**4.Shell Scripting**

**1.Write a Shell program to check the given number is even or odd.**

**Source Code:**

#!/bin/bash

#author:sreekanth pradeep

#roll no: 52

echo "enter a number"

read num

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

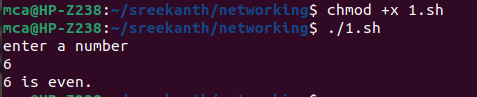
echo "$num is even."

else

echo "$num is odd."

fi

**Output:**



**2.Write a Shell program to check a leap year.**

**Source Code:**

#!/bin/bash

#author:sreekanth pradeep

#roll no: 52

echo "enter a year:"

read year

if [ $((year % 4)) -eq 0 ] && [ $((year % 100)) -ne 0 ] || [ $((year % 400)) -eq 0 ]; then

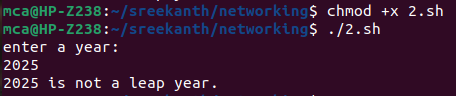
echo "$year is a leap year."

else

echo "$year is not a leap year."

fi

**Output:**

****

**3. Write a Shell program to find the area and circumference of a circle.**

**Source Code:**

#!/bin/bash

#author:sreekanth

#rool no:52

echo "enter the radius of circle:"

read r

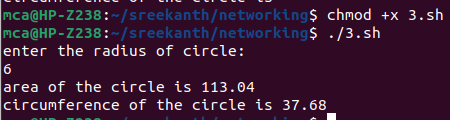
area=$(echo "3.14\*$r\*$r" | bc)

circum=$(echo "3.14\*2\*$r" | bc)

echo "area of the circle is" $area

echo "circumference of the circle is" $circum

**Output:**

****

**4. Write a Shell program to check the given number and its reverse are same.**

**Source Code:**

#!/bin/bash

echo "enter a number:"

read number

reverse=$(echo $number | rev)

if [ $number -eq $reverse ]; then

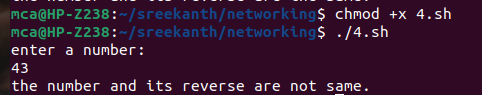
echo "the number and its reverse are the same."

else

echo "the number and its reverse are not same."

fi

**Output:**

****

**5.Write a Shell program to check the given string is palindrome or not.**

**Source Code:**

#!/bin/bash

echo "Enter a string:"

read string

reverse=$(echo $string | rev)

if [ "$string" == "$reverse" ]; then

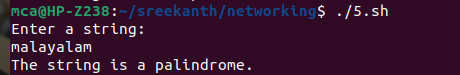
echo "The string is a palindrome."

else

echo "The string is not a palindrome."

fi

**Output:**

****

**6.Write a Shell program to find the sum of odd and even numbers from a set of numbers.**

**Source Code:**

#!/bin/bash

echo "Enter numbers separated by space:"

read -a numbers

sum\_even=0

sum\_odd=0

for num in "${numbers[@]}"; do

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

sum\_even=$((sum\_even + num))

else

sum\_odd=$((sum\_odd + num))

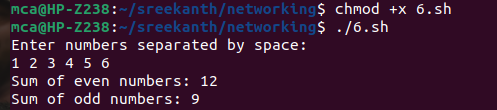
fi

done

echo "Sum of even numbers: $sum\_even"

echo "Sum of odd numbers: $sum\_odd"

**Output:**

****

**7.Write a Shell program to find the roots of a quadratic equation.**

**Source Code:**

#!/bin/bash

echo "Enter the coefficients (a, b, c) of the quadratic equation (ax^2 + bx + c):"

read a b c

discriminant=$((b \* b - 4 \* a \* c))

if [ $discriminant -gt 0 ]; then

root1=$(echo "(-$b + sqrt($discriminant)) / (2 \* $a)" | bc -l)

root2=$(echo "(-$b - sqrt($discriminant)) / (2 \* $a)" | bc -l)

echo "The roots are real and different."

echo "Root 1 = $root1, Root 2 = $root2"

elif [ $discriminant -eq 0 ]; then

root=$(echo "-$b / (2 \* $a)" | bc -l)

echo "The roots are real and equal."

echo "Root 1 = Root 2 = $root"

else

real\_part=$(echo "-$b / (2 \* $a)" | bc -l)

imaginary\_part=$(echo "sqrt($((-1 \* discriminant))) / (2 \* $a)" | bc -l)

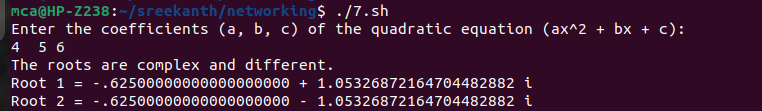
echo "The roots are complex and different."

echo "Root 1 = $real\_part + $imaginary\_part i"

echo "Root 2 = $real\_part - $imaginary\_part i"

fi

**Output:**

****

**8. Write a Shell program to check the given integer is Armstrong number or not.**

**Source Code:**

#!/bin/bash

echo "Enter a number:"

read number

length=${#number}

sum=0

for ((i=0; i<$length; i++)); do

digit=${number:i:1}

sum=$((sum + digit \*\* length))

done

if [ $sum -eq $number ]; then

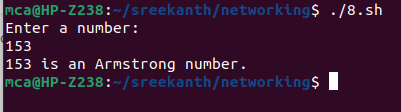
echo "$number is an Armstrong number."

else

echo "$number is not an Armstrong number."

fi

**Output:**

****

**9.Write a Shell program to check the given integer is prime or not.**

**Source Code:**

#!/bin/bash

echo "Enter a number:"

read number

is\_prime=true

if [ $number -lt 2 ]; then

is\_prime=false

fi

for ((i=2; i<=number/2; i++)); do

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

is\_prime=false

break

fi

done

if $is\_prime; then

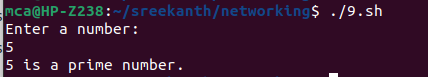
echo "$number is a prime number."

else

echo "$number is not a prime number."

fi

**Output:**

****

**10.Write a Shell program to generate prime numbers between 1 and 50.**

**Source Code:**

#!/bin/bash

echo "Prime numbers between 1 and 50 are:"

for ((i=2; i<=50; i++)); do

is\_prime=true

for ((j=2; j<=i/2; j++)); do

if [ $((i % j)) -eq 0 ]; then

is\_prime=false

break

fi

done

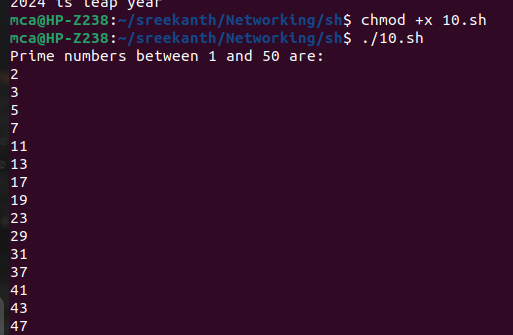
if $is\_prime; then

echo $i

fi

done

**Output:**

****

**11.Write a Shell program to find the sum of square of individual digits of a number.**

**Source Code:**

#!/bin/bash

echo "Enter a number:"

read num

sum=0

while [ $num -gt 0 ]; do

digit=$(( $num % 10 ))

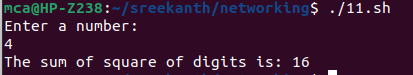
sum=$(( $sum + $digit \* $digit ))

num=$(( $num / 10 ))

done

echo "The sum of square of digits is: $sum"

