

**NETWORKING & SYSTEM ADMINISTRATION LAB
(20MCA136)**

LAB RECORD

Submitted in partial fulfilment of the requirements for the award of the degree
of Master of Computer Applications of A P J Abdul Kalam Technological
University

Submitted by:

CHRISTEENA JOY (SJC22MCA-2020)



**MASTER OF COMPUTER APPLICATIONS
ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY, PALAI
CHOONDACHERRY P.O, KOTTAYAM**

KERALA

August 2023

ST. JOSEPH' S COLLEGE OF ENGINEERING AND TECHNOLOGY, PALAI

(An ISO 9001: 2015 Certified College)

CHOONDACHERRY P.O, KOTTAYAM KERALA



CERTIFICATE

This is to certify that the NETWORKING & SYSTEM ADMINISTRATION LAB (20MCA136) submitted by Christeena Joy, student of Second semester MCA at ST. JOSEPH'S COLLEGE OF ENGINEERING AND TECHNOLOGY, PALAI in partial fulfilment for the award of Master of Computer Applications is a bonafide record of the lab work carried out by her under our guidance and supervision. This record in any form has not been submitted to any other University or Institute for any purpose.

Prof. Anish Augustine K

Faculty In- Charge

Prof. Anish Augustine K

(HoD In Charge-MCA)

Submitted for the End Semester Examination held on _____

Examiner 1:

Examiner 2:

DECLARATION

I Christeena Joy, do hereby declare that the NETWORKING & SYSTEM ADMINISTRATION LAB (20MCA136) is a record of work carried out under the guidance of Mr. Anish Augustine K , Asst.Professor, Department of Computer Applications, SJCET, Palai as per the requirement of the curriculum of Master of Computer Applications Programme of A P J Abdul Kalam Technological University, Thiruvananthapuram. Further, I also declare that this record has not been submitted, full or part thereof, in any University / Institution for the award of any Degree / Diploma.

Place: Choondacherry

CHRISTEENA JOY

Date :

(SJC22MCA-2020)

CONTENT

Sl. No.	Content	Page No.
1	Introduction to Computer hardware	1
2	Install latest version of Ubuntu on a virtual box	6
3	Study of a terminal based text editor such as Vim or Gedit, Basic Linux commands: - familiarity with following commands/operations expected	21
4	Shell scripting	31
4.1	Shell program to check the given number and its reverse are same.	32
4.2	Shell program to find the sum of odd and even numbers from a set of numbers	33
4.3	Shell program to find the roots of a quadratic equation	34
4.4	Shell program to generate prime numbers between 1 and 50	36
4.5	Shell program to find the sum of digits of a number using function	41
4.6	Shell program to print the reverse of a number using function	42
4.7	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.	43
4.8	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	45
4.9	Shell program to create Pascal's triangle	48

4.10	Shell script to find out the unique words in a file and also count the occurrence of each of these words	50
5	File system hierarchy in a common Linux distribution	54
6	Installation and configuration of LAMP stack. Deploy a/n open source application such as phpmyadmin and Wordpress.	60
7	Build and install software from source code, familiarity with make and cmake utilities expected	67
8	Introduction to command line tools for networking IPv4 networking, network commands	69
9	Analyzing network packet stream using tcpdump and wireshark	73
10	Introduction to Hypervisors and VMs, Xen or KVM , Introduction to Containers: Docker, installation and deployment	76
11	Installing and configuring modern frameworks like Laravel	81

1. Introduction to Computer hardware: Physical identification of major components of a computer system such as mother board, RAM modules, daughter cards, bus slots, SMPS, internal storage devices, interfacing ports.

Procedure:

What is Computer Hardware?

Computer hardware is a hardware part of a computer system. In simple words, only those parts of the computer system which we can see or touch are called computer hardware.

Hardware is an important part of our computer system without which the computer is incomplete. You cannot use a computer without hardware and without hardware, there cannot be a computer system or construction.

1. Mouse



A mouse is a hardware input device that is used to move the cursor or pointer on computer screens. It can also be used to run computer programs, select items in a graphical user interface, and manipulate objects in the computer world. Some common examples of how it can be used are clicking on buttons, scrolling up and down the screen, selecting files, opening folders, and so on.

2. Keyboard



A keyboard is an input device that you use to enter data into a computer. It's also called the input device for your computer. Keyboards are used with PCs, laptops, tablets, and other devices. There are many different types of keyboards, but the most common one is the QWERTY keyboard. A QWERTY keyboard has all the letters in alphabetical order on it. This is different from some other types of keyboards, like Dvorak or Colemak keyboards. For example, these keyboards have keys arranged differently than what you're used to seeing on a QWERTY keyboard. And that means that typing on these keyboards will feel like typing in another language at first! But don't worry - once you get accustomed to it, it feels natural!

3. Monitor



Personal computers use a monitor to display data, run the software, and interact with the user. A monitor is an electronic visual display that connects to your computer or laptop. It is used for displaying images, text, videos, games, web pages, and more. Monitors are available in different sizes depending on the needs of the person using them. The most common types of monitors are CRT (cathode ray tube), LCD (liquid crystal display), and LED (light-emitting diode).

4. Motherboard



The motherboard is the backbone of our computer system. It's the central processing unit or CPU. It connects all the other components, like memory and graphics card, to the power supply. The motherboard is where all the wires are plugged in and it's also where you place your RAM, which is your computer's working memory. The motherboard is what makes one machine different from another.

Motherboards are made up of tiny transistors that control the flow of electricity through copper tracks on their surface. These transistors are called Integrated Circuits or ICs for short.

5. CPU (Central Processing Unit)



A CPU, or central processing unit, is the brain of a computer. The CPU processes information and runs programs. It functions as a control unit that executes programs according to instructions in its program memory. The CPU contains elements such as registers, an arithmetic logic unit (ALU), and control logic for sequencing instructions.

6. RAM Memory



A computer's RAM is a type of computer memory that stores information so the CPU can access it directly. Computer systems use main memory to store both data and programs. The more RAM you have, the more data your system can process at one time. This will lead to more efficient operations on your computer, which translates into better performance for the user.

7. ROM Memory



ROM stands for a type of memory chip that can be read from but not written to. In other words, it's a form of data storage that can't be changed after being programmed. It's sometimes called "non-volatile" memory because the stored information will remain even when not powered up or in use. ROM is often used to store a computer's basic start-up instructions and certain types of data, such as your car's on-board computer system and a calculator's data tables.

8. Hard Disk Drive



A hard disk drive is a piece of hardware inside a computer that stores information. It's used to store software and data in a safe place, which can be accessed when needed. With magnetic storage, there are no moving parts - unlike a CD or DVD player in which you need to move a

disk in order to access data. You can think of it as "a closet" where all your stuff is stored safely. As long as you have power, you can get to your things when you need them.

9. Optical Drive



shutterstock.com · 117172483

Optical Drives are used in PCs to read and write CDs and DVDs. The optical drive reads the data from the disc, which can then be transformed into a digital file that is readable by the computer. This makes it easy to backup files, play music or movies, or copy data from one disc to another. The term "CD" refers to Compact Discs, which are the most common type of optical drive on modern computers. They are often used for installing software on your computer, moving data between computers, or writing new programs.

10. Power Supply



A power supply is an electrical appliance that provides the necessary power to operate a computer. Computers are powered by electricity, and the power supply converts the alternating current (AC) from the electric outlet into direct current (DC). The power supply in a computer can be an internal or external component.

2. Install latest version of Ubuntu on a virtualbox**Procedure:**

1. Download and Virtualbox Windows 10 Installation
2. Ubuntu ISO download
3. Install Virtualbox
4. Create an Ubuntu VM
5. Install Ubuntu on Virtualbox Windows 10 6. Install Virtualbox Guest Additions

Download and Virtualbox Windows 10 Installation

1. Install Ubuntu on VirtualBox
2. HowTo Install Ubuntu On VirtualBox?
 - 2.1. Open VirtualBox
 - 2.2. Click on “New” to create a virtual machine
 - 2.3. Enter Name for your Virtual Machine
 - 2.4. Select “Linux” Operating System from “Type”
 - 2.5. Click “Next”
 - 2.6. Enter amount of memory (RAM) = 1024 MB and click “Next”
 - 2.7. Click “Create” to create hard drive
 - 2.8. Click “Next”
 - 2.9. Click “Next”
 - 2.10. Enter Size of Virtual Hard Drive = 20 GB and Click “Create”
 - 2.11. Select Virtual Machine
 - 2.12. Click on “Start” to start the virtual machine
 - 2.13. Select disk file source
 - 2.14. After selecting the OS file to be installed click “Open”
 - 2.15. Click “Start”
 - 2.16. Click “Ok”
 - 2.17. Click “Install Ubuntu”

- 2.18. Click “Continue”
- 2.19. Click “Install Now”
- 2.20. Click “Continue”
- 2.21. Select location and click “Continue”
- 2.22. Select keyboard layout & click “Continue”
- 2.23. Fill all the details and Click “Continue”
- 2.24. Now the installation process will start and installation window will appear
- 2.25. Click “Restart Now”
- 2.26. When the system will get restarted the following message will appear. Press “Enter”
- 2.27. Close the pop-up messages by clicking on the Close (×) button
3. Steps To Maximize The Size Of Ubuntu Desktop
 - 3.1. Go to “Devices”
 - 3.2. Click “Insert Guest Additions CD Image...”
 - 3.3. Click “Run”
 - 3.4. Click “Authenticate”
 - 3.5. Press “Enter”
 - 3.6. Now “Restart” your system for the changes to be applied.
 - 3.7. After the system gets restarted. Go to “View”
 - 3.8. Click “Switch to Full screen”
 - 3.9. Click “Switch”

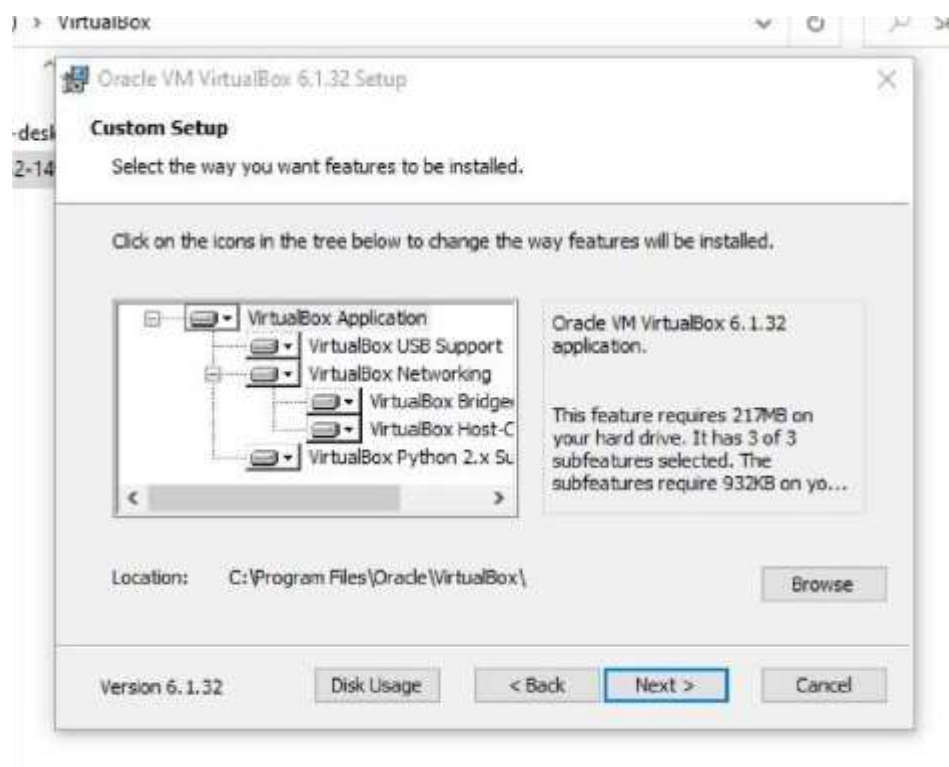
Output Screenshot

STEP 1: Installing Virtual Box.

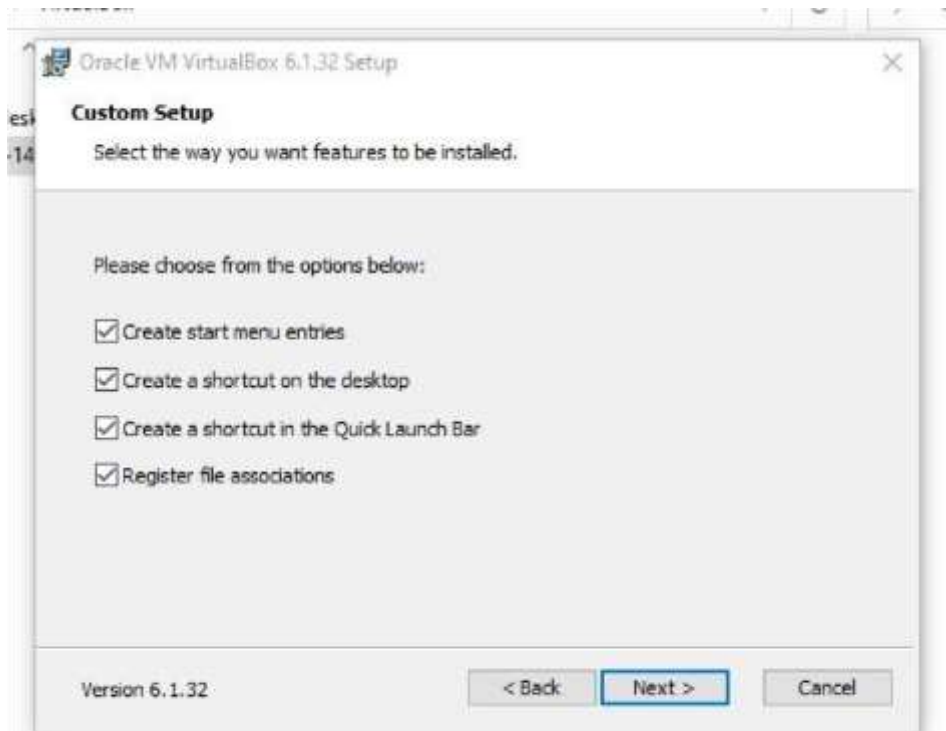
a. Starting pop-up window for the installation.



b. Custom setup window to select the features you want and select installation location



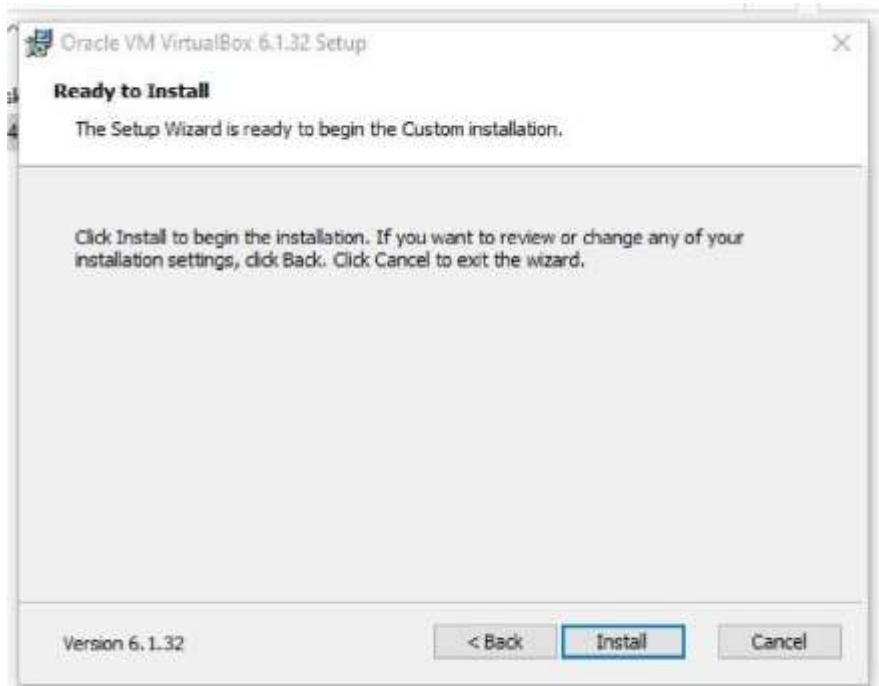
c. Custom setup window to choose from the option below.



