a process
Kill PID	Kills a process
df	Gives free hard disk space on your system
free	Tells the free RAM on your system

vi Editor Insert mode:

- This mode is for inserting text in the file.
- You can switch to the Insert mode from the command mode **by pressing 'i' on the keyboard**
- Once you are in Insert mode, any key would be taken as an input for the file on which you are currently working.
- To return to the command mode and save the changes you have made you need to press the Esc key

vi <filenameNEW> or vi <filenameExisting>

VI Editing commands

- i - Insert at cursor (goes into insert mode)
- a - Write after cursor (goes into insert mode)
- A - Write at the end of line (goes into insert mode)
- ESC - Terminate insert mode
- u - Undo last change
- U - Undo all changes to the entire line
- o - Open a new line (goes into insert mode)
- dd - Delete line
- 3dd - Delete 3 lines.
- D - Delete contents of line after the cursor
- C - Delete contents of a line after the cursor and insert new text. Press ESC key to end insertion.
- dw - Delete word
- 4dw - Delete 4 words
- cw - Change word
- x - Delete character at the cursor
- r - Replace character
- R - Overwrite characters from cursor onward
- s - Substitute one character under cursor continue to insert
- S - Substitute entire line and begin to insert at the beginning of the line
- ~ - Change case of individual character

Moving within a file

- k - Move cursor up
- j - Move cursor down
- h - Move cursor left
- l - Move cursor right

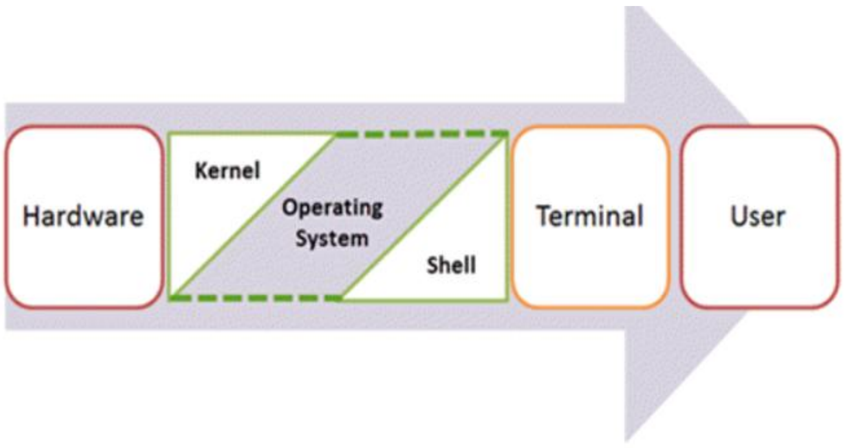
Saving and Closing the file

- Shift+zz - Save the file and quit
- :w - Save the file but keep it open
- :q - Quit without saving
- :wq - Save the file and quit

Shell Scripting >>>>>

$ sign is shell in terminal.

The Shell wraps around the delicate interior of an Operating system protecting it from accidental damage. Hence the name **Shell**.



Components of Shell Program

Writing a series of commands, combine lengthy and repetitive commands

There are two main shells in Linux:

1. The **Bourne Shell**: The prompt for this shell is $ and its derivatives are listed below:

- POSIX shell also is known as sh
- Korn Shell also knew as sh
- **B**ourne **A**gain **SH**ell also knew as bash (most popular)

2. The C shell: The prompt for this shell is %, and its subcategories are:

- C shell also is known as csh
- Tops C shell also is known as tcsh

steps in creating a Shell Script:

1. **Create a file using** a **vi** editor(or any other editor). Name script file with **extension .sh**
2. **Start** the script with **#! /bin/sh**