**Output:**

****

**12.Write a Shell program to count the number of vowels in a line of text.**

**Source Code:**

#!/bin/bash

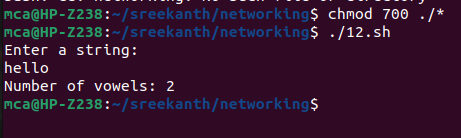
echo "Enter a string:"

read str

count=$(echo $str | grep -o -i "[aeiou]" | wc -l)

echo "Number of vowels: $count"

**Output:**

****

**13.Write a Shell program to display student grades.**

**Source Code:**

#!/bin/bash

calculate\_grade() {

if [ $1 -ge 90 ]; then

grade="A"

elif [ $1 -ge 80 ]; then

grade="B"

elif [ $1 -ge 70 ]; then

grade="C"

elif [ $1 -ge 60 ]; then

grade="D"

else

grade="F"

fi

echo $grade

}

echo "Enter student name:"

read name

echo "Enter student's mark:"

read mark

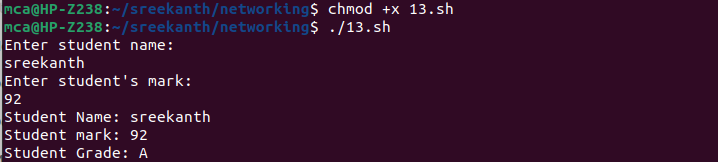
grade=$(calculate\_grade $mark)

echo "Student Name: $name"

echo "Student mark: $mark"

echo "Student Grade: $grade"

**Output:**

****

**14.Write a Shell program to find the smallest and largest numbers from a set of numbers.**

**Source code:**

#!/bin/bash

echo "Enter numbers separated by space:"

read -a numbers

largest=${numbers[0]}

smallest=${numbers[0]}

for num in "${numbers[@]}"; do

if [ $num -gt $largest ]; then

largest=$num

fi

if [ $num -lt $smallest ]; then

smallest=$num

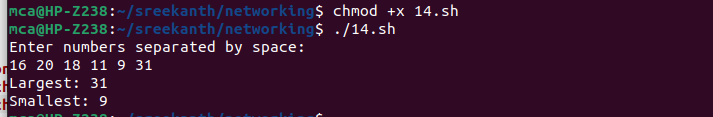
fi

done

echo "Largest: $largest"

echo "Smallest: $smallest"

**Output:**

****

**15.Write a Shell program to find the smallest digit from a number.**

**Source Code:**

#!/bin/bash

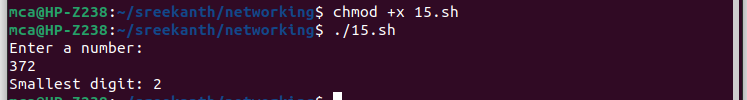
echo "Enter a number:"

read num

smallest=$(echo $num | grep -o "[0-9]" | sort | head -n1)

echo "Smallest digit: $smallest"

**Output:**

****

**16.Write a Shell program to find the sum of all numbers between 50 and 100, which are divisible by 3 and not divisible by 5.**

**Source Code:**

#!/bin/bash

sum=0

for ((i=50; i<=100; i++)); do

if [ $((i % 3)) -eq 0 ] && [ $((i % 5)) -ne 0 ]; then

sum=$((sum + i))

fi

done

echo "Sum of numbers divisible by 3 and not by 5 between 50 and 100: $sum"

**Output:**

**17.Write a Shell program to find the second highest number from a set of numbers.**

**Source Code:**

#!/bin/bash

echo "Enter numbers separated by space:"

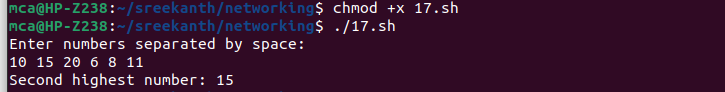
read -a numbers

IFS=$'\n' sorted=($(sort -n <<<"${numbers[\*]}"))

len=${#sorted[@]}

echo "Second highest number: ${sorted[len-2]}"

**Output:**

****

**18.Write a Shell program to find the sum of digits of a number using function.**

**Source Code:**

#!/bin/bash

echo "Enter a number:"

read num

sum\_digits() {

local n=$1

local sum=0

while [ $n -gt 0 ]; do

sum=$((sum + n % 10))

n=$((n / 10))

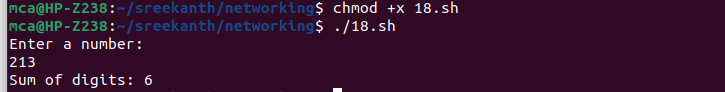
done

echo $sum

}

echo "Sum of digits: $(sum\_digits $num)"

**Output:**

****

**19.Write a Shell program to print the reverse of a number using function.**

**Source Code:**

#!/bin/bash

echo "Enter a number:"

read num

reverse() {

local n=$1

local rev=0

while [ $n -gt 0 ]; do

remainder=$((n % 10))

rev=$((rev \* 10 + remainder))

n=$((n / 10))

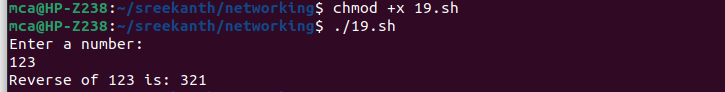
done

echo $rev

}

echo "Reverse of $num is: $(reverse $num)"

**Output:**

****

**20.Write a Shell program to find the factorial of a number using for loop.**

**Source Code:**

#!/bin/bash

echo "Enter a number:"

read num

fact(){

fact=1

for((i=1; i<=num; i++)); do

fact=$((fact \* i))

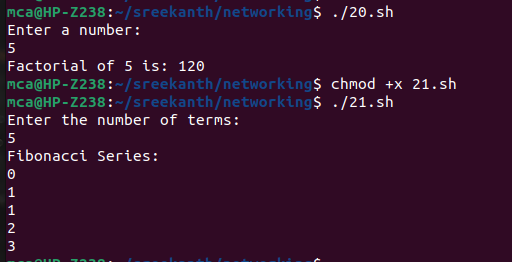
done

echo $fact

}

echo "Factorial of $num is: $(fact)"

**Output:**

****

**21.Write a Shell program to generate Fibonacci series.**

**Source Code:**

#!/bin/bash

echo "Enter the number of terms:"

read n

a=0

b=1

echo "Fibonacci Series:"

for ((i=0; i<n; i++)); do

echo "$a"

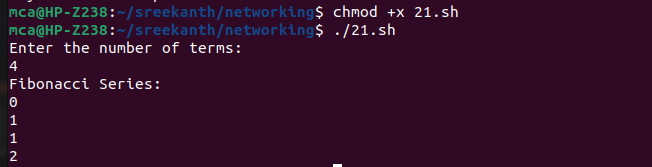
fn=$((a + b))

a=$b

b=$fn

done

**Output:**

****

**22) Write a shell script, which receives two filenames as arguments. It checks whether the two files contents are same or not. If they are same then second file is deleted.**

**Source Code:**

#!/bin/bash

if [ $# -ne 2 ]; then

echo "Try : $0 <file1> <file2>"

exit 0

fi

if cmp -s $1 $2; then

echo "Files are the same. Deleting $2"

rm $2

else

echo "Files are different"

fi

**Output:**

****

**23) Write a Menu driven Shell script that Lists current directory, Prints Working Directory, displays Date and displays Users logged in.**