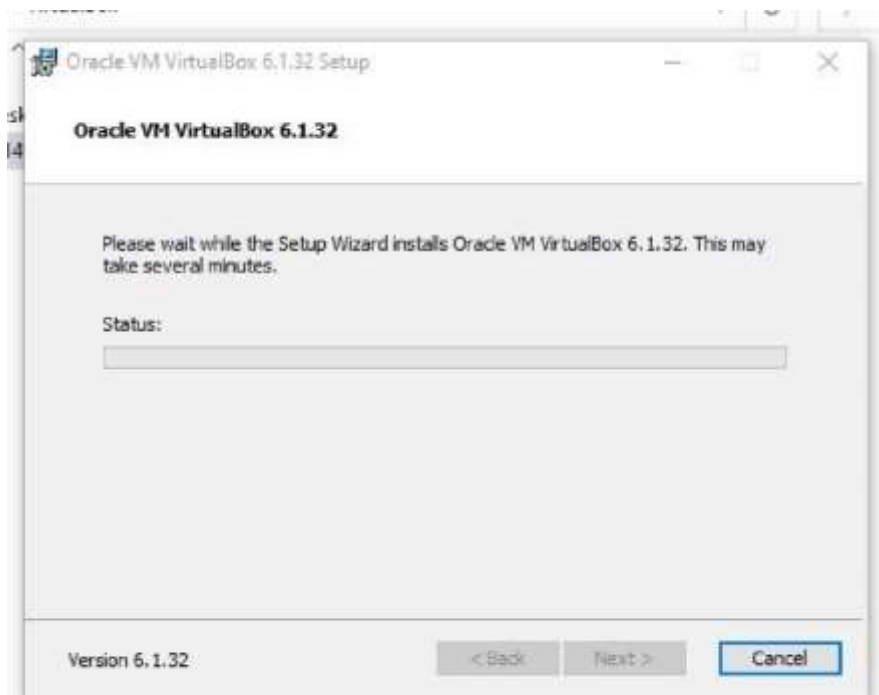
d. Custom setup window to confirm the installation.



e. Custom setup window to install the virtual box with the install button.



f. Installation box showing the installation status.



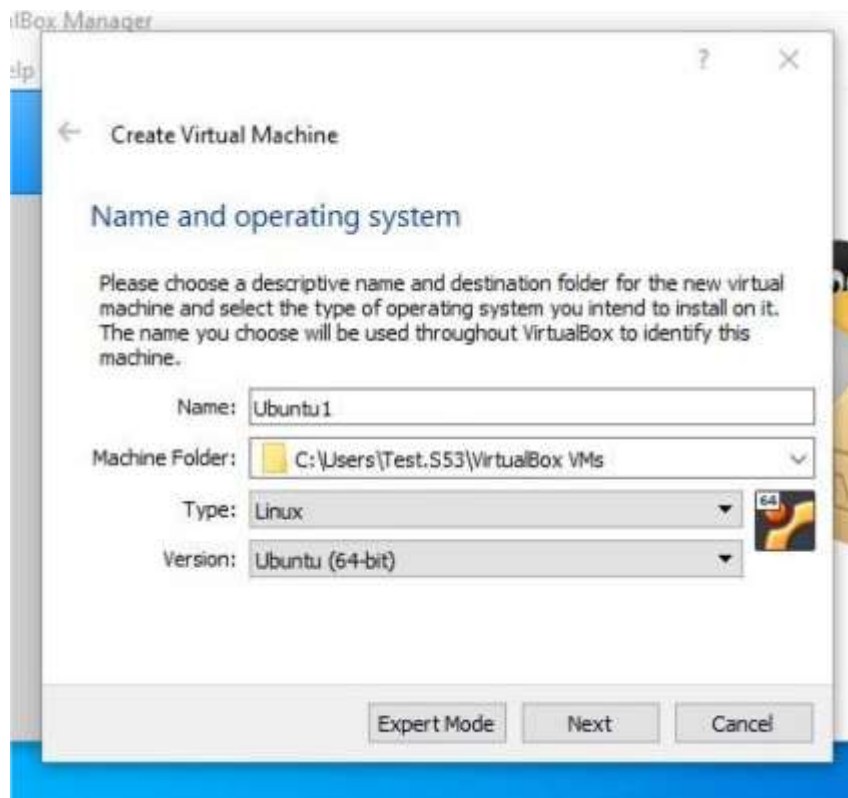
g. Installation complete pop-up windows with the finish button.



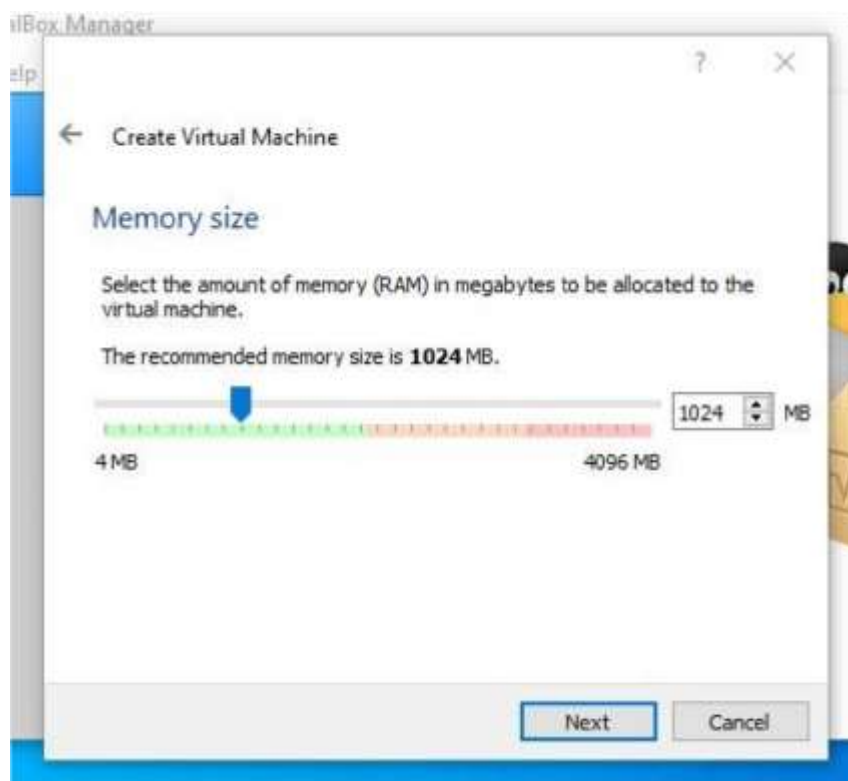
STEP 2: Setup the Ubuntu Instance in the VM Virtual Box.



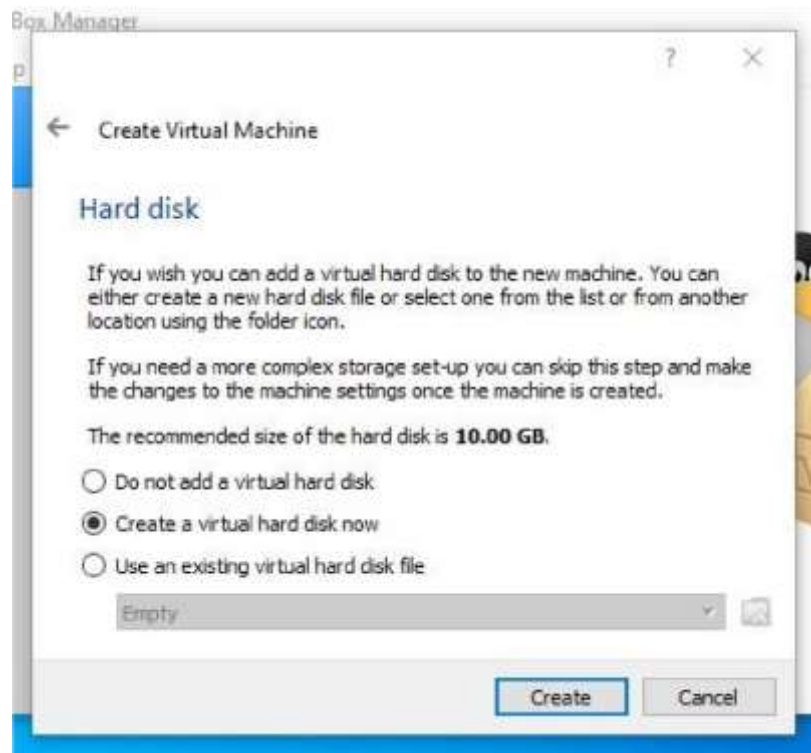
- a. After selecting the NEW button to create the Ubuntu instance, the pop-up window to enter & select the name, type and version of the OS.



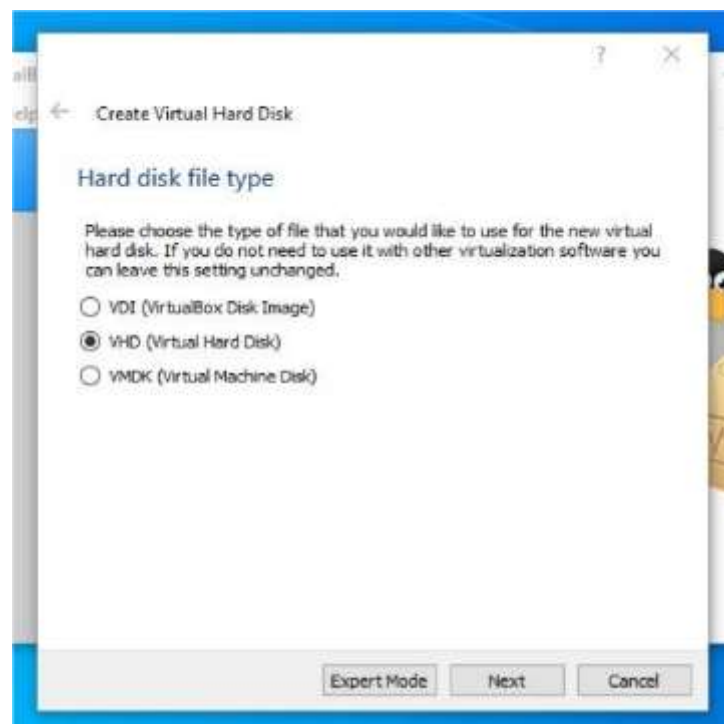
- b. Choose the main memory size for the OS.



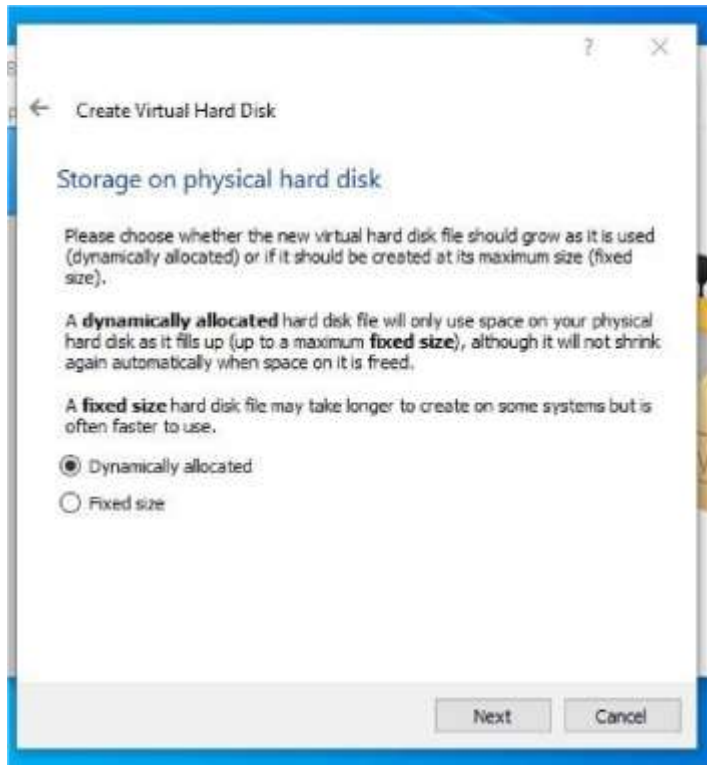
c. Option to add a virtual hard disk to the new machine instance.