3. Save the script file as filename.sh
4. For **executing** the script type **bash filename.sh**

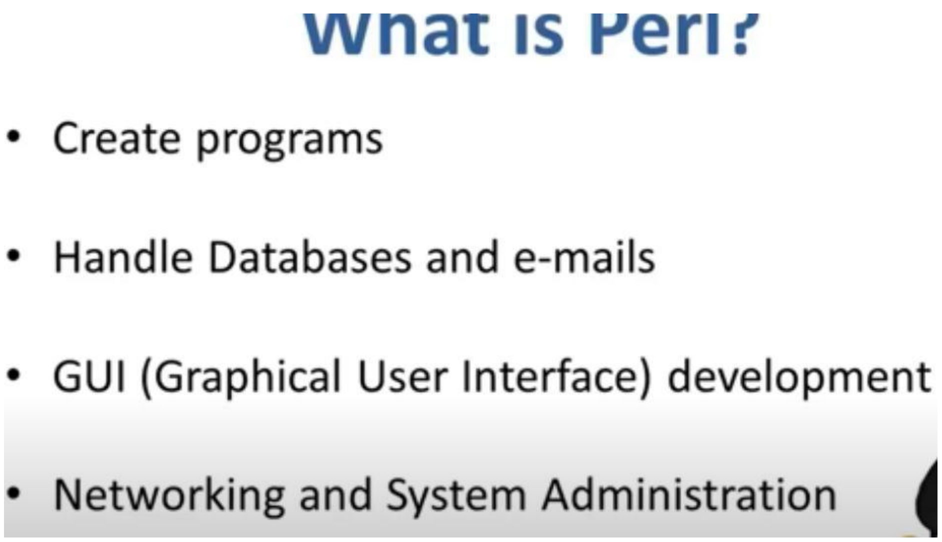
"#!" is an operator called shebang which directs the script to the interpreter location. So, if we use"#! /bin/sh" the script gets directed to the bourne-shell.

Perl

#!/usr/bin/perl .pl extension

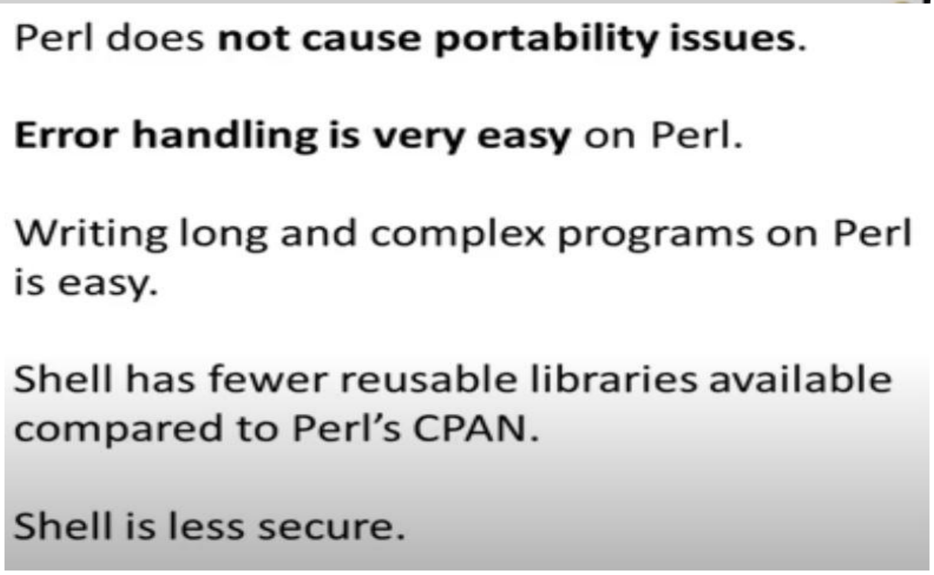
every statement in Perl ends with a semi-colon.

Do not use space while naming the Perl script file.



What is Perl?