**Source Code:**

#!/bin/bash

PS3="Select option: "

select opt in "List current directory" "Print working directory" "Display date" "Display users logged in" "Exit"; do

case $opt in

"List current directory")

ls

;;

"Print working directory")

pwd

;;

"Display date")

date

;;

"Display users logged in")

who

;;

"Exit")

exit

;;

\*)

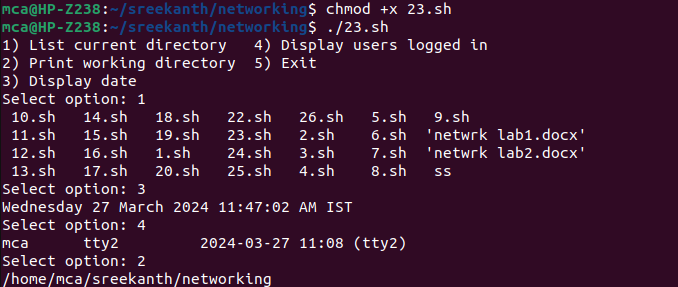
echo "Invalid option"

;;

esac

done

**Output:**

****

**24. Shell script to check executable rights for all files in the current directory, if a file does not have the execute permission then make it executable.**

**Source Code:**

#!/bin/bash

for file in \*; do

if [[ -f $file && ! -x $file ]]; then

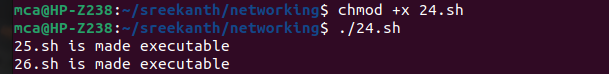
chmod u+rwx "$file"

echo "$file is made executable"

fi

done

**Output:**

****

**25.Write a Shell program to generate all combinations of 1, 2, and 3 using loop.**

**Source Code:**

#!/bin/bash

for i in 1 2 3; do

for j in 1 2 3; do

for k in 1 2 3; do

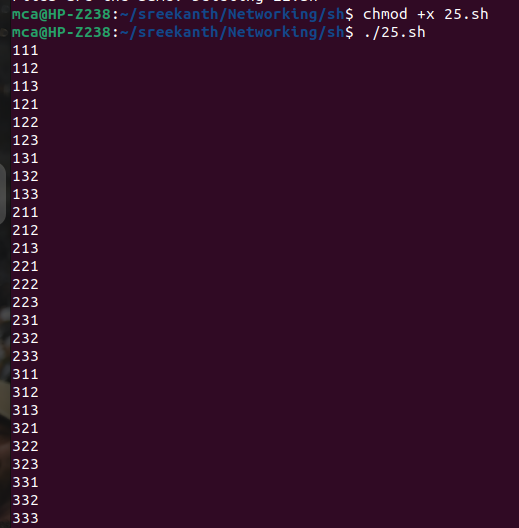
echo "$i$j$k"

done

done

done

**Output:**

****

**26.Write a Shell program to create the number series.**

**Source Code:**

#!/bin/bash

echo "Enter the number of terms:"

read n

echo "Number series:"

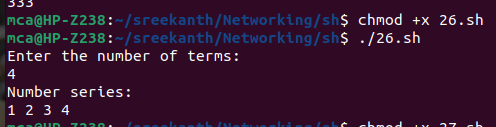
for ((i=1; i<=n; i++)); do

echo -n "$i "

done

echo ""

**Output:**

****

**27.Write a Shell program to create Pascal’s triangle.**

**Source Code:**

#!/bin/bash

echo "Enter the number of rows:"

read rows

for ((i=0; i<rows; i++)); do

for ((j=0; j<=i; j++)); do

if [ $j -eq 0 ] || [ $i -eq $j ]; then

coef=1

else

num=$((i-j+1))

den=$j

coef=$((coef \* num / den))

fi

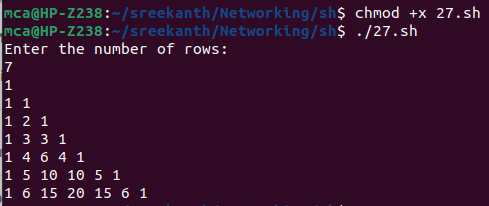
echo -n "$coef "

done

echo

done

**Output:**

****

**28.Write a Decimal to Binary Conversion Shell Script.**

**Source Code:**

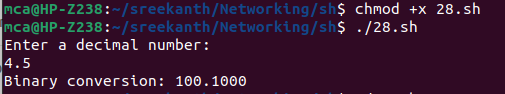
#!/bin/bash

echo "Enter a decimal number:"

read decimal

echo "Binary conversion: $(echo "obase=2; $decimal" | bc)"

**Output:**

****

**29.Write a Shell Script to Check Whether a String is Palindrome or not**

**Source Code:**

#!/bin/bash

echo "Enter a string:"

read str

reverse=$(echo $str | rev)

if [ "$str" = "$reverse" ]; then

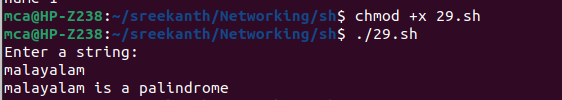
echo "$str is a palindrome"

else

echo "$str is not a palindrome"

fi

**Output:**

****

**30.Write a shell script to find out the unique words in a file and also count the occurrence of each of these words.**

**Source Code:**

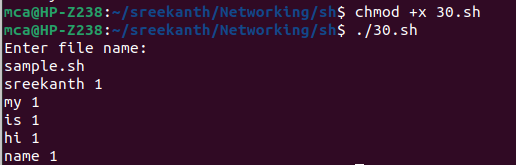
#!/bin/bash

echo "Enter file name:"

read filename

awk '{for(i=1;i<=NF;i++) a[$i]++} END {for(k in a) print k, a[k]}' $filename

**Output:**

****

**31.Write a shell script to get the total count of the word “Linux” in all the “.txt” files and also across files present in subdirectories.**

**Source code:**

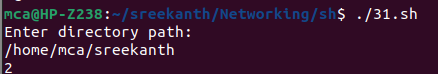
#!/bin/bash

echo "Enter directory path:"

read dir

grep -roh "Linux" $dir | wc -w

**Output:**

****

**32.Write a shell script to validate password strength. Here are a few assumptions for the password string. ( Length – minimum of 8 characters. Contain both alphabet and number. Include both the small and capital case letters.)**

**Source Code:**

#!/bin/bash

echo "Enter password:"

read password

if [[ ${#password} -lt 8 ]]; then

echo "Password length should be at least 8 characters"

exit 1

fi

if ! [[ $password =~ [0-9] ]]; then

echo "Password should contain at least one digit"

exit 1

fi

if ! [[ $password =~ [A-Z] ]]; then

echo "Password should contain at least one uppercase letter"

exit 1

fi

if ! [[ $password =~ [a-z] ]]; then

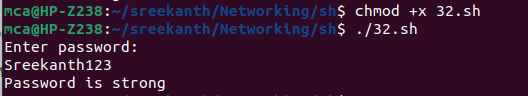
echo "Password should contain at least one lowercase letter"

exit 1

fi

echo "Password is strong"

**Output:**

****

**33.Write a shell script to print the count of files and subdirectories in the specified directory.**

**Source Code:**

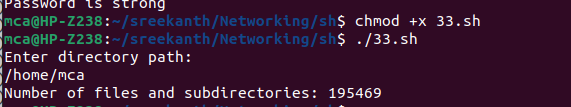
#!/bin/bash

echo "Enter directory path:"

read dir

echo "Number of files and subdirectories: $(find $dir -type d -or -type f | wc -l)"