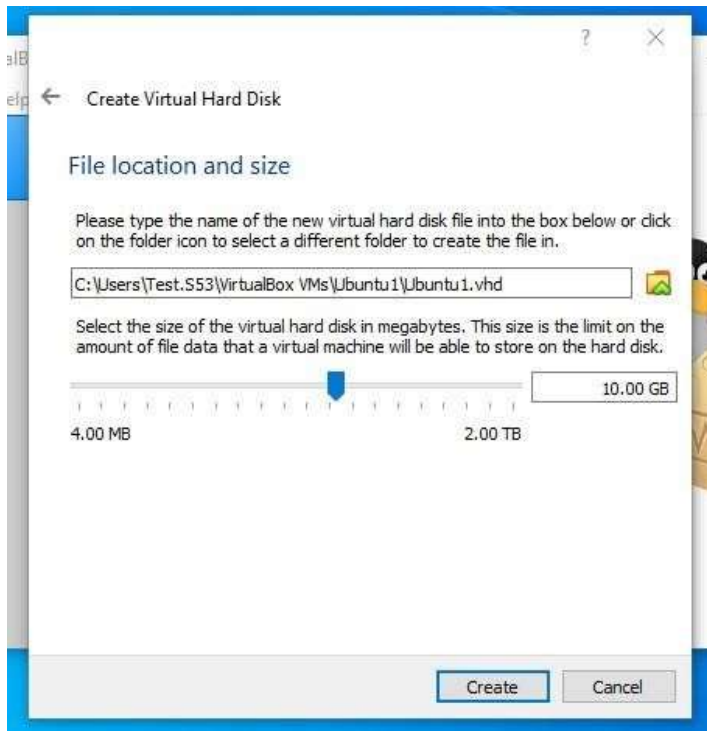
d. Option to choose the type of new virtual hard disk for new instance of OS.



e. Options to choose the methods of accessing the physical hard disk space for the new instance from the existing hard disk



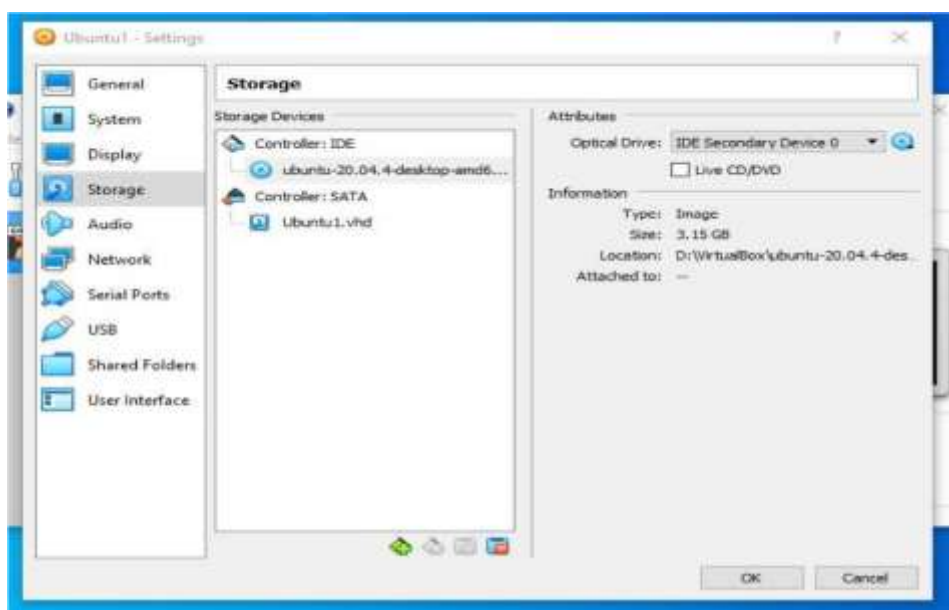
f. Panel to select the size of the virtual disk in megabytes and location and name of the instance and final submit to create the instance of OS.



g. The newly created OS instance and at the left side of the application.

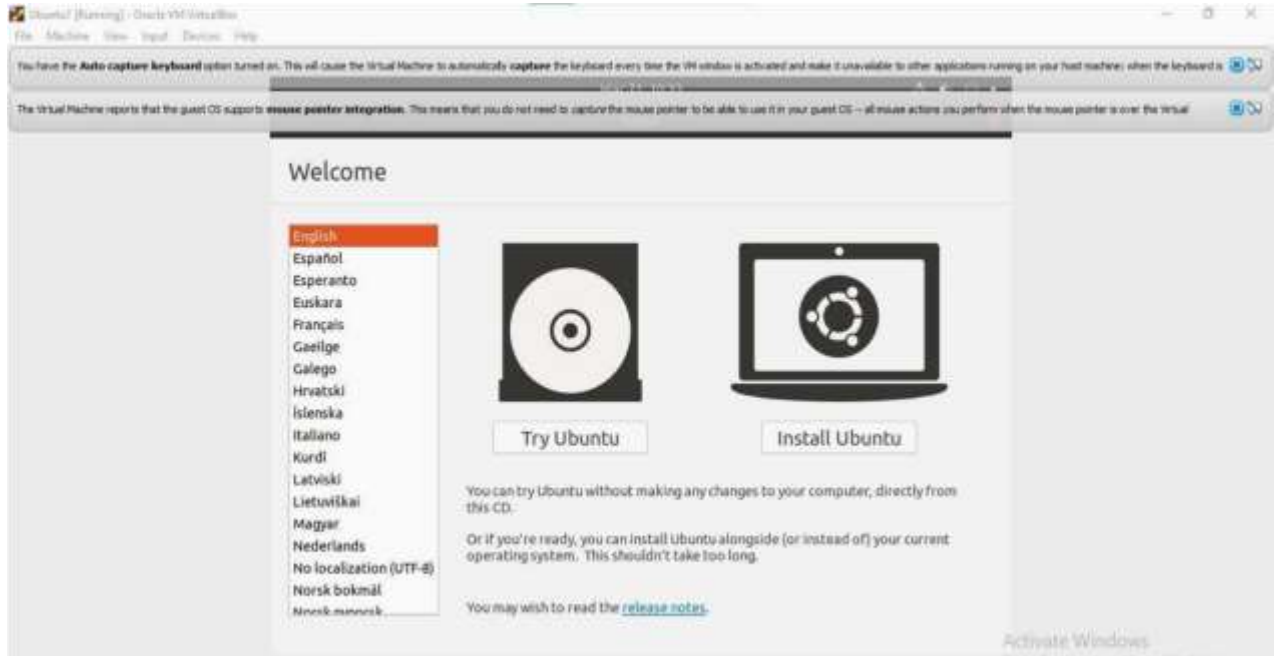


h. Settings the configurations of the instance created and adding the ISO image file of the OS correspondingly and Selecting the ISO image file from the local device.

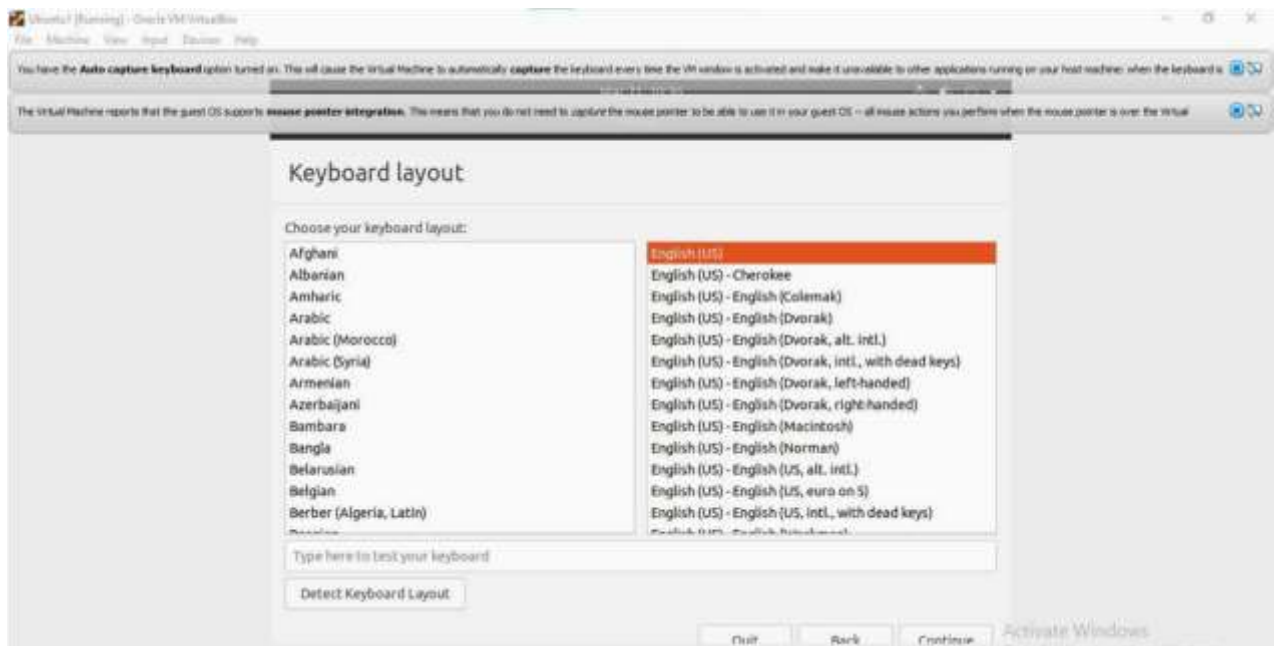


STEP 3: Installation of the Ubuntu OS within the newly created instance.

- a. Running the new OS instance and selecting the “Install Ubuntu” to install the loaded ISO file.



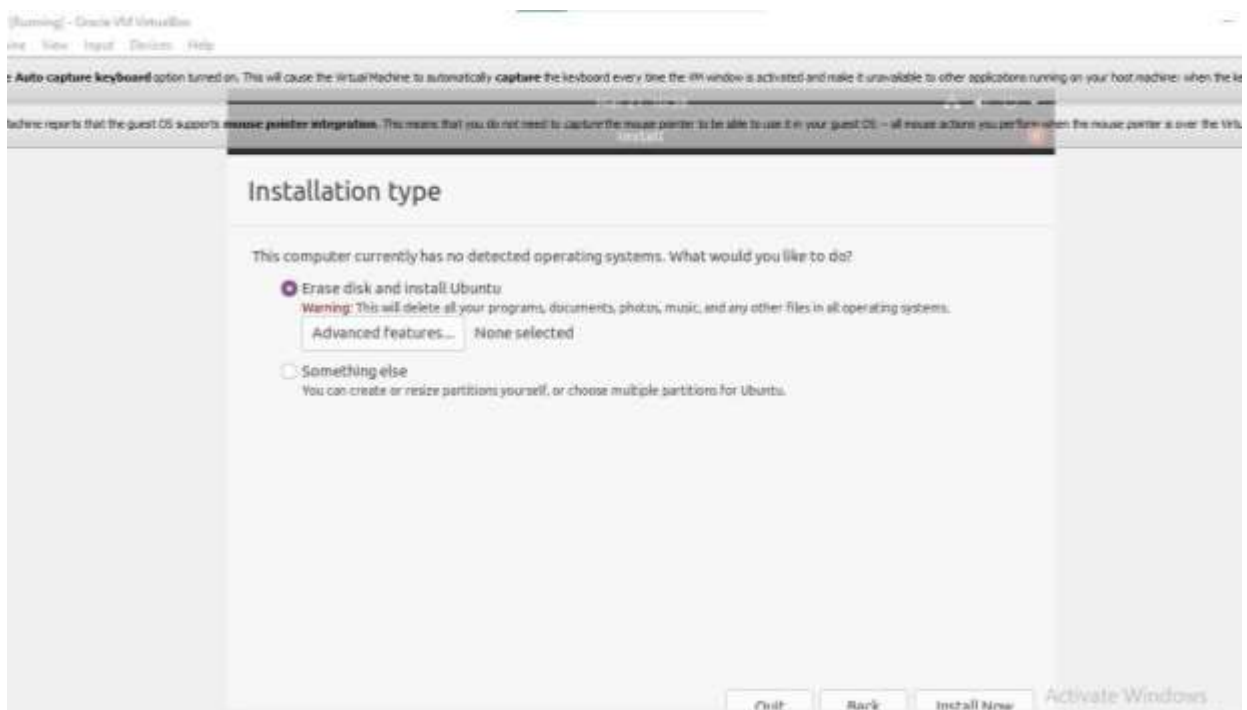
- b. Selecting the language for install the ubuntu OS.



c. Selection of other installation along & within with the installation of ubuntu.



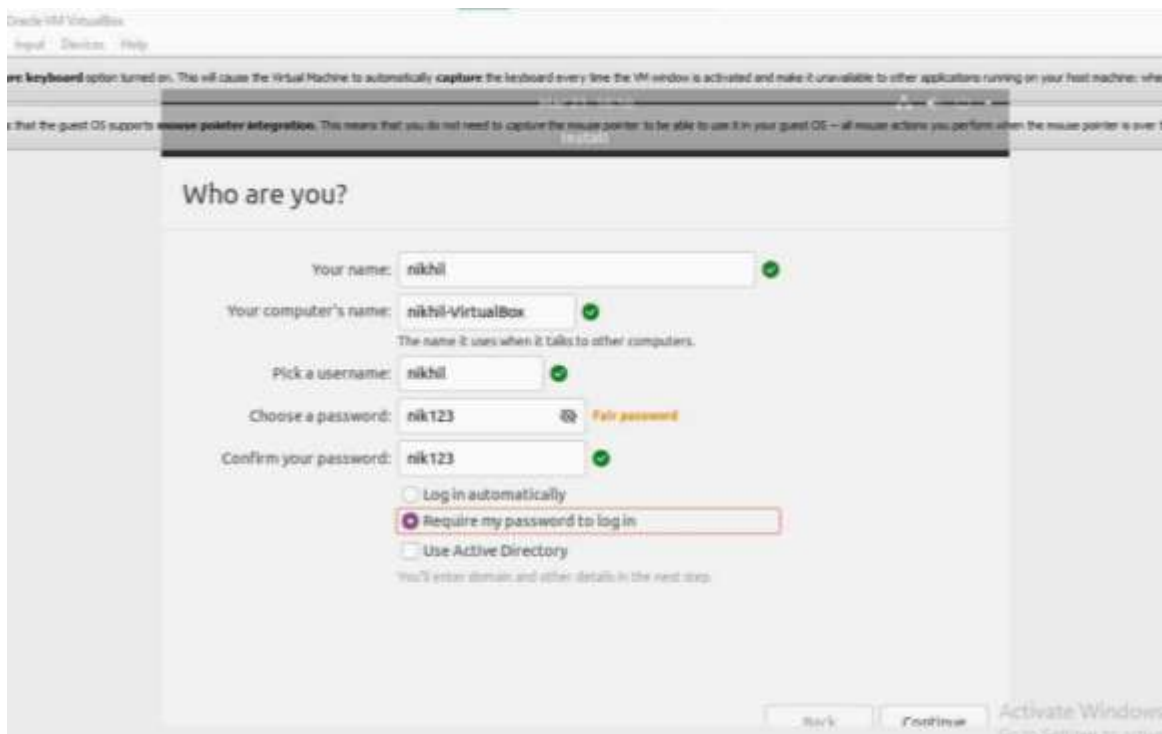
d. Selecting the disk partitioning allocation options from the given below.