- Create programs
- Handle Databases and e-mails
- GUI (Graphical User Interface) development
- Networking and System Administration



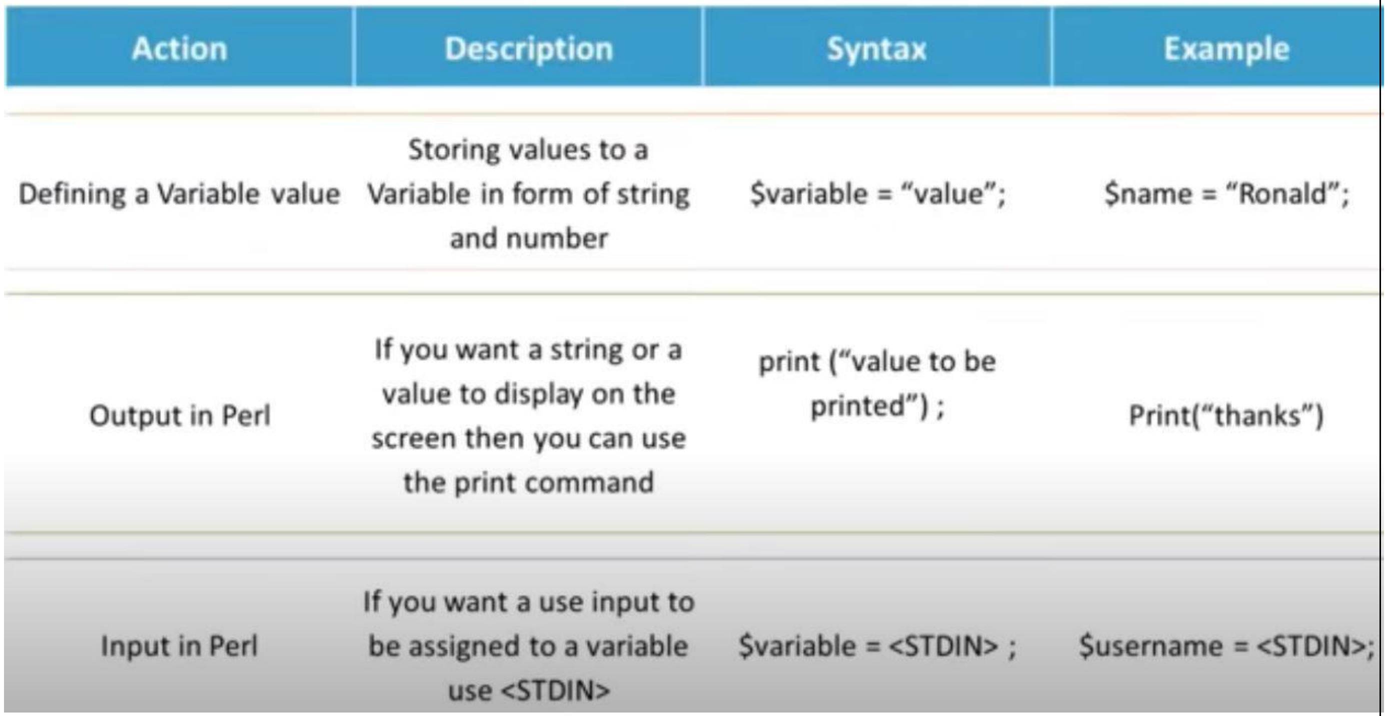
Perl does **not cause portability issues.**

Error handling is very easy on Perl.