**Output:**

****

**34.Write a shell script to reverse the list of strings and reverse each string further in the list.**

**Source Code:**

#!/bin/bash

echo "Enter strings separated by space:"

read -a strings

leng=${#strings[@]}

echo $leng

reversed\_strings=()

for ((i=($leng-1); i>=0; i--)); do

reversed\_strings+=(${strings[i]})

done

echo "Initial array : ${strings[@]}"

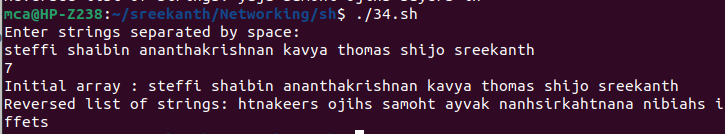
echo -n "Reversed list of strings: "

for string in "${reversed\_strings[@]}"; do

echo -n "$(echo $string | rev) "

done; echo

**Output:**

****

**5. File system hierarchy in a common Linux distribution, file and device permissions,study of system configuration files in /etc, familiarizing log files for system events, user activity, network events.**

The Linux File Hierarchy Structure or the Filesystem Hierarchy Standard (FHS) defines the

directory structure and directory contents in Unix-like operating systems.It is maintained by

the Linux Foundation.

 In the FHS, all files and directories appear under the root directory /, even if they are

stored on different physical or virtual devices.

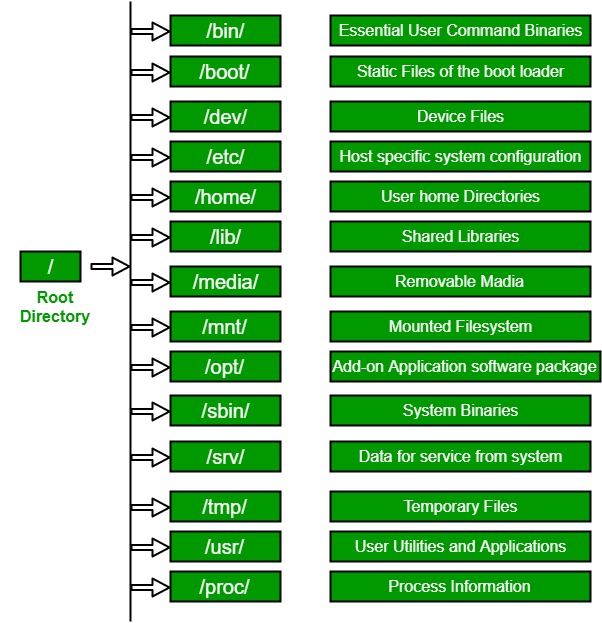
 Some of these directories only exist on a particular system if certain subsystems, such

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.



**– The Root Directory**

Everything on your Linux system is located under the / directory, known as the root directory.You can think of the / directory as being similar to the C:\ directory on Windows – but this isn’t strictly true, as Linux doesn’t have drive letters. While another partition would be located at D:\ on Windows, this other partition would appear in another folder under / on Linux.

**/bin :** Essential command binaries that need to be available in single user mode; for all users,

e.g., 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 e.g. ps, ls, ping, grep,

Cp

The /bin directory contains the essential user binaries (programs) that must be present when

the system is mounted in single-user mode. Applications such as Firefox are stored in /usr/bin,while important system programs and utilities such as the bash shell are located in /bin. The /usr directory may be stored on another partition – placing these files in the /bin directory ensures the system will have these important utilities even if no other file systems are mounted.The /sbin directory is similar – it contains essential system administration binaries.

**/boot – Static Boot File:** The /boot directory contains the files needed to boot the system – for example, the GRUB boot loader’s files and your Linux kernels are stored here. The boot

loader’s configuration files aren’t located here, though – they’re in /etc with the other

configuration files.

**/cdrom –** Historical Mount Point for CD-ROMs

The /cdrom directory isn’t part of the FHS standard, but you’ll still find it on Ubuntu and other operating systems. It’s a temporary location for CD-ROMs inserted in the system. However,the standard location for temporary media is inside the /media directory.

**/dev – Device Files**

Linux exposes devices as files, and the /dev directory contains a number of special files that

represent devices. These are not actual files as we know them, but they appear as files – for

example, /dev/sda represents the first SATA drive in the system. If you wanted to partition it, you could start a partition editor and tell it to edit /dev/sda.

This directory also contains pseudo-devices, which are virtual devices that don’t actually

correspond to hardware. For example, /dev/random produces random numbers. /dev/null is a

special device that produces no output and automatically discards all input – when you pipe

the output of a command to /dev/null, you discard it.

**/etc – Configuration Files**

The /etc directory contains configuration files, which can generally be edited by hand in a text editor. Note that the /etc/ directory contains system-wide configuration files – user-specific configuration files are located in each user’s home directory**.**

**/home – Home Folders**

The /home directory contains a home folder for each user. For example, if your user name is

bob, you have a home folder located at /home/bob. This home folder contains the user’s data

files and user-specific configuration files. Each user only has write access to their own home

folder and must obtain elevated permissions (become the root user) to modify other files on the system.

**/lib – Essential Shared Libraries**

The /lib directory contains libraries needed by the essential binaries in the /bin and /sbin folder.Libraries needed by the binaries in the /usr/bin folder are located in /usr/lib.

**/lost+found – Recovered Files**

Each Linux file system has a lost+found directory. If the file system crashes, a file system

check will be performed at next boot. Any corrupted files found will be placed in the lost+found directory, so you can attempt to recover as much data as possible.

**/media – Removable Media**

The /media directory contains subdirectories where removable media devices inserted into the computer are mounted. For example, when you insert a CD into your Linux system, a directory will automatically be created inside the /media directory. You can access the contents of the CD inside this directory.

**/mnt – Temporary Mount Points**

Historically speaking, the /mnt directory is where system administrators mounted temporary

file systems while using them. For example, if you’re mounting a Windows partition to perform

some file recovery operations, you might mount it at /mnt/windows. However, you can mount other file systems anywhere on the system.

**/opt – Optional Packages**

The /opt directory contains subdirectories for optional software packages. It’s commonly used

by proprietary software that doesn’t obey the standard file system hierarchy – for example, a

proprietary program might dump its files in /opt/application when you install it.

**/proc – Kernel & Process Files**

The /proc directory similar to the /dev directory because it doesn’t contain standard files. It

contains special files that represent system and process information.

**/root – Root Home Directory**

The /root directory is the home directory of the root user. Instead of being located at /home/root,

it’s located at /root. This is distinct from /, which is the system root directory.

**/run – Application State Files**

The /run directory is fairly new, and gives applications a standard place to store transient files

they require like sockets and process IDs. These files can’t be stored in /tmp because files in

/tmp may be deleted.

**/sbin – System Administration Binaries**

The /sbin directory is similar to the /bin directory. It contains essential binaries that are

generally intended to be run by the root user for system administration

**/selinux – SELinux Virtual File System**

If your Linux distribution uses SELinux for security (Fedora and Red Hat, for example), the

/selinux directory contains special files used by SELinux. It’s similar to /proc. Ubuntu doesn’t

use SELinux, so the presence of this folder on Ubuntu appears to be a bug.

**/srv – Service Data**

The /srv directory contains “data for services provided by the system.” If you were using the

Apache HTTP server to serve a website, you’d likely store your website’s files in a directory

inside the /srv directory.