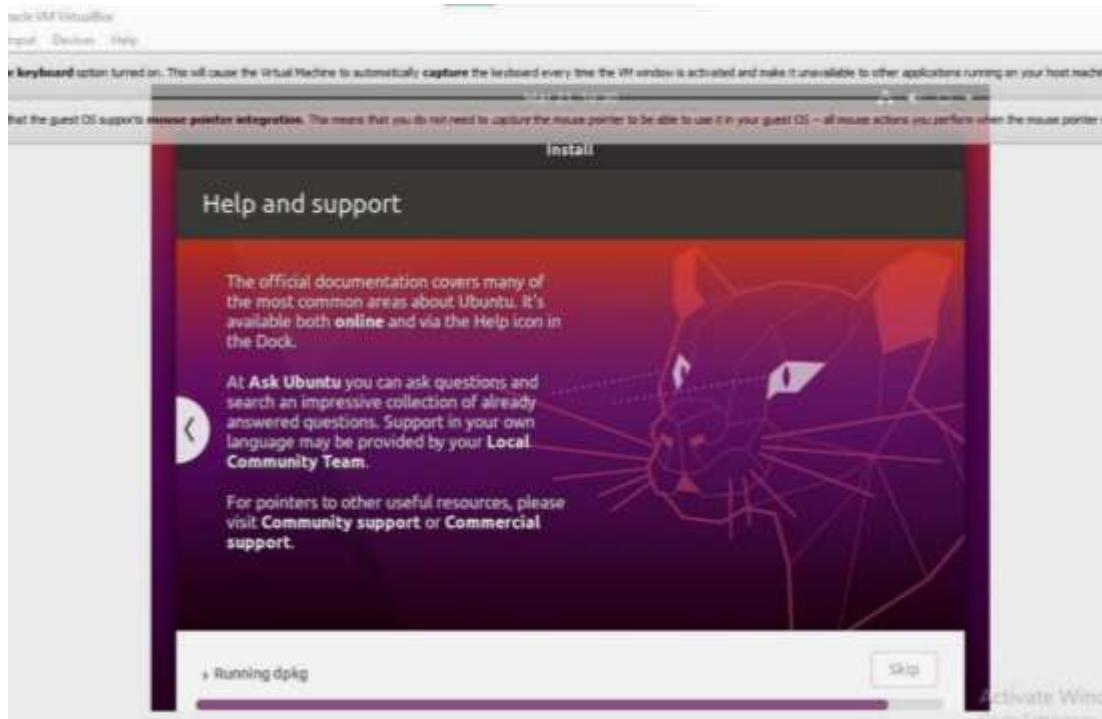
e. Selecting the geographical location for the time/location.



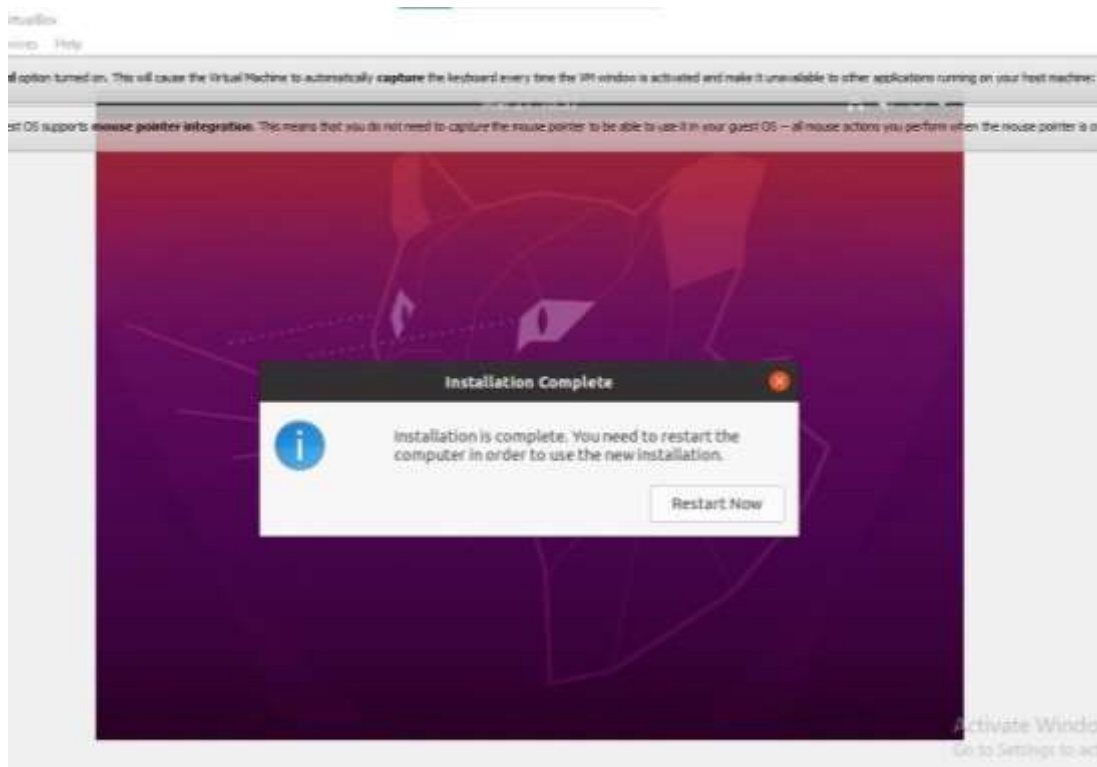
f. Entering the name, username & password for the account to sign in to the ubuntu OS after installation.



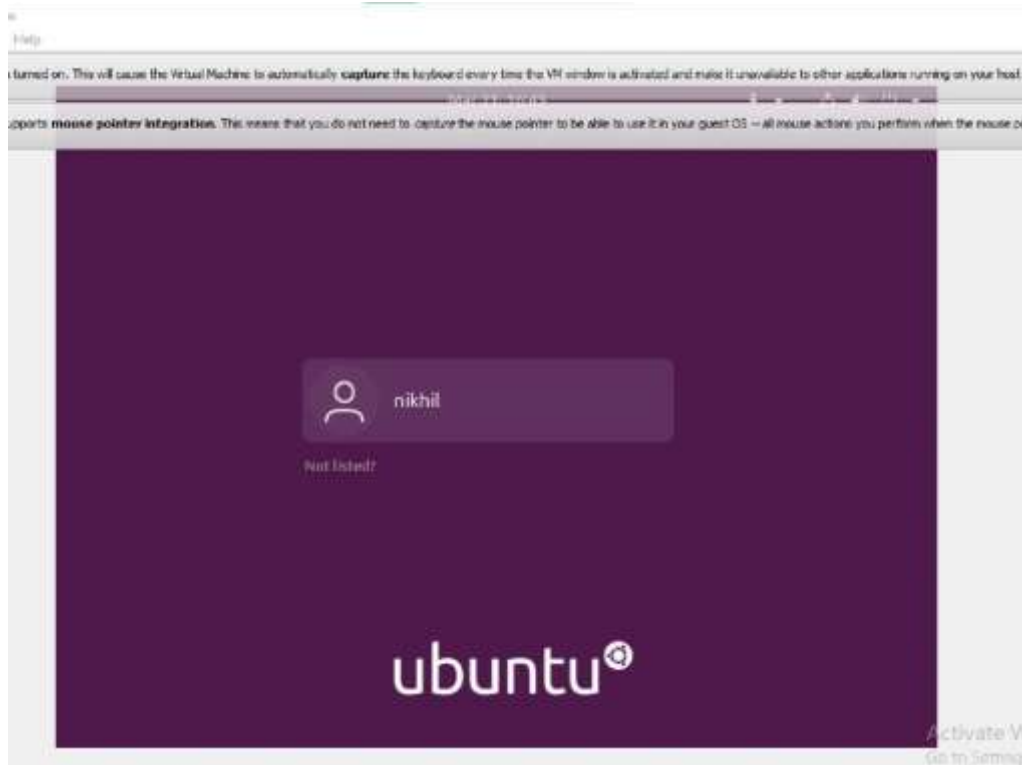
g. Installing the ubuntu OS in the instance, extracting the ubuntu ISO file, setting configurations, setting the various software within, etc.



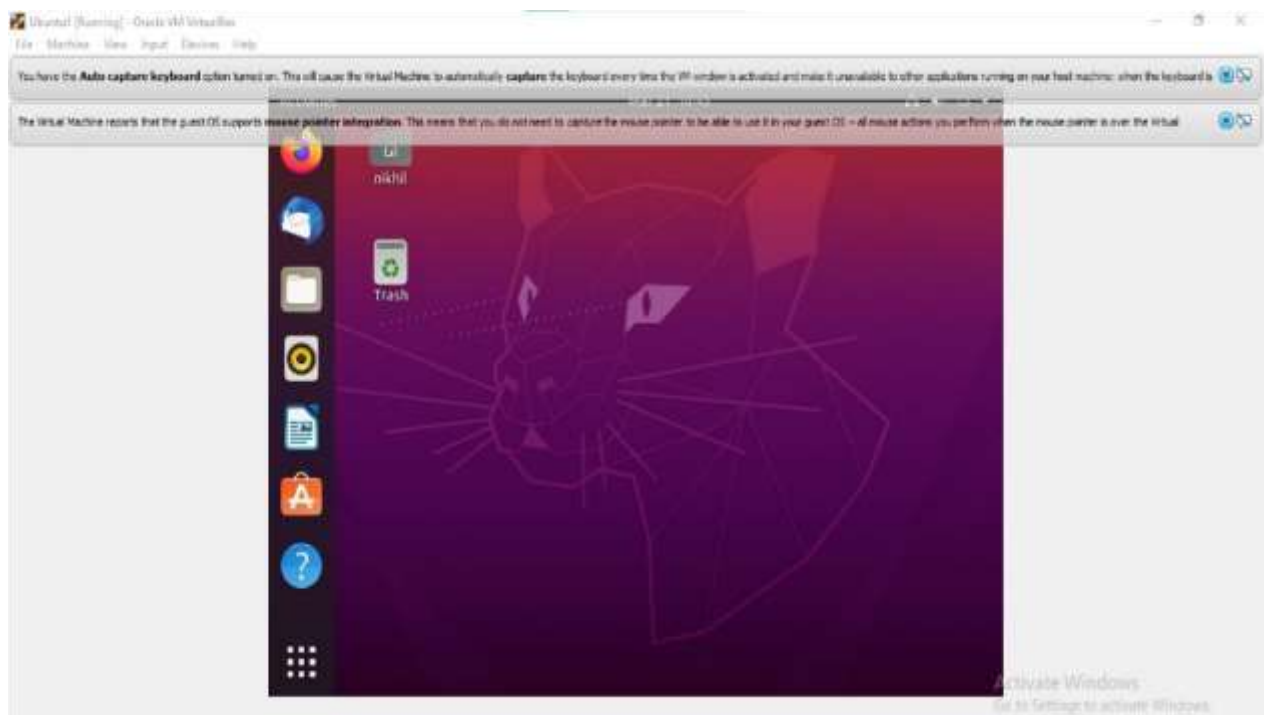
h. Restarting the OS instance to finalize the installation.



i. Signing in and visiting the home screen of the ubuntu OS using the previously registered username & password.



Output



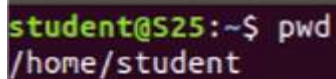
3. Study of a terminal based text editor such as Vim or Gedit, Basic Linux commands: - familiarity with following commands/operations expected

Procedure

Pwd: This command is used to display the location of the current working directory.

Syntax :- \$ pwd

Output

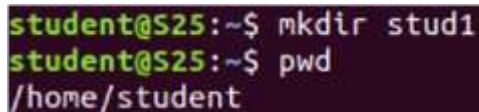


```
student@S25:~$ pwd
/home/student
```

Mkdir: This command is used to create a new directory under any directory.

Syntax :- \$ mkdir <directory name>

Output



```
student@S25:~$ mkdir stud1
student@S25:~$ pwd
/home/student
```

ls: This command is used to display a list of content of directory.

Syntax :- \$ ls

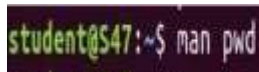
Output



```
student@S25:~$ ls
Desktop    Downloads    Music      Public      snap    Templates
Documents  examples.desktop  Pictures   PycharmProjects  stud1   Videos
```

Man: This command is used to display the user manual of any command that we can run on the terminal.

Syntax :- \$ man <command name>



```
student@S47:~$ man pwd
```

Cd: This command is used to change the current directory.

Syntax :- \$ cd <directory name>

Output

```
student@S46:~$ cd stardust
student@S46:~/stardust$ cd ..
```

- **cd.. :** This command is used to move to the parent directory of current directory, or the directory one level up from the current directory.
- **cd -:** This command is used to switch back to previous directory we were working earlier.

cat > filename: This command is used to create a file and add contents to that file.

Syntax :- \$ cat > filename.txt

cat filename: This command is used to view the contents in the file.

Syntax :- \$ cat filename.txt

Output

```
student@S46:~/stardust$ cat >a.txt
Nertwork is good
^Z
[1]+  Stopped                  cat > a.txt
```

cat>>filename: This command is used to add contents to an existing file.

Syntax :- \$ cat >> filename.txt

Output

```
student@S46:~/stardust$ cat>>a.txt
rlmca136
^Z
[2]+  Stopped                  cat >> a.txt
```

cat filename1 > filename2: This command is used to copy the content from one file to another file.