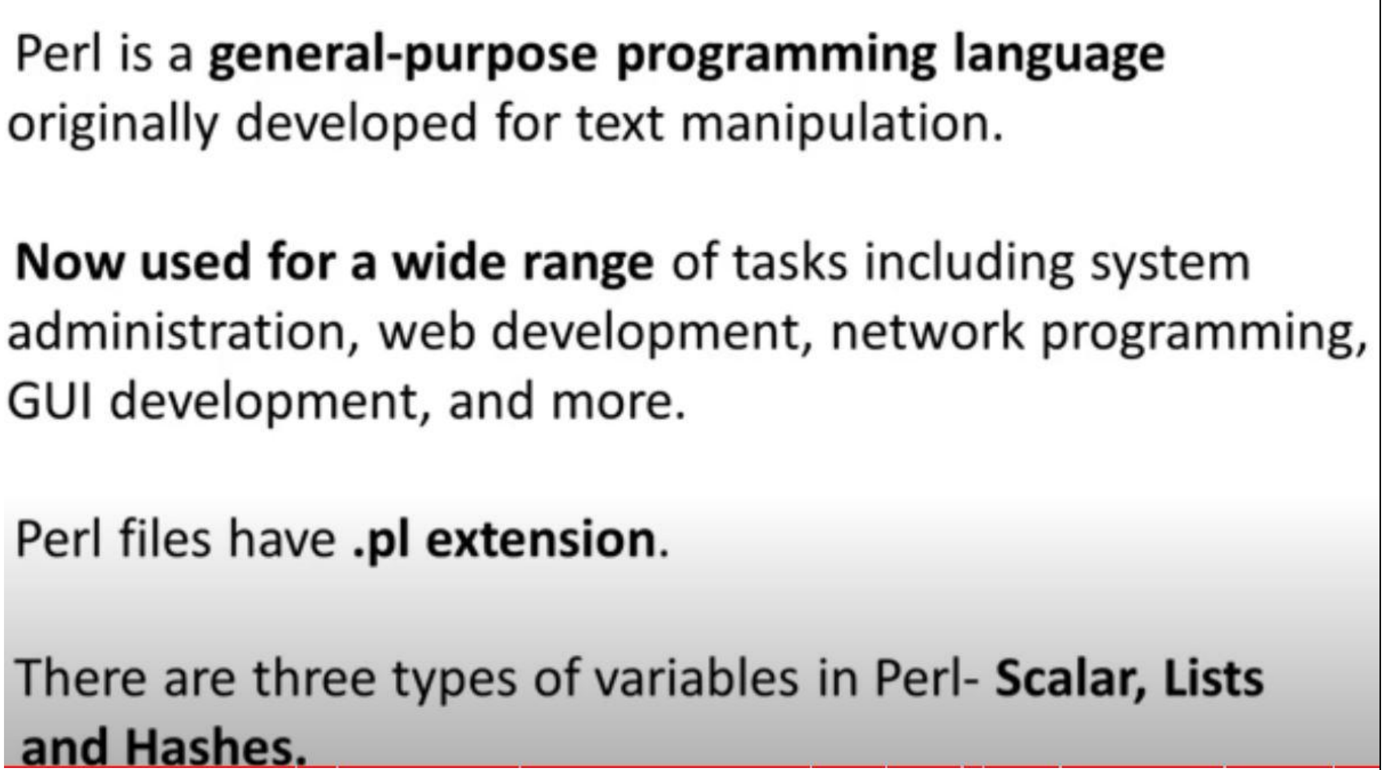
Writing long and complex programs on Perl is easy.

Shell has fewer reusable libraries available compared to Perl’s CPAN.

Shell is less secure.



Action	Description	Syntax	Example
Defining a Variable value	Storing values to a Variable in form of string and number	$variable = “value”;	$name = “Ronald”;
Output in Perl	If you want a string or a value to display on the screen then you can use the print command	print (“value to be printed”) ;	Print(“thanks”)
Input in Perl	If you want a use input to be assigned to a variable use <STDIN>	$variable = <STDIN> ;	$username = <STDIN>;



Perl is a **general-purpose programming language** originally developed for text manipulation.

Now used for a wide range of tasks including system administration, web development, network programming, GUI development, and more.

Perl files have **.pl extension**.

There are three types of variables in Perl- **Scalar, Lists and Hashes.**

Virtual Terminal used for executing commands and offering input, cannot use the mouse with the virtual terminals.

Virtual Terminal = Ctrl+Alt+F1 then Enter User ID and Password. 6 Virtual Terminal for different Users to conduct different tasks.

Navigating different terminal using Crtl+Alt+F(1 to 6) Key

The seventh terminal is the one which we have been using so for in Linux tutorials. It can be accessed by pressing the below given key combination. Ctrl + Alt + F7

Shortcut	Function
Home or Ctrl + a	Move the cursor to the start of the current line
End or Ctrl + e	Move the cursor to the end of the current line
Tab	Autocomplete commands
Ctrl + u	Erase the current line
Ctrl + w	Delete the word before the cursor
Ctrl + k	Delete the line from the cursor position to the end
reset	Reset the terminal
history	List of commands executed by the user
Arrow up	Scroll up in history and enter to execute
Arrow down	Scroll down in history and enter to execute
Ctrl + d	Logout from the terminal
Ctrl + Alt + Del	Reboot the system