**/tmp – Temporary Files**

Applications store temporary files in the /tmp directory. These files are generally deleted

whenever your system is restarted and may be deleted at any time by utilities such as tmpwatch**.**

**/usr – User Binaries & Read-Only Data**

The /usr directory contains applications and files used by users, as opposed to applications and files used by the system. For example, non-essential applications are located inside the /usr/bin directory instead of the /bin directory and non-essential system administration binaries are located in the /usr/sbin directory instead of the /sbin directory. Libraries for each are located inside the /usr/lib directory. The /usr directory also contains other directories – for example,architecture-independent files like graphics are located in /usr/share.The /usr/local directory is where locally compiled applications install to by default – this prevents them from mucking up the rest of the system.

**/var – Variable Data Files**

The /var directory is the writable counterpart to the /usr directory, which must be read-only in

normal operation. Log files and everything else that would normally be written to /usr during

normal operation are written to the /var directory. For example, you’ll find log files in /var/log.

**6. Installation and configuration of LAMP stack. Deploy an open source**

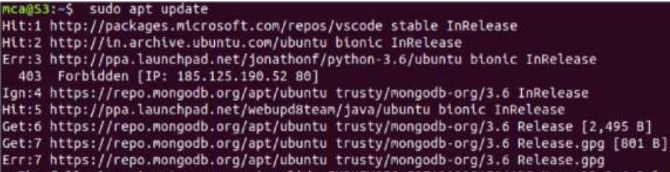
**application such as phpmyadmin and Wordpress.**

**Procedure**

**Install Apache2**

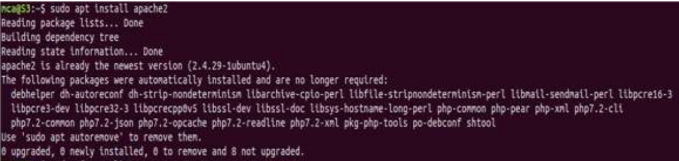
** Update your system:**

sudo apt update

****

** Install Apache using apt:**

sudo apt install apache2



** Confirm that Apache is now running with the following command:**

sudo systemctl status apache2

** If it is not working !**

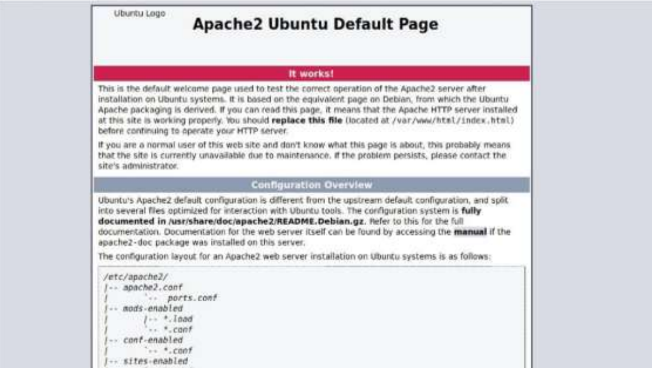
sudo systemctl stop apache2 # to stop if running

sudo systemctl start apache2 # to start if not running

** Once installed, test by accessing your server’s IP in your browser:**

<http://127.0.0.1/>

<http://localhost/>



**Install mariadb**

sudo apt install mariadb-server mariadb-client

sudo systemctl status mysql # to check status

sudo systemctl start mysq # if not running

sudo mysql\_secure\_installation # Secure your newly installed MariaDB

**Install PHP and commonly used modules**

 sudo apt install php libapache2-mod-php php-opcache php-cli php-gd

php- curl php-mysql

 sudo systemctl restart apache2

** Test PHP Processing on Web Server**

sudo nano /var/www/html/phpinfo.php

** Inside the file, type in the valid PHP code:**

<?php

phpinfo ();

?>

** Press CTRL + X to save and close the file. Press y and ENTER to**

**Confirm**

http://127.0.0.1/phpinfo.php

Install phpmyadmin

sudo apt install phpmyadmin php-mbstring php-zip php-gd php-json php-

curl

sudo systemctl restart apache2

** Open a browser :** [**http://localhost/phpmyadmin**](http://localhost/phpmyadmin)

**username:root**

**password : yourpassword If php my admin page not found :**

nano /etc/apache2/apache2.conf

** Add this line to last of the file.**

**Press CTRL + X to save and close the file. Press y and ENTER to**

**Confirm**

Include /etc/phpmyadmin/apache.conf

** restart apache2 - now try :** [**http://localhost/phpmyadmin**](http://localhost/phpmyadmin)

sudo systemctl restart apache2

** If any problem for login run the following command**

**sudo mysql**

ALTER USER root@localhost IDENTIFIED BY "yourpassword";

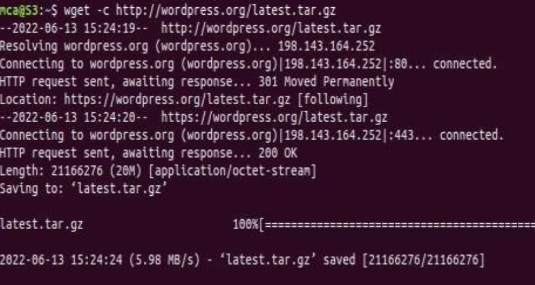
**Install WordPress with LAMP on Ubuntu 18.04**

**Step 1 – Download WordPress**

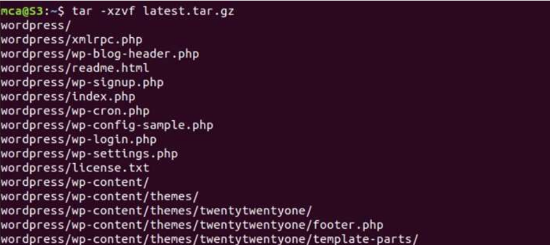
Download the latest version of the WordPress package and extract it by

issuing the commands below on the terminal:

 wget -c <http://wordpress.org/latest.tar.gz>



 tar -xzvf latest.tar.gz



Then move the WordPress files from the extracted folder to the Apache

default root directory, /var/www/html/:

 sudo mv wordpress/\* /var/www/html/

Next, set the correct permissions on the website directory, that is give

ownership of the WordPress files to the webserver as follows:

 sudo chown -R www-data:www-data /var/www/html/

 sudo chmod -R 755 /var/www/html/

**Step 2 – Creating a MySQL Database and User for WordPress**

The first step you’ll take is a preparatory one. Even though MySQL is

already installed, you still need to create a database to manage and store

the user information for WordPress to use. To get started, log into the

MySQL root(administrative) account by issuing the following command:

 sudo mysql

You will be prompted for the password you set for the MySQL root

accountwhen you installed the software. However, if you have password

authenticationenabled for your root user, you can run the following

command and enter your password information when prompted:

 mysql -u root –p

From there, you’ll create a new database that WordPress will control. You

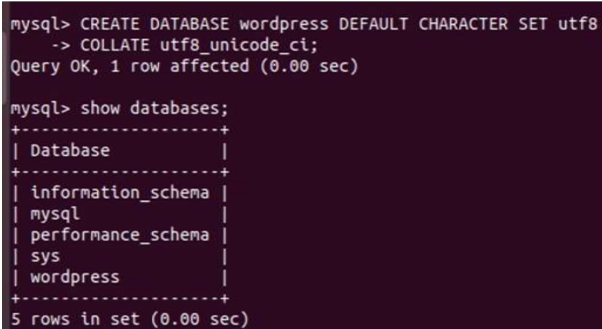
can call this whatever you would like, but we will be using wordpress in

this guideas an example. Create the database for WordPress by writing the

following:

 CREATE DATABASE wordpress DEFAULT CHARACTER

SET utf8 COLLATE utf8\_unicode\_ci;



Next, you’re going to create a separate MySQL user account that you’ll

useexclusively to operate on the new database. Creating one-function

databases andaccounts is a good idea from a management and security

standpoint. We willuse the name wordpressuser as an example in this

guide. Feel free to change thisif you’d like.You can create this account,

set a password for it, and then grant it access to thedatabase you created

all by running the following command. Remember tochoose a strong

password here for your database user:

 GRANT ALL ON wordpress.\* TO

'wordpressuser'@'localhost'IDENTIFIED BY 'password';

After creating this user, flush the privileges to ensure that the current

instance ofMySQL knows about the recent changes you’ve made:

 FLUSH PRIVILEGES;

Exit out of MySQL:

 EXIT

You now have a database and user account in MySQL, each made

specifically for WordPress. Go the /var/www/html/ directory and rename

existing wp-config-sample.php to wpconfig.php. Also, make sure to

remove the default Apache index page.

 cd /var/www/html/

 sudo mv wp-config-sample.php wp-config.php

 sudo rm -rf index.html







Then update it with your database information under the MySQL settings

section (refer to the highlighted boxes in the image below):This setting

can be added after the database connection settings, or anywhere

else in the file: Save and close the file when you are finished.Restart the

web server and mysql service using the commands below:

 sudo systemctl restart apache2.service

 sudo systemctl restart mysql.service



**Step 3 – Completing the Installation Through the Web Interface**

Now that the server configuration is complete, you can complete the

installation through the web interface. In your web browser, navigate to

your server’s domain name or public IP address:

 <https://server_domain_or_IP>

Select the language you would like to use: Next you will be directed to the

main setup page. Select a name for your WordPress site and choose a

username (it is recommended not to choose something like “admin” for

security purposes). A strong password is generated automatically. Save

this password or select an alternative strong password.Enter your email

address and select whether you want to discourage search engines from

indexing your site:Once you log in, you will be taken to the WordPress

administration dashboard: From there, you can begin using and

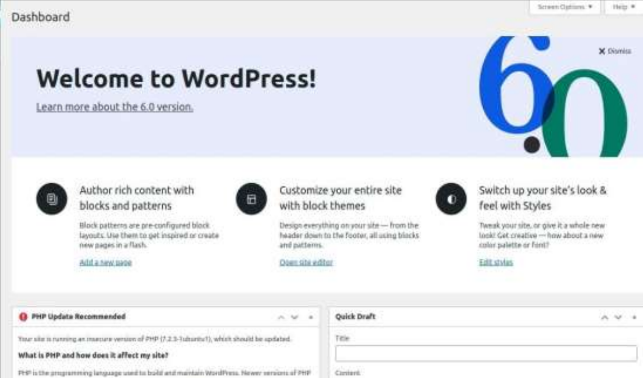
customizing your WordPress site



Once you log in, you will be taken to the WordPress administration

dashboard: From there, you can begin using and customizing your

WordPress site.



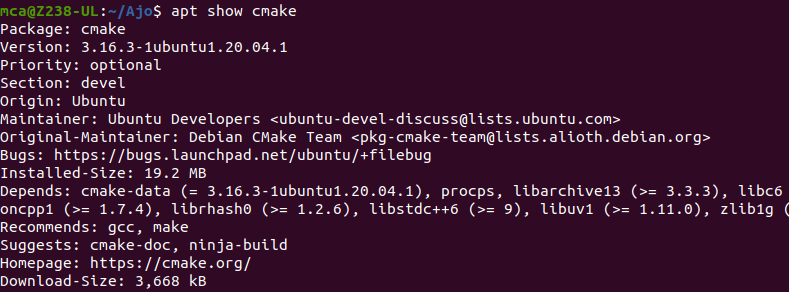
**7. Build and install software from source code, familiarity with make and**

**cmake utilities expected.**

**Procedure& Output Screenshot:**

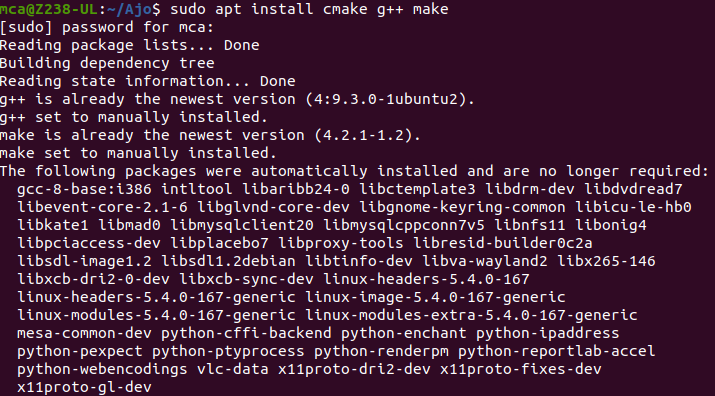
Install the cmake

**Apt show cmake**



 **$sudo apt install cmake g++ make:** To install cmake , g++ and make using the

apt command.



**Create directory**

 **Mkdir cmake**: creating a different directory for our project usingthe mkdir and cd commands.



** Cd cmake**



** gedit Helloworld.cpp**

Now create a C++ source file named Hello\_world.cpp and add the

following **: gedit CmakeLists.txt** Create a CMakeLists.txt

file(with this exact capitalization) which is required by CMake:

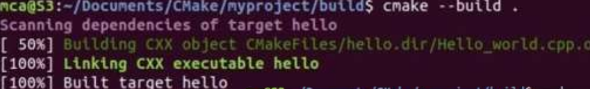
**Create directory called**

**Mkdir build:**

To run cmake we need to change into the build directory:

**Cmake..**

 Cmake –build : To generate the executable simply by typing:run hello



 ./hello: Run the executable by typing:



**8. Introduction to command line tools for networking IPv4 networking, network**

**commands: ping route traceroute, nslookup, ip. Setting up static and dynamic IP**

**addresses. Concept of Subnets, CIDR address schemes, Subnet masks, iptables, setting**

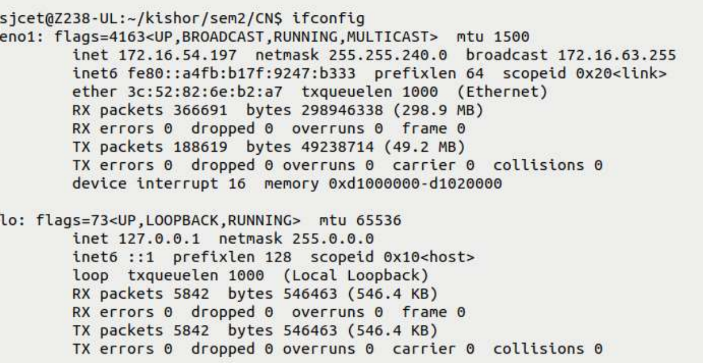
**up a firewall for LAN, Application layer (L7) proxies.**

**Procedure:**

**a**. **ifconfig**:This commands in windows allows you to see a summarized information of your

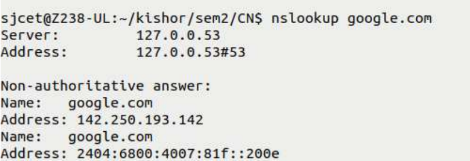
network such as ip address, subnet mask , server address etc.