Syntax :- \$ cat filename1 > filename2

Output

```
student@S46:~/stardust$ cat a.txt > b.txt
student@S46:~/stardust$ cat b.txt
Nertwork is good
^Z
rlmca136
```

read : This command is used to read the content of a line to a variable.

Syntax :- \$ read variablename

Find: This command is used to display contents of particular directory.

Syntax :- \$ find filename.txt

grep : This command will let you search through all the text in a given file.

Syntax :- \$ grep word filename.txt

Output:-

- **grep -i :** command used for a case insensitive search

Syntax: \$ grep -i filename.txt

- **grep -v :** command used for inverted search.

Syntax: \$ grep -v filename.txt

- **grep -A1:** command used to display line after the result.

Syntax: \$ grep -A1 filename.txt

- **grep -B1:** command used to display line before the result.

Syntax: \$ grep -B1 filename.txt

- **grep -C1:** command used to display line before and after the result.

Syntax: `$ grep -C1 filename.txt`

wc -word count: This command is used for counting purpose which is used to find the number of lines, the number of words, the number of characters and the number of bytes.

- **wc -l** (count number of lines)
- **wc -w** (count number of words)
- **wc -c** (count number of characters)
- **wc -m** (count number of bytes)

Syntax :- `$ wc -l filename.txt`

`$ wc -w filename.txt`

`$ wc -c filename.txt`

`$ wc -m filename.txt`

Output



```
student@S3:~$ cat marvel1
captian america
  ironman
spiderman
hulk
xmen
strange
student@S3:~$ wc marvel1
 6  7 53 marvel1
```

```
student@S3:~$ wc -c marvel1
53 marvel1
```

```
student@S3:~$ wc -w marvel1
7 marvel1
```

```
student@S3:~$ wc -l marvel1
6 marvel1
```

```
student@S3:~$ wc -m marvel1
53 marvel1
```

df: This command is used to get a report on system disc space usage.

Syntax :- \$ df filename.txt

Output

```
student@S3:~$ df
Filesystem      1K-blocks      Used Available Use% Mounted on
udev            3989460         0   3989460   0% /dev
tmpfs           803792        1824    801968   1% /run
/dev/sda6       114460828 33493392  75110056  31% /
tmpfs           4018948       26024   3992924   1% /dev/shm
tmpfs           5120           4        5116   1% /run/lock
tmpfs           4018948         0   4018948   0% /sys/fs/cgroup
/dev/loop11     164096       164096         0 100% /snap/gnome-3-28-1804/116
/dev/loop17     144128       144128         0 100% /snap/gnome-3-26-1604/98
/dev/loop21     207872       207872         0 100% /snap/vlc/1397
/dev/loop15       640         640         0 100% /snap/gnome-logs/106
/dev/loop3       2688        2688         0 100% /snap/gnome-system-monitor/174
/dev/loop7       2560        2560         0 100% /snap/gnome-calculator/884
/dev/loop27      1024        1024         0 100% /snap/gnome-logs/81
/dev/loop2      144128       144128         0 100% /snap/gnome-3-26-1604/104
```

➤ **df -m**: This command is used to see the report in mega bytes.

Syntax : \$ df -m filename.txt

cut -d: This command is used to cut and display the content based on the delimiter given.

Syntax :- \$ cut -d delimiter -fieldnumber filename

```
student@S3:~$ cut -d- -f2 b3.txt
33
56
77
student@S3:~$ cut -d- -f1 b3.txt
english
hindi
maths
```

cut -b: This command is used to cut and display the content based on the specified byte number.

Syntax :- \$ cut -b bytenumber filename

Output

```
student@S3:~$ cut -b 2 mark1
n
a
c
```

cut --complement -c: This command is used to erase the specified character and display the remaining content of the file.

Syntax :- \$ cut --complement -c characternumber filename.txt

Output

```
student@S3:~$ cut --complement -c 1 mark1
nglish 67
aths 78
cience 90
```

Paste: This command is used to paste the contents from the specified file.

Syntax :- \$ paste filename

```
student@S3:~$ paste marvel1 marvel2
captian america black pink
ironman          bts
spiderman        batman
hulk             cartoon
xmen             ton
strange jerry
```

More: This command is used to view the text files in the command prompt, displaying one screen at a time in case the file is large.

Syntax :- \$ more filename

Cp: This command is used to copy the contents from an existing file to a new file.

Syntax :- \$ cp existing_filename new_filename

Output

```
student@S33:~$ ls
b3.txt  doc      Downloads  file1  flower  ls      Pictures  Public  sample  sandra  Templates
Desktop Documents examples.desktop  files  fruits  Music    plants   PycharmProjects  samples  snap    Videos
student@S33:~$ cat world
cat: world: No such file or directory
student@S33:~$ cat flower
lotus
jasmine
rose
marigold
student@S33:~$ cp flower flowerlist
student@S33:~$ cat flowerlist
lotus
jasmine
rose
marigold
```

Mv: This command is used to move an existing file or directory from one location to another.

Syntax :- \$ mv filename directory_name

Output

```
student@S3:~$ mv dq.txt akhila
student@S3:~$ cd akhila
student@S3:~/akhila$ ls
a.txt  b.txt  dq.txt
```

Head: This command is used to display the first 10 lines of the file by default.

Syntax :- \$ head filename

Output

```
student@S3:~$ head b1.txt
Familiarisation of cat command
Cat having different option
new file

adding content
appending
updating
qweerrfttf
adfgtttg
weryhbvf
student@S3:~$ head -4 b1.txt
Familiarisation of cat command
Cat having different option
new file
```


- **head -number:** This command is used to display the lines of the file to the specified number from head.

Tail: This command is used to display the last 10 lines of the file by default.

Syntax :- \$ tail filename

Output

```
student@S3:~$ tail b1.txt
new file

adding content
appending
updating
qweerfttf
adfgtttg
weryhbvf
zfsfg
ojkhh
student@S3:~$ tail -3 b1.txt
weryhbvf
zfsfg
ojkhh
```

- **tail -number:** This command is used to display the lines of the file to the specified number from tail.

sudo useradd : This command is used to add new user.

Syntax :- \$ sudo useradd username

Output

```
mca@S3:~$ sudo useradd Akhila
[sudo] password for mca:
Sorry, try again.
[sudo] password for mca:
```

- **sudo passwd :** This command is used to add password to the user.

Syntax :- \$ sudo passwd username

- **sudo usermod :** This command is used to add members.

Syntax :- \$ sudo usermod -G groupname username **delete**

- **sudo userdel username** - used to delete user.

- **sudo groupdel groupname** - used to delete group name.

Syntax :- \$ **sudo userdel username**

- **sudo groupdel groupname**

chmod : This command is used to change directory permission of files.

- **chmod +rwx**

- **chmod -wx**

- **chmod -rwx**

Syntax :- \$ **chmod +wx filename**

\$ **chmod -wx filename**

\$ **chmod -rwx filename**

Output

```
mca@S3:~$ ls
a1.txt Desktop Documents Downloads examples.desktop mozilla.pdf Music Pictures Public PycharmProjects
mca@S3:~$ chmod +rwx a1.txt
mca@S3:~$ chmod -wx a1.txt
mca@S3:~$ cat >>a1.txt
bash: a1.txt: Permission denied
mca@S3:~$ chmod -rwx a1.txt
mca@S3:~$ cat a1.txt
cat: a1.txt: Permission denied
```

chown: This command is used to give ownership to user .

Syntax :- \$ **sudo chown username filename**

Output

```
mca@S3:~$ sudo useradd Anjali
mca@S3:~$ sudo chown Anjali a1.txt
```

Ssh: This command is used to provide a secure encrypted connection between two hosts over an insecure network.

Syntax :- \$ ssh mca@ipaddress

```
mca@S40:~$ sudo ssh mca@192.168.6.46
The authenticity of host '192.168.6.46 (192.168.6.46)' can't be established.
ECDSA key fingerprint is SHA256:hQC0bgw7WBI7zuABHq2AKWIpGnXDeBBGWGvJqDHDPNY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.6.46' (ECDSA) to the list of known hosts.
mca@192.168.6.46's password:
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.15.0-23-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

8 packages can be updated.
0 updates are security updates.

Last login: Mon Apr 25 15:48:44 2022 from 192.168.6.63
mca@S46:~$
```

4. Shell scripting: study bash syntax, environment variables, variables, control constructs such as if, for and while, aliases and functions, accessing command line arguments passed to shell scripts.

Source code &Output

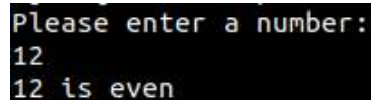
Program 1

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

```
check_odd_even() {  
    if [ $((number % 2)) -eq 0 ]; then  
        echo "$number is even"  
    else  
        echo "$number is odd"  
    fi  
}
```

```
echo "Please enter a number: "  
read number
```

```
check_odd_even "$number"
```



```
Please enter a number:  
12  
12 is even
```

Program 2

Write a Shell program to check a leap year.

```
echo "Enter the year (YYYY)"  
read year  
if [ $((year % 4)) -eq 0 ]  
then  
    if [ $((year % 100)) -eq 0 ]  
    then  
        if [ $((year % 400)) -eq 0 ]  
        then  
            echo "$year is a leap year"  
        else  
            echo "$year is not a leap year"
```

```
fi
else
echo "$year is a leap year"
fi
else
echo "$year is not a leap year"
fi
```

```
sjcet@Z238-UL:~/kishor/sem2/CN$ bash 3.sh
Enter the year (YYYY)
2022
2022 is not a leap year
```

Program 3

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

```
echo "Enter the radius:"
read r
area=`echo 3.14 \* $r \* $r | bc`
cir=`echo 2 \* 3.14 \* $r | bc`
echo "Area : $area"
echo "Circumference : $cir"
```

```
Enter the radius:
5
Area : 78.50
Circumference : 31.40
```

Program 4

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

```
read num
reverse=$(echo "$num" | rev)
if [ "$num" -eq "$reverse" ]; then
echo "$num is same when reversed."
else
echo "$num is not same when reversed."
```

fi

```
Enter a number:
12345
12345 is not same when reversed.
```

Program 5

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

```
echo Enter the string
read s
echo $s>temp
rvs="$(rev temp)"
if [ $s = $rvs ]
then
echo "it is palindrome"
else
echo " it is not a Palindrome"
fi
```

```
Enter the string
malayalam
it is palindrome
```

Program 6

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

```
echo "Enter a set of numbers separated by spaces: "
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 is: $sum_even"
echo "Sum of odd numbers is: $sum_odd"

```

```

sjcet@Z238-UL:~/kishor/sem2/CN$ bash 7.sh
Enter a set of numbers separated by spaces:
1 2 3 4 5 6 7
Sum of even numbers is: 12
Sum of odd numbers is: 16

```

Program 7

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

```

echo "Enter the coefficients of the quadratic equation (a, b, c): "
read a b c
discriminant=$((b*b - 4*a*c))
if [ $discriminant -lt 0 ]
then
    echo "The quadratic equation has no real roots."
else

    root1=$(echo "scale=2; (-$b + sqrt($discriminant)) / (2*$a)" | bc)
    root2=$(echo "scale=2; (-$b - sqrt($discriminant)) / (2*$a)" | bc)

    echo "The roots of the quadratic equation are: $root1 and $root2"
fi

```

```

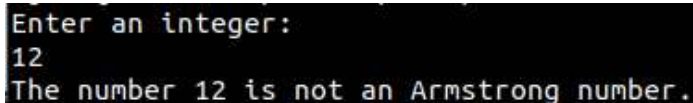
Enter the coefficients of the quadratic equation (a, b, c):
1 2 3
The quadratic equation has no real roots.

```

Program 8

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

```
echo "Enter an integer: "
read number
count=${#number}
sum=0
for (( i=0; i<count; i++ ))
do
    digit=${number:i:1}
    sum=$((sum + digit**count))
done
if [ "$sum" -eq "$number" ]
then
    echo "The number $number is an Armstrong number."
else
    echo "The number $number is not an Armstrong number."
fi
```



```
Enter an integer:
12
The number 12 is not an Armstrong number.
```

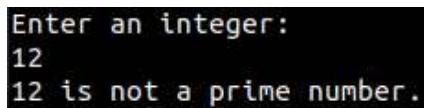
Program 9

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

```
echo "Enter an integer: "
read number
flag=1
for (( i=2; i<=number/2; i++ ))
do
    if [ $((number%i)) -eq 0 ]
    then
        flag=0
        break
    fi
```



```
done
if [ $number -eq 1 ]
then
    echo "1 is neither prime nor composite."
elif [ $flag -eq 1 ]
then
    echo "$number is a prime number."
else
    echo "$number is not a prime number."
fi
```



```
Enter an integer:
12
12 is not a prime number.
```

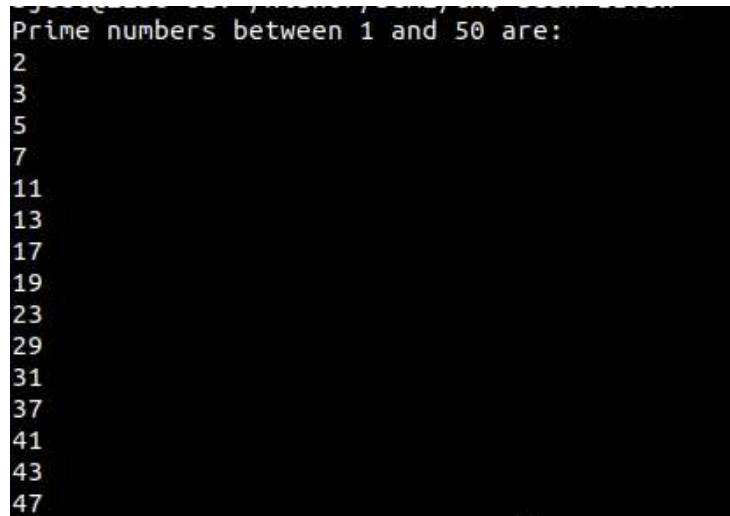
Program 10

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

```
echo "Prime numbers between 1 and 50 are:"
for (( number=2; number<=50; number++ ))
do
    flag=1

    for (( i=2; i<=number/2; i++ ))
    do
        if [ $((number%i)) -eq 0 ]
        then
            flag=0
            break
        fi
    done
    if [ $flag -eq 1 ]
    then
        echo $number
    fi
```