- Virtual terminals are CLIs which execute the user commands
- There are six virtual terminals which can be launched using the shortcut keys
- They offer multi-user environment, and up to six users can work on them at the same time
- Unlike terminals, you cannot use mouse with virtual terminals
- To launch a virtual terminal press Ctrl+Alt+F(1 to 6) on the keyboard
- Use the same command for navigating through the different terminals
- To return to the home screen of the Linux system, use Ctrl+Alt+F7 and it would take to you the terminal

Unix/Linux Administration

Linux/Unix user management commands

User management in Linux is done by using Linux administration commands. Here is a list of user management commands in Linux:

Command	Description
sudo adduser username	Adds a user
sudo passwd -l 'username'	Disable a user
sudo userdel -r 'username'	Delete a user
sudo usermod -a -G GROUPNAME USERNAME	Add user a to a usergroup
sudo deluser USER GROUPNAME	Remove user from a user group
finger	Gives information on all logged in user
finger username	Gives information of a particular user

grep Regular Expression Operator

I hope following table will help you quickly understand regular expressions in grep when using under Linux or Unix-like systems:

grep regex operator	Meaning	Example
`.`	Matches any single character.	`grep '.' file` `grep 'foo.' input`
`?`	The preceding item is optional and will be matched, at most, once.	`grep 'vivek?'` `/etc/passwd`
`*`	The preceding item will be matched zero or more times.	`grep 'vivek*'` `/etc/passwd`
`+`	The preceding item will be matched one or more times.	`ls /var/log/ | grep` `-E "^[a-z]+\.log."`
`{N}`	The preceding item is matched exactly N times.	`egrep '[0-9]{2}` `input`
`{N,}`	The preceding item is matched N or more times.	`egrep '[0-9]{2,}` `input`
`{N,M}`	The preceding item is matched at least N	`egrep '[0-9]{2,4}`

	times, but not more than M times.	`input`
`-`	Represents the range if it's not first or last in a list or the ending point of a range in a list.	`grep ':/bin/[a-z]*'` `/etc/passwd`
`^`	Matches the empty string at the beginning of a line; also represents the characters not in the range of a list.	`grep '^vivek'` `/etc/passwd` `grep '[^0-9]*'` `/etc/passwd`
`$`	Matches the empty string at the end of a line.	`grep '^$'` `/etc/passwd`
`\b`	Matches the empty string at the edge of a word.	`vivek '\bvivek'` `/etc/passwd`
`\B`	Matches the empty string provided it's not at the edge of a word.	`grep '\B/bin/bash` `/etc/passwd`
`\<`	Match the empty string at the beginning of word.	`grep '\`
`\>`	Match the empty string at the end of word.	`grep 'bash\>'` `/etc/passwd` `grep '\' /etc/passwd`

TUTORIAL >> A Basic MySQL Tutorial >> https://www.digitalocean.com/community/tutorials/a-basic-mysql-tutorial

---

Edureka Shell Script = https://youtu.be/GtovwKDemnI

Linux can be customized it according to the nature of your work which brings to access to source code. Tweaks in the code which suits needs.

The computer programs that allocate the system resources and co-ordinate all the details of the computer's internals is called the operating system or the kernel. Users communicate with the OS through a program called the Shell.

CLI is a text-based interface used to interact with software and operating system by typing commands into the interface and receive a response in the same way.

#!/bin/sh >> The Shebang = # symbol is called a hash and ! symbol is called a bang.

The Shell is a Command Line Interpreter.It translates commands entered by the user and converts them into a language that is understood by the Kernel.

Shell script is a list of commands, which are listed in the order of execution.

ls /bin/ = specify the path	BOURNE SHELL TYPES	C SHELL TYPES
ls -l	Bourne shell(sh)	C shell(csh)
less <file.txt>	Korn shell(ksh)	TENEX/TOPS C shell(tcsh)
mv -v	Bourne-Ahain shell(bash)	Z shell
man +help	POSIX shell(sh)	

```
#!/bin/sh
#Script is as follow

echo "What is your Name?"
read PERSON
echo "Hello, $PERSON"
```

A variable is a character string to which we assign a value.The value assigned could be a number, text, filename, device, or any other typr of data.