Output



**b. nslookup** : To show the server to which the system is connected by default. If we want to

find the ip address of a particular domain name, we can also use nslookup

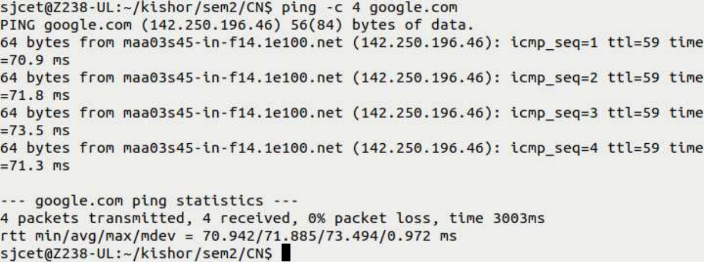


**c. ping :** The command used to check the availability of a host. The response shows the URL

you are pinging, the ip address associated with the URL and the size of packets being sent on

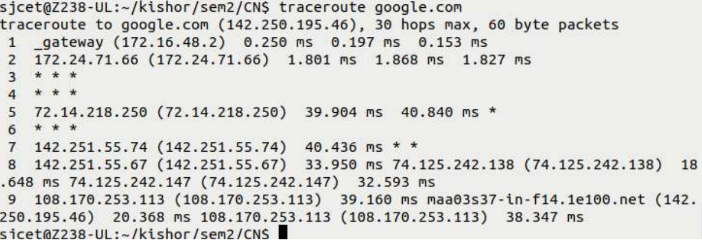
the first line . The next four lines shows the replies from each individual packets including the time(in milliseconds) for the response and the time to live(TLL) of the packet, that is the

amount of time that must pass before the packet discarded.



**d. traceroute :** traceroute is a command-line utility in Linux and other Unix-like operating

systems that allows you to track the path that packets take from your computer to a destination host on a network. It's used for troubleshooting network connectivity issues and identifying network delays

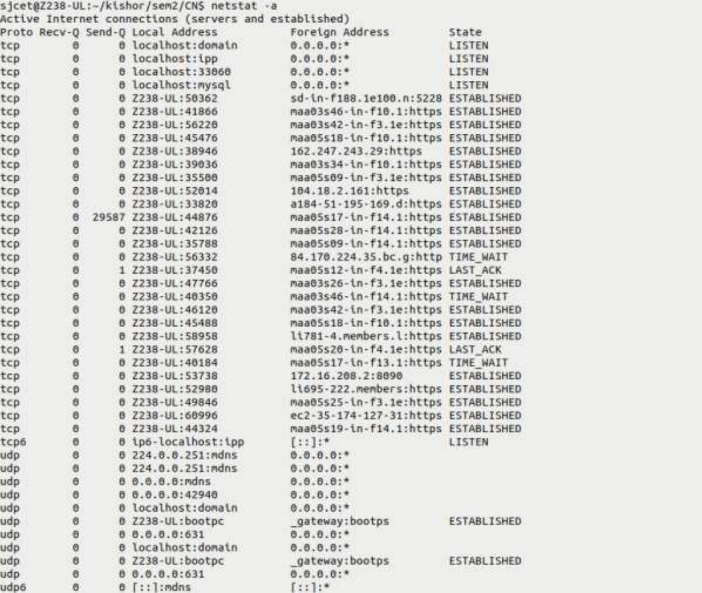


**e. netstat :** netstat is a command-line utility in Linux and other Unix-like operating systems

that provides information about network connections, routing tables, interface statistics,

masquerade connections, and more. It's used for monitoring network-related information and

diagnosing network issues.



**f. hostname :** The hostname command is a command-line utility in Linux and other Unix-like

operating systems that allows you to view or set the hostname of the system. The hostname is

the unique name assigned to a computer within a network.



**g. arp :** The arp command is a command-line utility in Linux and other Unix-like operating

systems that allows you to view and manipulate the Address Resolution Protocol (ARP) cache, which is used to map IP addresses to MAC addresses on a local network. ARP is essential for communication between devices within the same subnet.



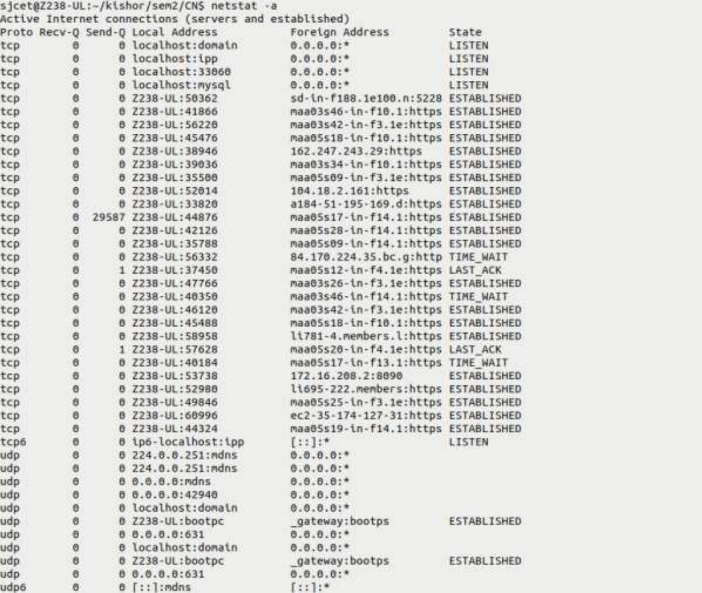
**h. uname :** The uname command is a command-line utility in Linux and other Unix-like

operating systems that provides information about the system's kernel and operating system.

It's used to retrieve information about the system's architecture, release version, and other

details.





**f. hostname :** The hostname command is a command-line utility in Linux and other Unix-like

operating systems that allows you to view or set the hostname of the system. The hostname is

the unique name assigned to a computer within a network.



**g. arp :** The arp command is a command-line utility in Linux and other Unix-like operating

systems that allows you to view and manipulate the Address Resolution Protocol (ARP) cache, which is used to map IP addresses to MAC addresses on a local network. ARP is essential for communication between devices within the same subnet.



**h. uname :** The uname command is a command-line utility in Linux and other Unix-like

operating systems that provides information about the system's kernel and operating system.

It's used to retrieve information about the system's architecture, release version, and other

details.



**9. Analyzing network packet stream using tcpdump and wireshark. Perform basic**

**network service tests using nc.**

**Procedure:**

**a. How to Install tcpdump in Linux**

Many Linux distributions already shipped with the tcpdump tool, if in case you

don’t have it on a system, you can install it using the command.

 $ sudo apt-get install tcpdump [On Debian, Ubuntu and Mint]

**b. Display Available Interfaces**

To list the number of available interfaces on the system, run the following command

with -D option.

**c. Capture Packets from Specific Interface**

The command screen will scroll up until you interrupt and when we execute the

tcpdump command it will captures from all the interfaces, however with -i switch

only capture from the desired interface.

**d. Capture Only N Number of Packets**

When you run the tcpdump command it will capture all the packets for the

specified interface, until you hit the cancel button. But using -c option, you can

capture a specified number of packets.

# tcpdump –c 5 –i enp3s0

**e. Display Captured Packets in HEX and ASCII**

The following command with option -XX capture the data of each packet,

including its link level header in HEX and ASCII format

**f. Capture and Save Packets in a File**

As we said, that tcpdump has a feature to capture and save the file in a .pcap

format, to do this just execute the command with -w option.