Done


A terminal window with a black background and white text. The text reads: "Prime numbers between 1 and 50 are:" followed by a list of prime numbers: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, and 47.

```
Prime numbers between 1 and 50 are:
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
```

Program 11

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

```
echo "Enter a number: "
read number
sum=0
while [ $number -ne 0 ]
do
    digit=$((number % 10))
    sum=$((sum + digit * digit))
    number=$((number / 10))
done
echo "The sum of the squares of the digits is $sum."
```

A terminal window with a black background and white text. The text shows the user inputting '123' and the program outputting 'The sum of the squares of the digits is 14.'.

```
Enter a number:
123
The sum of the squares of the digits is 14.
```

Program 12

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

```
echo "Enter a line of text:"
read line
```

```
vowel_count=0
for (( i=0; i<${#line}; i++ )); do
    char=${line:i:1}
    case $char in
        [aeiouAEIOU])
            vowel_count=$((vowel_count + 1))
            ;;
    esac
done
echo "The number of vowels in the line of text is: $vowel_count"
```

```
Enter a line of text:
hi welcome to shell
The number of vowels in the line of text is: 6
```

Program 13

Write a Shell program to display student grades.

```
INPUT_FILE="grades.txt"
if [[ ! -f "$INPUT_FILE" ]]; then
    echo "Input file not found!"
    exit 1
fi
declare -A grades
while read line; do
    name=$(echo "$line" | cut -d ',' -f 1)
    grade=$(echo "$line" | cut -d ',' -f 2)
    grades["$name"]=$grade
done < "$INPUT_FILE"
for name in "${!grades[@]}"; do
    echo "$name: ${grades[$name]}"
done
```

```
Alice: 90%  
Emma: 50%  
Charlie: 70%  
David: 60%  
Bob: 80%
```

Program 14

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

```
NUMBERS=(5 3 8 1 9 4 7 2)  
smallest=${NUMBERS[0]}  
largest=${NUMBERS[0]}  
for number in "${NUMBERS[@]"; do  
    if (( number < smallest )); then  
        smallest=$number  
    fi  
    if (( number > largest )); then  
        largest=$number  
    fi  
done  
echo "Smallest number: $smallest"  
echo "Largest number: $largest"
```

```
sjcet@Z238-UL:~/kishor/sem2/CN$ bash 15.sh  
Smallest number: 1  
Largest number: 9
```

Program 15

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

```
echo "Enter a number:"  
read number  
smallest=${number:0:1}  
for (( i=1; i<${#number}; i++ )); do  
    digit=${number:i:1}  
    if (( digit < smallest )); then
```

```

    smallest=$digit
fi
done
echo "Smallest digit: $smallest"

```

```

Enter a number:
12345
Smallest digit: 1

```

Program 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.

```

sum=0
for (( i=50; i<=100; i++ )); do
    if (( i % 3 == 0 )) && (( i % 5 != 0 )); then
        sum=$(( sum + i ))
    fi
done
echo "Sum of numbers between 50 and 100, which are divisible by 3 and not divisible by 5:
$sum"

```

```

sjcet@Z238-UL:~/kishor/sem2/CN$ bash 17.sh
Sum of numbers between 50 and 100, which are divisible by 3 and not divisible by 5: 1050
sjcet@Z238-UL:~/kishor/sem2/CN$ █

```

Program 17

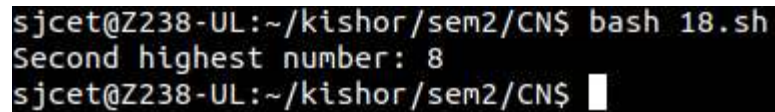
Write a Shell program to find the second highest number from a set of numbers.

```

NUMBERS=(5 3 8 1 9 4 7 2)
highest=${NUMBERS[0]}
second_highest=${NUMBERS[0]}
for number in "${NUMBERS[@]}"; do
    if (( number > highest )); then
        second_highest=$highest
        highest=$number
    elif (( number != highest )) && (( number > second_highest )); then

```

```
second_highest=$number
fi
done
echo "Second highest number: $second_highest"
```

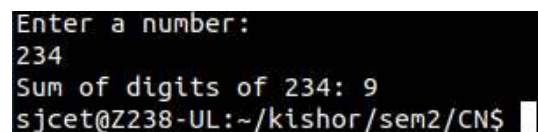


```
sjcet@Z238-UL:~/kishor/sem2/CN$ bash 18.sh
Second highest number: 8
sjcet@Z238-UL:~/kishor/sem2/CN$
```

Program 18

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

```
function sum_of_digits {
    local number=$1
    local sum=0
    while (( number > 0 )); do
        sum=$(( sum + number % 10 ))
        number=$(( number / 10 ))
    done
    echo "$sum"
}
echo "Enter a number:"
read number
result=$(sum_of_digits $number)
echo "Sum of digits of $number: $result"
```

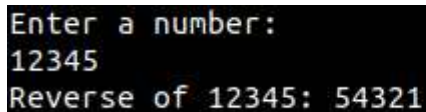


```
Enter a number:
234
Sum of digits of 234: 9
sjcet@Z238-UL:~/kishor/sem2/CN$
```

Program 19

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

```
function reverse_number {  
    local number=$1  
    local reverse=0  
    while (( number > 0 )); do  
        reverse=$(( reverse * 10 + number % 10 ))  
        number=$(( number / 10 ))  
    done  
    echo "$reverse"  
}  
echo "Enter a number:"  
read number  
result=$(reverse_number $number)  
echo "Reverse of $number: $result"
```




```
Enter a number:  
12345  
Reverse of 12345: 54321
```

Program 20

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

```
echo "Enter a number:"  
read number  
factorial=1  
for (( i=1; i<=number; i++ )); do  
    factorial=$(( factorial * i ))  
done  
echo "Factorial of $number: $factorial"
```

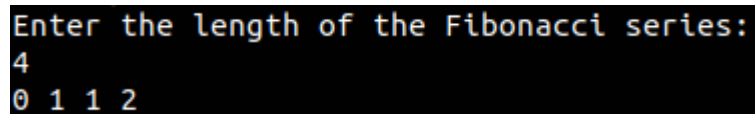


```
Enter a number:  
5  
Factorial of 5: 120
```

Program 21

Write a Shell program to generate Fibonacci series.

```
echo "Enter the length of the Fibonacci series:"
read length
num1=0
num2=1
echo -n "$num1 $num2 "
for (( i=2; i<length; i++ )); do
    next=$(( num1 + num2 ))
    echo -n "$next "
    num1=num2
    num2=next
done
echo ""
```



```
Enter the length of the Fibonacci series:
4
0 1 1 2
```

Program 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.

```
#!/bin/bash

if [ $# -ne 2 ]; then
    echo "Usage: $0 file1 file2"
    exit 1
fi

if cmp -s "$1" "$2"; then
    echo "The contents of $1 and $2 are the same. Deleting $2..."
    rm "$2"
```



```
else
    echo "The contents of $1 and $2 are different."
Fi
```

```
sjcet@Z238-UL:~/kishor/sem2/CN$ bash 23.sh file1.txt file2.txt
The contents of file1.txt and file2.txt are the same. Deleting file2.txt...
sjcet@Z238-UL:~/kishor/sem2/CN$
```

Program 23

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

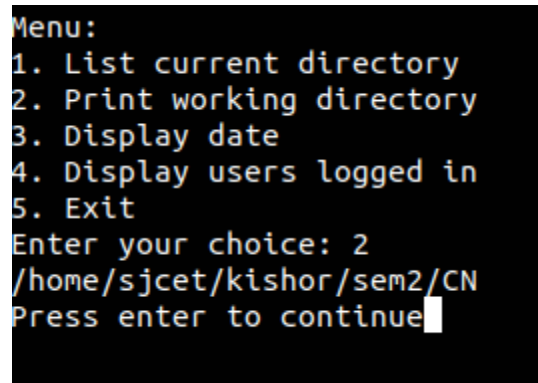
```
#!/bin/bash
while true
do
    clear
    echo "Menu:"
    echo "1. List current directory"
    echo "2. Print working directory"
    echo "3. Display date"
    echo "4. Display users logged in"
    echo "5. Exit"

    read -p "Enter your choice: " choice
    case $choice in
        1)
            ls -l
            read -p "Press enter to continue"
            ;;
        2)
            pwd
            read -p "Press enter to continue"
            ;;
        3)
```

```

    date
    read -p "Press enter to continue"
    ;;
4)
    who
    read -p "Press enter to continue"
    ;;
5)
    exit 0
    ;;
*)
    echo "Invalid choice. Press enter to try again"
    read
    ;;
esac
done

```



```

Menu:
1. List current directory
2. Print working directory
3. Display date
4. Display users logged in
5. Exit
Enter your choice: 2
/home/sjcet/kishor/sem2/CN
Press enter to continue

```

Program 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.

```

for file in *; do
    if [[ ! -x "$file" ]]; then
        chmod +x "$file"
        echo "Made $file executable"
    fi
done

```

done

```
sjcet@Z238-UL:~/kishor/sem2/CN$ bash 25.sh
Made 10.sh executable
Made 11.sh executable
Made 12.sh executable
Made 13.sh executable
Made 14.sh executable
Made 15.sh executable
Made 16.sh executable
Made 17.sh executable
Made 18.sh executable
Made 19.sh executable
Made 20.sh executable
Made 21.sh executable
Made 22.sh executable
Made 23.sh executable
Made 24.sh executable
Made 25.sh executable
Made 26.sh executable
Made 27.sh executable
Made 28.sh executable
Made 29.sh executable
Made 2.sh executable
Made 30.sh executable
Made 31.sh executable
Made 32.sh executable
Made 33.sh executable
Made 34.sh executable
Made 3.sh executable
```

Program 25

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

```
#!/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
```

```
1 1 1
1 1 2
1 1 3
1 2 1
1 2 2
1 2 3
1 3 1
1 3 2
1 3 3
2 1 1
2 1 2
2 1 3
2 2 1
2 2 2
2 2 3
2 3 1
2 3 2
2 3 3
3 1 1
3 1 2
3 1 3
3 2 1
3 2 2
3 2 3
3 3 1
3 3 2
3 3 3
```

Program 26

Write a Shell program to create the number series.

1

2 3

4 5 6

7 8 9 10

```
#!/bin/bash
```

```
num=1
```

```
row=1
```

```
while [ $row -le 4 ]; do
```

```
  for (( i=1; i<=$row; i++ )); do
```

```
    echo -n "$num "
```

```
    num=$((num+1))
```

```
  done
```

```

echo ""
row=$((row+1))
done

```

```

1
2 3
4 5 6
7 8 9 10

```

Program 27

Write a Shell program to create Pascal's triangle.

```

#!/bin/bash
function binom {
    if [ $2 -eq 0 ] || [ $2 -eq $1 ]; then
        echo 1
    else
        echo $(( $(binom $(( $1 - 1 )) $(( $2 - 1 ))) + $(binom $(( $1 - 1 )) $2) ))
    fi
}
echo "Enter the number of rows in Pascal's triangle: "
read rows
for (( i=0; i<$rows; i++ )); do
    for (( j=0; j<=$i; j++ )); do
        val=$(binom $i $j)
        echo -n "$val "
    done
    # Move to next row
    echo ""
done

```

```

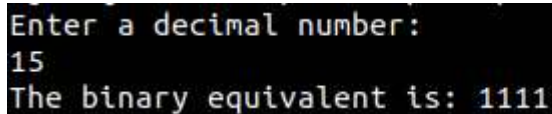
Enter the number of rows in Pascal's triangle:
3
1
1 1
1 2 1

```

Program 28

Write a Decimal to Binary Conversion Shell Script

```
#!/bin/bash
# Prompt the user for the decimal number to convert
echo "Enter a decimal number: "
read decimal
# Convert the decimal number to binary
binary=""
while [ $decimal -gt 0 ]; do
    remainder=$((decimal % 2))
    binary="$remainder$binary"
    decimal=$((decimal / 2))
done
# Print the binary number
echo "The binary equivalent is: $binary"
```



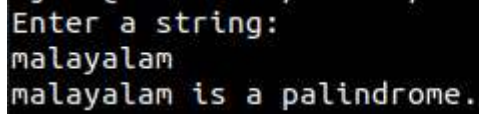
```
Enter a decimal number:
15
The binary equivalent is: 1111
```

Program 29

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

```
#!/bin/bash
# Prompt the user for the string to check
echo "Enter a string: "
read string
# Reverse the string
reverse=$(echo $string | rev)
# Check if the string is equal to its reverse
if [ "$string" == "$reverse" ]; then
    echo "$string is a palindrome."
```

```
else
    echo "$string is not a palindrome."
fi
```



```
Enter a string:
malayalam
malayalam is a palindrome.
```

Program 30

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

```
#!/bin/bash

# Prompt the user for the file name
echo "Enter the file name: "
read file

# Check if the file exists
if [ ! -f "$file" ]; then
    echo "File not found."
    exit 1
fi

# Convert the contents of the file to lowercase and replace all non-alphanumeric characters
# with spaces
contents=$(tr '[:upper:]' '[:lower:]' < $file | sed 's/[^a-z0-9]/ /g')

# Create an array of words from the file contents
words=( $contents )

# Loop through the array of words and count their occurrences
declare -A count
for word in "${words[@]"; do
    if [ -n "$word" ]; then
        ((count[$word]++))
    fi
done

# Print the unique words and their counts
echo "Unique words in $file:"
```

```
for word in "${!count[@]}"; do
    echo "$word: ${count[$word]}"
done
```

```
Enter the file name:
file1.txt
Unique words in file1.txt:
linux: 2
```

Program 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.

```
#!/bin/bash
# Set the search directory
search_dir="."
# Find all ".txt" files in the search directory and its subdirectories
files=$(find "$search_dir" -type f -name "*.txt")
# Initialize the count
count=0
# Loop through each file and count the occurrences of "Linux"
for file in $files; do
    occurrences=$(grep -o "Linux" "$file" | wc -l)
    count=$((count + occurrences))
done

# Print the total count
echo "Total count of 'Linux' in all .txt files: $count"
```

```
sjcet@Z238-UL:~/kishor/sem2/CN$ bash 32.sh
Total count of 'Linux' in all .txt files: 2
```