A local variable is a variable that is present within the current instance of the shell.It is not available to programs that are started by the the shell. They are ste at the command prompt.

An environment variable is available to any child process of the shell.Some programs need environment variable in order to function correctly.

A shell variable is a special variable that is set by the shell and is required by the shell in order to function correctly.Some of these avriables are environmet variable whereasothers are local variables.

Defining Variable

VariableName="VariableValue" >> Scalar variables only hold single value.

#!/bin/sh #VariableName="VariableValue" NAME="ShellScript" echo $NAME	#!/bin/sh #VariableName="VariableValue" NAME="ShellScript" readonly NAME NAME="UPSET"	#!/bin/sh #VariableName="VariableValue" unset NAME echo $NAME

Special varibale correspods to argument with which script was invok.$# $* $@ $? $$ $

filename.sh Learning HAppy >>>>test the below scripts

#!/bin/sh
 	
echo "File name: $0"
echo "First parameter: $1"
echo "Second parameter: $2"
echo "Quoted Values: $@"
echo "Quoted Values: $*"
echo "No of parameter: $#"
	#!/bin/sh
	
	for TOKEN in $*
	do
	    echo $TOKEN
	done
		#!/bin/sh
		
		for TOKEN in $*
		do
		    echo $?
		done

Basic OPERATORS >> Arithmetic, Relational, Boolean, String, File Test

Shell Loops >> while, For, Until, Nested, Loop Control

```
#!/bin/sh

for var in 0 1 2 3 4 5 6 7 8 9
do
        echo $var
done
```

```
#!/bin/sh

a=0
while [ $a -lt 10 ]
do
        echo $a
        a=expr $a + 1
done
#Statement executed while condition is
true.
```

```
#!/bin/sh

a=0
until [ ! $a -lt 10 ]
do
        echo $a
        a=expr $a + 1
done
#Statement keeps executed until
condition is True.
```

```
#!/bin/sh

#nested
a=0
while [ "$a" -lt 10 ]      #loop1
do
  b="$a"
  while [ "$b" -ge 0 ]  #loop2
  do
        echo -n "$b"
        b=expr $b + 1
  done
  echo
  a='expr $a = 1'
done
```

```
#!/bin/sh

#infinite loop
a=0
until [ ! $a -ge 0 ]
do
        echo $a
        a=expr $a + 1
done
#Statement executed while condition is
true.
```

```
#!/bin/sh

#break
a=0

while [ $a -lt 10 ]
do
        echo $a
        if [ $a -eq 5 ]
        then
                break
        fi
        a='expr $a + 1'
done
```

```
#!/bin/sh

NUM="1 2 3 4 5 6 7"

for NUM in $NUMS
do
        Q='expr $NUM % 2'
        if [ $Q -eq 0 ]
        then
           echo "Number is an even number!!"
           continue
        fi
        echo "Found odd nmber"
done
```

Shell Functions >>> Creating Functions, Passing Parameters to function, Returning values from functions, Nested Functions, Function call from Prompt.

```
#!/bin/sh
#Define fuction
Hello(){
	echo "Hello $1 $2 "
}

#Invoke Functio
Hello Learning Path
```

```
#!/bin/sh

#USE CASE
for i in $@
do
ping -c l $i &> /dev/null

if [ $? -ne 0 ];then
	echo "'date': ping failed, $i host is down!" | mail -s "$i host is down!"
test1@yaoo.com
fi
done
```

fileneme.sh google.com yahoo.com >>> testing

https://www.tutorialspoint.com/unix/index.htm

https://www.nagios.org/documentation/

	df -Th	
	free -mh	
	top	
	ps -ef | grep <AppName>	
	systemctl status nginx.services	
	systemctl status redis.services	
	systemctl status cups.services	
	ls -ltr /log/ | tail -5	