**g. Capture Packet from Specific Port**

Let’s say you want to capture packets for specific port 80, execute the below

command by specifying port number 80

**h. Read Captured Packets File**

To read and analyze captured packet 0001.pcap file use the command with -r

option

wire shark

Installing Wireshark on Ubuntu 20.04

The Wireshark utility is available on all major desktop platforms, i.e., Linux, Microsoft

Windows, FreeBSD, MacOS, Solaris, and many more. Follow the steps below to install

Wireshark on Ubuntu 20.04.

**Step 1 :** Update APT

First, as always, update and upgrade your APT through the following command.

Syntax:

$ sudo apt update

**Step 2:** Download and Install Wireshark

Now that Wireshark’s latest version has been added to the APT, you can

download and install it with the following command.

syntax

$ sudo apt install wireshark

**Step 3:** Enable Root Privileges

When Wireshark installs on your system, you will be prompted by the following

window. As Wireshark requires superuser/root privileges to operate, this option

asks to enable or disable permissions for all every user on the system. Press the

“Yes” button to allow other users, or press the “No” button to restrict other users

from using Wireshark.

**Step 4:**

You must add a username to the Wireshark group so that this user can use

Wireshark. To do this, execute the following command, adding your required

username after “wireshark” in the command.

**Syntax:**

$ sudo adduser $user wireshark

**Step 5:** Launch Wireshark

In the terminal window, type the following command to start the

Wireshark application. **Syntax:**

$ wireshark

You can also open Wireshark through the Graphical User Interface (GUI) by opening

the activities on the Ubuntu desktop, and in the search bar, type “Wireshark,” and click

on the application result.

**10.a) Introduction to Hypervisors and VMs, Xen or KVM .**

**Procedure:**

For the Ubuntu system, all packages required to run KVM are available on official upstream

repositories.

Install them using the commands:

 sudo apt update

 apt-get install qemu qemu-kvm libvirt-bin bridge-utils virt-manager virtviewer–

y

Create Virtual Machine • You can create virtual machine using virt-manager utility. Run the

following command to start the virt-manager:

 sudo virt-manager

 virsh help

 virsh help

 virsh help list

 Sudo virsh nodeinfo

 Virsh start

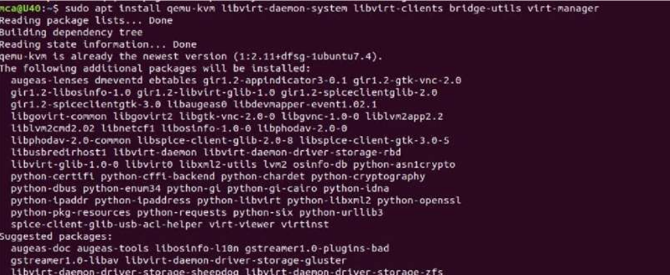
 vm virsh start

 virsh start testvm1

**Step 1:** Update the repositories

**Step 2:**Install essential KVM packages

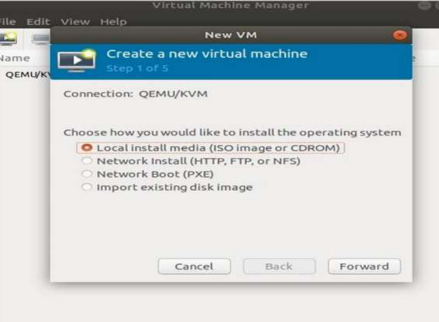
Install virt-manager, a tool for creating and managing VMs



**Step 3:** Start virt-manager with

**Step 4:**In the first window, click the computer icon in the upper-left corner,In the dialogue box

that opens, select the option to install the VM using an ISO image. Then click Forward.



**Step 5:** Choose ISO, click Forward

**Step 6:** Enter the amount of RAM and the number of CPUs you wish to allocate to the VM and

proceed to the next step.

**Step 7:** Allocate hard disk space to the VM. Click Forward to go to the last step.



**Step 8:** Specify the name for your VM and click Finish to complete the setup.

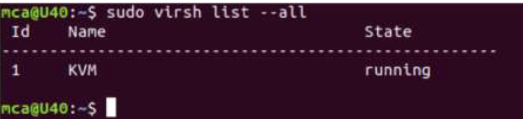


**Step 9:** Select language

**Step 10:** The VM starts automatically, prompting you to start installing the OS that’s on

the ISO file.

**Step 11:** Check the state of KVM



**b)Introduction to Containers: Docker installation and deployment**

**Procedure:**

**Steps for Installing Docker:**

**Step 1 :** Open the terminal on Ubuntu.

**Step 2 :** Remove any Docker files that are running in the system, using the following command

 Command : $ sudo apt-get remove docker docker-engine docker.io

After entering the above command, you will need to enter the password of the root and press

enter.

**Step 3 :** Check if the system is up-to-date using the following command:

 Command : $ sudo apt-get update

**Step 4 :** Install Docker using the following command:

 Command :$ sudo apt install docker.io

You’ll then get a prompt asking you to choose between y/n – choose ‘y’

**Step 5 :** Install all the dependency packages using the following command:

 Command :$ sudo snap install docker

**Step 6 :** Before testing Docker, check the version installed using the following command:

 Command :$ docker –version

**Step 7 :** Pull an image from the Docker hub using the following command:

 Command : $ sudo docker run hello-world

Here, hello-world is the docker image present on the Docker hub.

**Step 8 :** Check if the docker image has been pulled and is present in your system using the

following command:

 Command : $ sudo docker images

Step 9 : To display all the containers pulled, use the following command:

 Command : $ sudo docker ps -a

Step 10 : To check for containers in a running state, use the following command:

 Command : $ sudo docker ps

**11. Installing and configuring modern frameworks like Laravel typically involves setting up a web server, PHP, a database, and the framework itself. Below is a general guide to installing and configuring Laravel on a Linux system. Please note that specific steps may vary based on your distribution and environment.**

**Step 1: Prerequisites**

 Install Required Software: Make sure you have a web server (e.g., Apache or Nginx),

PHP, Composer (dependency manager), and a database server (e.g., MySQL) installed

on your system.

** Install Composer:** Download and install Composer by following the instructions on

the official Composer website.

**Step 2: Install Laravel**

**1. Create a New Laravel Project:** Open a terminal and navigate to the directory where

you want to create your Laravel project. Run the following command:

 composer create-project --prefer-dist laravel/laravel myproject

This will create a new Laravel project named "myproject."

**Step 3: Configure the Web Server**

**1.Apache:**

 Create a new virtual host configuration for your Laravel project in your Apache

configuration.

 Set the DocumentRoot to the public directory of your Laravel project.

 Enable the necessary Apache modules (e.g., rewrite) and restart Apache.

**2. Nginx:**

 Create a new server block configuration for your Laravel project in your Nginx

configuration.

 Set the root directive to the public directory of your Laravel project.

 Configure the necessary location directives and restart Nginx

.

**Step 4: Configure Laravel**

**1.Environment Configuration:**

 Rename the .env.example file in your Laravel project root to .env.

 Set database connection details, application key, and other settings in the .env

File.

**2. Generate Application Key:** Run the following command in your Laravel project

directory:

 php artisan key:generate

**3. Run Migrations:** If your .env file is configured with database details, run migrations

to create necessary database tables:

 php artisan migrate

**Step 5: Testing the Setup**

**1. Access the Application**: Open a web browser and visit the URL you configured for

your Laravel project. You should see the Laravel welcome page.

**2. Create Routes and Views:** Begin building your application by defining routes and

creating views in the resources/views directory.