Program 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.

```
#!/bin/bash
```

```
read -p "Enter your password: " password
```

```
# Check if password is at least 8 characters long
```

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

```
    echo "Password length must be at least 8 characters."
```

```
    exit 1
```

```
fi
```

```
# Check if password contains both alphabet and number
```

```
if ! [[ "$password" =~ [A-Za-z]+[0-9]+ ]]; then
```

```
    echo "Password must contain both alphabet and number."
```

```
    exit 1
```

```
fi
```

```
# Check if password includes both small and capital case letters
```

```
if ! [[ "$password" =~ [a-z]+ ]] || ! [[ "$password" =~ [A-Z]+ ]]; then
```

```
    echo "Password must include both small and capital case letters."
```

```
    exit 1
```

```
fi
```

```
echo "Password is valid."
```

```
sjcet@Z238-UL:~/kishor/sem2/CN$ bash 33.sh
Enter your password: Kishor245@
Password is valid.
```

Program 33

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

```
echo "Enter directory path: "
```

```
read directory
```

```
num_files=$(find $directory -type f | wc -l)
```

```
num_directories=$(find $directory -type d | wc -l)
```

```
echo "Number of files: $num_files"
echo "Number of directories: $num_directories"
```

```
Enter directory path:
/home/sjcet/kishor/sem2/CN
Number of files: 119
Number of directories: 2
```

Program 34

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

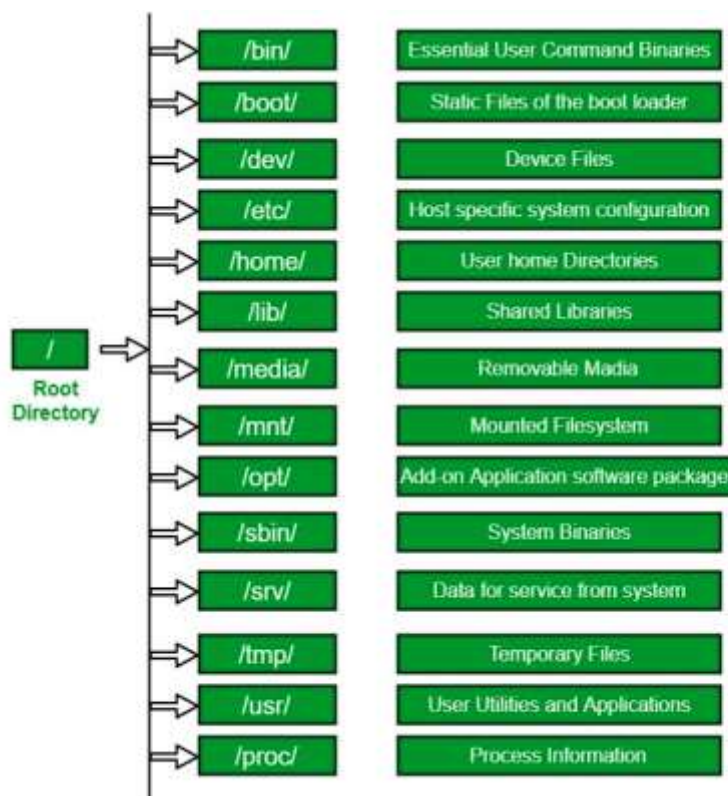
```
#!/bin/bash
my_list=("string1" "string2" "string3" "string4")
# Reverse the order of the list
my_list=($(echo "${my_list[@]}" | tr ' ' '\n' | tac | tr '\n' ' '))
# Reverse each string in the list
for i in "${!my_list[@]}"
do
    my_list[$i]=`echo ${my_list[$i]} | rev`
done
# Print the reversed list of strings
echo "${my_list[@]}"
```

```
sjcet@Z238-UL:~/kishor/sem2/CN$ bash 35.sh
4gnirts 3gnirts 2gnirts 1gnirts
```

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 is 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

```
mca@53:~$ sudo apt update
Hit:1 http://packages.microsoft.com/repos/vscode stable InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu bionic InRelease
Err:3 http://ppa.launchpad.net/jonathonf/python-3.6/ubuntu bionic InRelease
 403 Forbidden [IP: 185.125.190.52 80]
Ign:4 https://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.6 InRelease
Hit:5 http://ppa.launchpad.net/webupd8team/java/ubuntu bionic InRelease
Get:6 https://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.6 Release [2,495 B]
Get:7 https://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.6 Release.gpg [801 B]
Err:7 https://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.6 Release.gpg
The following signatures were invalid: EXPKEYSIG 59743122815111B5 Ubuntu 18.04 LTS
```

- Install Apache using apt:

sudo apt install apache2

```
mca@53:~$ sudo apt install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
apache2 is already the newest version (2.4.29-1ubuntu4).
The following packages were automatically installed and are no longer required:
 debhelper dh-autoreconf dh-strip-nondeterminism libarchive-cpio-perl libfile-stripnondeterminism-perl libmail-sendmail-perl libpcre16-3
 libpcre3-dev libpcre32-3 libpcrecpp0v5 libssl-dev libssl-doc libsys-hostname-long-perl php-common php-pear php-xml php7.2-cli
 php7.2-common php7.2-json php7.2-opcache php7.2-readline php7.2-xml pkg-php-tools po-debconf shtool
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 8 not upgraded.
```

- Confirm that Apache is now running with the following command:

sudo systemctl status apache2

- If it is not working !

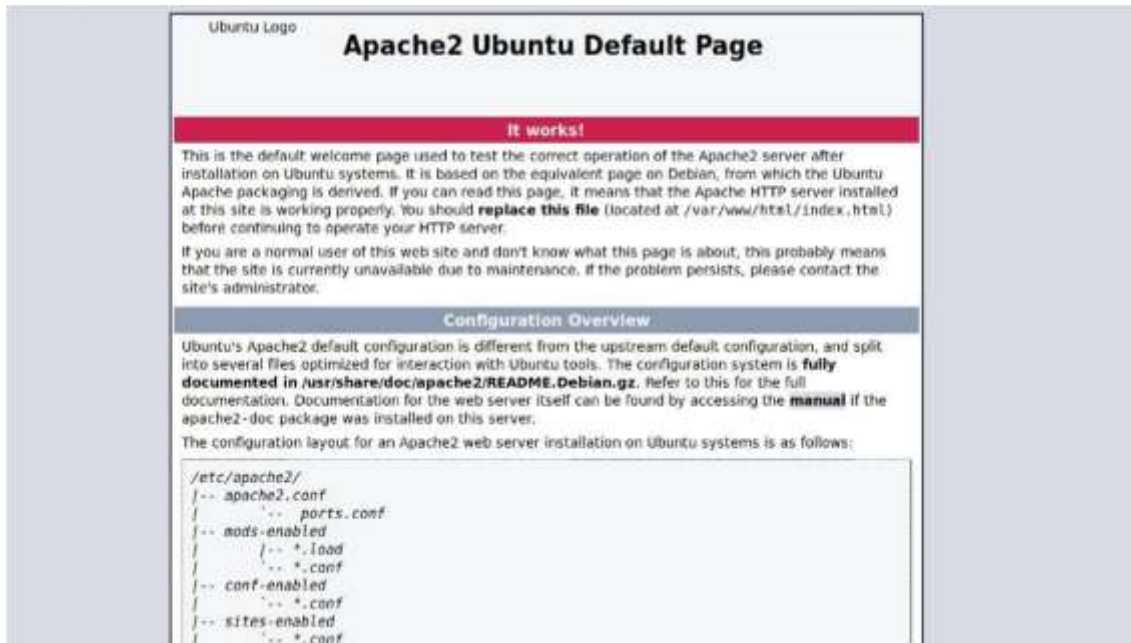
sudo systemctl stop apache2 # to stop if running

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 mysql # 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** Open a browser and type in your IP address/phpinfo.php

`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`
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`**

```
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`

```
mca@S3:~$ wget -c http://wordpress.org/latest.tar.gz
--2022-06-13 15:24:19-- http://wordpress.org/latest.tar.gz
Resolving wordpress.org (wordpress.org)... 198.143.164.252
Connecting to wordpress.org (wordpress.org)[198.143.164.252]:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://wordpress.org/latest.tar.gz [following]
--2022-06-13 15:24:20-- https://wordpress.org/latest.tar.gz
Connecting to wordpress.org (wordpress.org)[198.143.164.252]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 21166276 (20M) [application/octet-stream]
Saving to: 'latest.tar.gz'

latest.tar.gz                               100%[=====]
2022-06-13 15:24:24 (5.98 MB/s) - 'latest.tar.gz' saved [21166276/21166276]
```

- tar -xzf latest.tar.gz

```
mca@S3:~$ tar -xzf latest.tar.gz
wordpress/
wordpress/xmlrpc.php
wordpress/wp-blog-header.php
wordpress/readme.html
wordpress/wp-signup.php
wordpress/index.php
wordpress/wp-cron.php
wordpress/wp-config-sample.php
wordpress/wp-login.php
wordpress/wp-settings.php
wordpress/license.txt
wordpress/wp-content/
wordpress/wp-content/themes/
wordpress/wp-content/themes/twentytwentyone/
wordpress/wp-content/themes/twentytwentyone/footer.php
wordpress/wp-content/themes/twentytwentyone/template-parts/
```

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 account when you installed the software. However, if you have password authentication enabled 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 guide as an example. Create the database for WordPress by writing the following:

➤ `CREATE DATABASE wordpress DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;`

```
mysql> CREATE DATABASE wordpress DEFAULT CHARACTER SET utf8
-> COLLATE utf8_unicode_ci;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| wordpress |
+-----+
5 rows in set (0.00 sec)
```

Next, you're going to create a separate MySQL user account that you'll use exclusively to operate on the new database. Creating one-function databases and accounts is a good idea from a management and security standpoint. We will use the name `wordpressuser` as an example in this guide. Feel free to change this if you'd like. You can create this account, set a password for it, and then grant it access to the database you created all by running the following command. Remember to choose 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 of MySQL 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

```
mca@S3:~$ cd /var/www/html/
```

```
mca@S3:/var/www/html$ sudo mv wp-config-sample.php wp-config.php
```

```
mca@S3:/var/www/html$ 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

```
mca@S46:/var/www/html$ sudo systemctl restart apache2.service  
mca@S46:/var/www/html$ sudo systemctl restart mysql.service
```

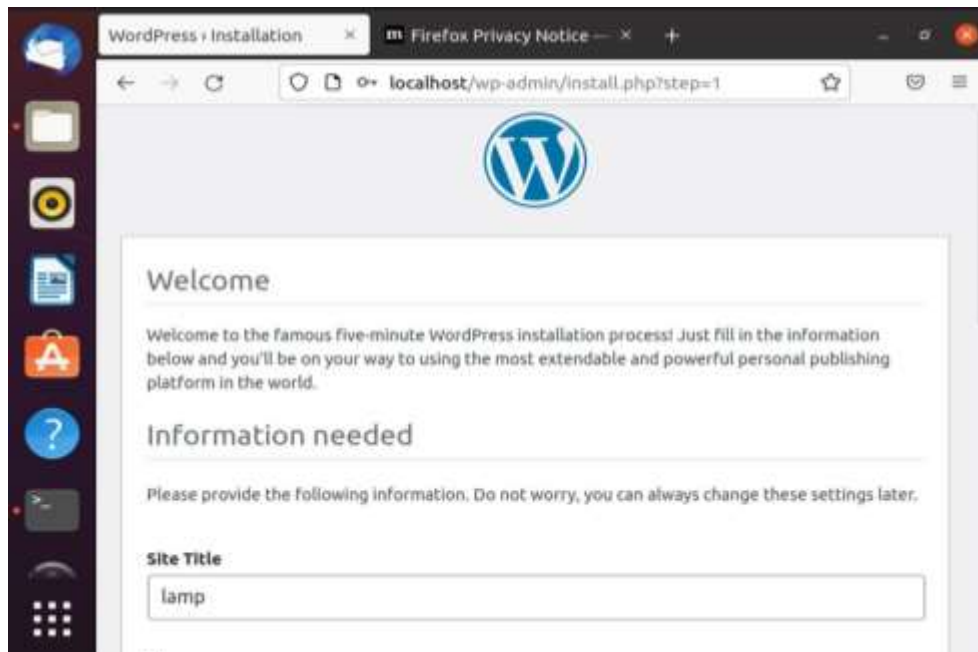
Step 3 – Completing the Installation Through the Web Interface

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:

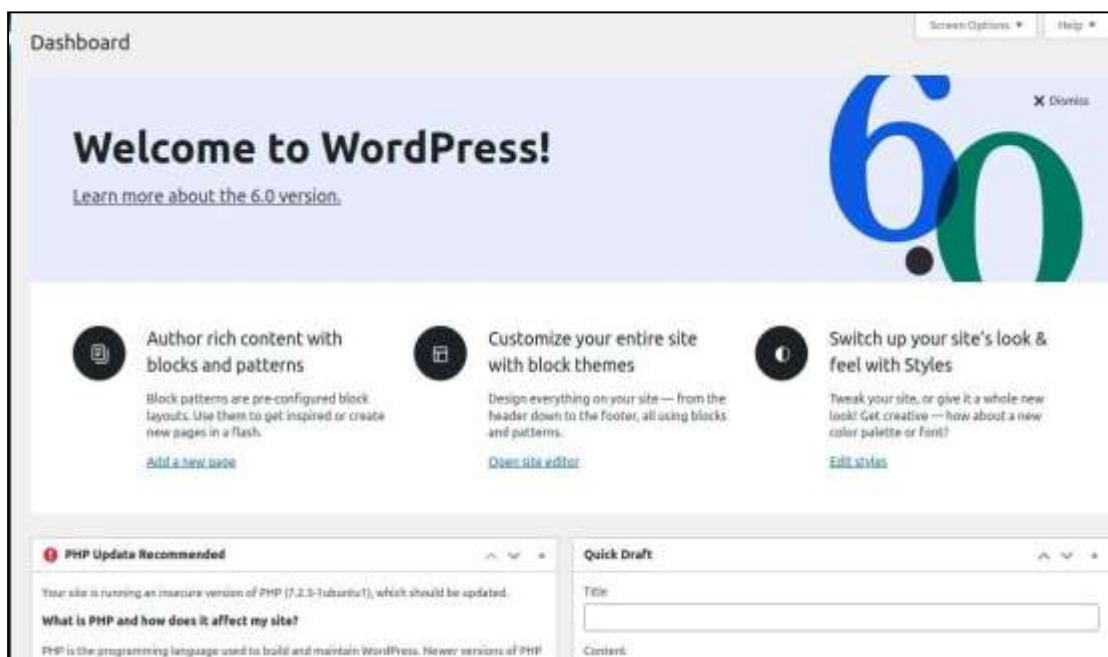
- https://server_domain_or_IP

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

```
mca@S3:~/Documents/CMake$ apt show cmake
Package: cmake
Version: 3.10.2-1ubuntu2
Priority: optional
Section: devel
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Debian CMake Team <pkg-cmake-team@lists.alioth.debian.org>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 17.3 MB
Depends: cmake-data (= 3.10.2-1ubuntu2), procps, libarchive13 (>= 3.0.4), libc6 (>= 2.15), libcurl4 (>= 7.16.2), libexpat1 (>= 2.0.1), libgcc1 (>= 1:3.0), libjsoncpp1 (>= 1.7.4), librhash0 (>= 1.2.6), libstdc++6 (>= 5.2), libuv1 (>= 1.4.2), zlib1g (>= 1:1.2.3.3)
Recommends: gcc, make
Suggests: cmake-doc, ninja-build
Homepage: https://cmake.org/
Supported: 5y
Download-Size: 3.13B kB
```

- **\$sudo apt install cmake g++ make:** To install cmake , g++ and make using the apt command.

```
mca@S3:~/Documents/CMake$ sudo apt install cmake g++ make
[sudo] password for mca:
Reading package lists... Done
Building dependency tree
Reading state information... Done
g++ is already the newest version (4:7.3.0-3ubuntu2).
make is already the newest version (4.1-9.1ubuntu1).
make set to manually installed.
The following packages were automatically installed and are no longer required:
debhelper dh-autoreconf dh-strip-nondeterminism libarchive-cpio-perl
libfile-stripnondeterminism-perl libmail-sendmail-perl libpcre16-3
libpcre3-dev libpcre32-3 libpcrecpp0v5 libssl-dev libssl-doc
libsys-hostname-long-perl php-common php-pear php-xml php7.2-cli
php7.2-common php7.2-json php7.2-openssl php7.2-readline php7.2-xml
pkg-php-tools po-debconf shtool
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
cmake-data libcurl4 libjsoncpp1 librhash0 libuv1
Suggested packages:
```

Create directory

- **Mkdir cmake:** creating a different directory for our project using the mkdir and cd commands.

```
mca@S3:~/Documents/CMake$ mkdir myproject
```


➤ **Cd cmake**

```
mca@S3:~/Documents/CMake$ cd myproject
```

➤ **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 ..**

```
mca@S3:~/Documents/CMake/myproject/build$ cmake ..
-- The C compiler identification is GNU 7.3.0
-- The CXX compiler identification is GNU 7.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/mca/Documents/CMake/myproject/build
```

➤ **Cmake --build :** To generate the executable simply by typing: **run hello**

```
mca@S3:~/Documents/CMake/myproject/build$ cmake --build .
Scanning dependencies of target hello
[ 50%] Building CXX object CMakeFiles/hello.dir/Hello_world.cpp.o
[100%] Linking CXX executable hello
[100%] Built target hello
```

➤ **./hello:** Run the executable by typing:

```
mca@S3:~/Documents/CMake/myproject/build$ ./hello
Hello World!
```

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

1. ifconfig: This command in windows allows you to see a summarized information of your network such as ip address, subnet mask, server address etc.

Output

```

sjcet@Z238-UL:~/kishor/sem2/CN$ ifconfig
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.16.54.197 netmask 255.255.240.0 broadcast 172.16.63.255
    inet6 fe80::a4fb:b17f:9247:b333 prefixlen 64 scopeid 0x20<link>
    ether 3c:52:82:6e:b2:a7 txqueuelen 1000 (Ethernet)
    RX packets 366691 bytes 298946338 (298.9 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 188619 bytes 49238714 (49.2 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 16 memory 0xd1000000-d1020000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 5842 bytes 546463 (546.4 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 5842 bytes 546463 (546.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

2. 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

```

sjcet@Z238-UL:~/kishor/sem2/CN$ nslookup google.com
Server:      127.0.0.53
Address:     127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.193.142
Name:   google.com
Address: 2404:6800:4007:81f::200e

```

3. 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(TTL) of the packet, that is the amount of time that must pass before the packet discarded.

```
sjcet@Z238-UL:~/kishor/sem2/CN$ ping -c 4 google.com
PING google.com (142.250.196.46) 56(84) bytes of data.
64 bytes from maa03s45-in-f14.1e100.net (142.250.196.46): icmp_seq=1 ttl=59 time
=70.9 ms
64 bytes from maa03s45-in-f14.1e100.net (142.250.196.46): icmp_seq=2 ttl=59 time
=71.8 ms
64 bytes from maa03s45-in-f14.1e100.net (142.250.196.46): icmp_seq=3 ttl=59 time
=73.5 ms
64 bytes from maa03s45-in-f14.1e100.net (142.250.196.46): icmp_seq=4 ttl=59 time
=71.3 ms

--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 70.942/71.885/73.494/0.972 ms
sjcet@Z238-UL:~/kishor/sem2/CN$
```

4. 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

```
sjcet@Z238-UL:~/kishor/sem2/CN$ traceroute google.com
traceroute to google.com (142.250.195.46), 30 hops max, 60 byte packets
 1 _gateway (172.16.48.2) 0.250 ms 0.197 ms 0.153 ms
 2 172.24.71.66 (172.24.71.66) 1.801 ms 1.868 ms 1.827 ms
 3 * * *
 4 * * *
 5 72.14.218.250 (72.14.218.250) 39.904 ms 40.840 ms *
 6 * * *
 7 142.251.55.74 (142.251.55.74) 40.436 ms * *
 8 142.251.55.67 (142.251.55.67) 33.950 ms 74.125.242.138 (74.125.242.138) 18
.648 ms 74.125.242.147 (74.125.242.147) 32.593 ms
 9 108.170.253.113 (108.170.253.113) 39.160 ms maa03s37-in-f14.1e100.net (142.
250.195.46) 20.368 ms 108.170.253.113 (108.170.253.113) 38.347 ms
sicet@Z238-UL:~/kishor/sem2/CN$
```

5. 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.


```

sjcet@Z238-UL:~/kishor/sem2/CN$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN
tcp        0      0 localhost:ipp            0.0.0.0:*               LISTEN
tcp        0      0 localhost:33060         0.0.0.0:*               LISTEN
tcp        0      0 localhost:mysql         0.0.0.0:*               LISTEN
tcp        0      0 Z238-UL:50362          sd-ln-f188.1e100.n:5228 ESTABLISHED
tcp        0      0 Z238-UL:41866          maa03s46-ln-f10.1:https ESTABLISHED
tcp        0      0 Z238-UL:56220          maa03s42-ln-f3.1e:https ESTABLISHED
tcp        0      0 Z238-UL:45476          maa05s18-ln-f10.1:https ESTABLISHED
tcp        0      0 Z238-UL:38946          162.247.243.29:https ESTABLISHED
tcp        0      0 Z238-UL:39036          maa03s34-ln-f10.1:https ESTABLISHED
tcp        0      0 Z238-UL:35500          maa05s09-ln-f3.1e:https ESTABLISHED
tcp        0      0 Z238-UL:52014          104.18.2.161:https ESTABLISHED
tcp        0      0 Z238-UL:33820          a184-51-195-169.d:https ESTABLISHED
tcp        0 29587 Z238-UL:44876          maa05s17-ln-f14.1:https ESTABLISHED
tcp        0      0 Z238-UL:42126          maa05s28-ln-f14.1:https ESTABLISHED
tcp        0      0 Z238-UL:35788          maa05s09-ln-f14.1:https ESTABLISHED
tcp        0      0 Z238-UL:56332          84.170.224.35.bc.g:http TIME_WAIT
tcp        0      1 Z238-UL:37450          maa05s12-ln-f4.1e:https LAST_ACK
tcp        0      0 Z238-UL:47766          maa03s26-ln-f3.1e:https ESTABLISHED
tcp        0      0 Z238-UL:40350          maa03s46-ln-f14.1:https TIME_WAIT
tcp        0      0 Z238-UL:46120          maa03s42-ln-f3.1e:https ESTABLISHED
tcp        0      0 Z238-UL:45488          maa05s18-ln-f10.1:https ESTABLISHED
tcp        0      0 Z238-UL:58958          li781-4.members.l:https ESTABLISHED
tcp        0      1 Z238-UL:57628          maa05s20-ln-f4.1e:https LAST_ACK
tcp        0      0 Z238-UL:40184          maa05s17-ln-f13.1:https TIME_WAIT
tcp        0      0 Z238-UL:53738          172.16.208.2:8090 ESTABLISHED
tcp        0      0 Z238-UL:52980          li695-222.members:https ESTABLISHED
tcp        0      0 Z238-UL:49846          maa05s25-ln-f3.1e:https ESTABLISHED
tcp        0      0 Z238-UL:60996          ec2-35-174-127-31:https ESTABLISHED
tcp        0      0 Z238-UL:44324          maa05s19-ln-f14.1:https ESTABLISHED
tcp6       0      0 ip6-localhost:ipp      [::]:*                  LISTEN
udp        0      0 224.0.0.251:mdns       0.0.0.0:*               *
udp        0      0 224.0.0.251:mdns       0.0.0.0:*               *
udp        0      0 0.0.0.0:mdns           0.0.0.0:*               *
udp        0      0 0.0.0.0:42940         0.0.0.0:*               *
udp        0      0 localhost:domain       0.0.0.0:*               *
udp        0      0 Z238-UL:bootpc         _gateway:bootps        ESTABLISHED
udp        0      0 0.0.0.0:631           0.0.0.0:*               *
udp        0      0 localhost:domain       0.0.0.0:*               *
udp        0      0 Z238-UL:bootpc         _gateway:bootps        ESTABLISHED
udp        0      0 0.0.0.0:631           0.0.0.0:*               *
udp6       0      0 [::]:mdns              [::]:*                  *

```

6. 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.

```

sjcet@Z238-UL:~/kishor/sem2/CN$ hostname
Z238-UL

```

7. 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.

```
sjcet@Z238-UL:~/kishor/sem2/CN$ arp -a  
_gateway (172.16.48.2) at 7c:5a:1c:cf:50:b9 [ether] on eno1
```

8. 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.

```
sjcet@Z238-UL:~/kishor/sem2/CN$ uname -a  
Linux Z238-UL 5.4.0-147-generic #164-Ubuntu SMP Tue Mar 21 14:23:17 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
```

9. Analyzing network packet stream using tcpdump and wireshark. Perform basic network service tests using nc.

Procedure

1. 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]

2. Display Available Interfaces

To list the number of available interfaces on the system, run the following command with -D option.

3. Capture Packets from Specific Interface

The command screen will scroll up until you interrupt and when we execute the tcpdump command it will capture from all the interfaces, however with -i switch only capture from the desired interface.

4. 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
```

5. 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

6. 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.

7. 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

8. 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. Introduction to Hypervisors and VMs, Xen or KVM , Introduction to Containers: Docker, installation and deployment

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

```
mca@U40:~$ sudo apt install qemu-kvm libvirt-daemon-system libvirt-clients bridge-utils virt-manager
Reading package lists... Done
Building dependency tree
Reading state information... Done
qemu-kvm is already the newest version (1:2.11+dfsg-1ubuntu7.4).
The following additional packages will be installed:
  augeas-lenses dmeventd ebttables gir1.2-appindicator3-0.1 gir1.2-gtk-vnc-2.0
  gir1.2-libosinfo-1.0 gir1.2-libvirt-glib-1.0 gir1.2-spiceclientglib-2.0
  gir1.2-spiceclientgtk-3.0 libaugeas0 libdevmapper-event1.02.1
  libgovirt-common libgovirt2 libgtk-vnc-2.0-0 libgvnc-1.0-0 liblvm2app2.2
  liblvm2cmd2.02 libnetcf1 libosinfo-1.0-0 libphodav-2.0-0
  libphodav-2.0-common libspice-client-glib-2.0-8 libspice-client-gtk-3.0-5
  libusbredirhost1 libvirt-daemon libvirt-daemon-driver-storage-rbd
  libvirt-glib-1.0-0 libvirt0 libxml2-utils lvm2 osinfo-db python-asn1crypto
  python-certifi python-cffi-backend python-chardet python-cryptography
  python-dbus python-enun34 python-gi python-gi-cairo python-ldna
  python-ipaddr python-ipaddress python-libvirt python-libxml2 python-openssl
  python-pkg-resources python-requests python-six python-urllib3
  spice-client-glib-usb-acl-helper virt-viewer virtinst
Suggested packages:
  augeas-doc augeas-tools libosinfo-l10n gstreamer1.0-plugins-bad
  gstreamer1.0-libav libvirt-daemon-driver-storage-gluster
  libvirt-daemon-driver-storage-sheepdog libvirt-daemon-driver-storage-zfs
```

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.



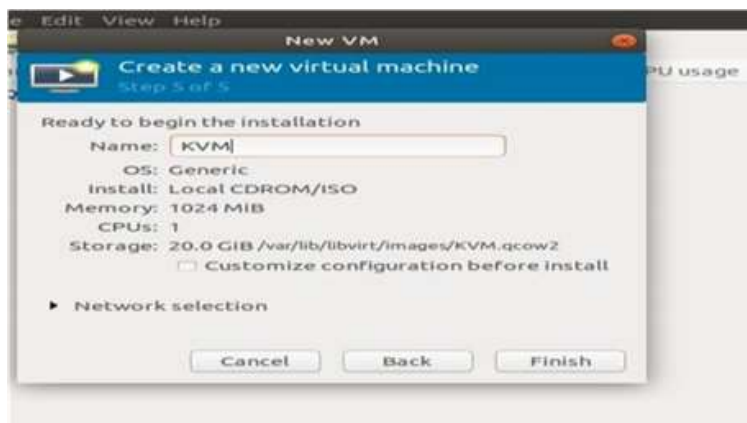
Step 5: Choose ISO, click Forward

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

```
mca@U40:~$ sudo virsh list --all
Id      Name      State
-----
1       KVM       running
mca@U40:~$
```

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.