LAB CYCLE:1 **DATE: 07-03-2024**

EXPERIMENT NO:1

Aim: Familiarisation with computer hardware.

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. Specifications of desktop and server class computers. Installation of common operating systems for desktop and server use.

Description:

a) Introduction to Computer hardware

Components of a computer system are the primary elements which make the functioning of an electronic device smooth and faster. Computer systems consist of three components as shown in below image: Central Processing Unit, Input devices and Output devices. Input devices provide data input to processor, which processes data and generates useful information that's displayed to the user through output devices. This is stored in computer's memory.

The operations of computer components are given below:

1) Inputting: It is the process of entering raw data, instructions and information into the computer. It is performed with the help of input devices.

2) Storing: The computer has primary memory and secondary storage to store data and instructions. It stores the data before sending it to CPU for processing and also stores the processed data before displaying it as output.

3) Processing: It is the process of converting the raw data into useful information. This process is performed by the CPU of the computer. It takes the raw data from storage, processes it and then sends back the processed data to storage.

4) Outputting: It is the process of presenting the processed data through output devices like monitor, printer and speakers.

5) Controlling: This operation is performed by the control unit that is part of CPU. The control unit ensures that all basic operations are executed in a right manner and sequence.

Central Processing Unit (CPU)

A Central Processing Unit is also called a processor, central processor, or microprocessor. It carries out all the important functions of a computer. It receives instructions from both the hardware and active software and produces output accordingly. It stores all important programs like operating systems and application software. CPU also helps Input and output devices to communicate with each other. Owing to these features of CPU, it is often referred to as the brain of the computer.

CPU has three components:

i ALU (Arithmetic Logic Unit)
ii Control Unit
iii Memory or Storage Unit

b) Physical identification of major components

Hardware, which is abbreviated as HW, refers to all physical components of a computer system, including the devices connected to it. You cannot create a computer or use software without using hardware. The screen on which you are reading this information is also a hardware

1. Motherboard: The motherboard is generally a thin circuit board that holds together almost all parts of a computer except input and output devices. All crucial hardware like CPU, memory, hard drive, and ports for input and output devices are located on the motherboard. It is the biggest circuit board in a computer chassis. It allocates power to all hardware located on it and enables them to communicate with each other. It is meant to hold the computer's microprocessor chip and let other components connect to it. Each component that runs the computer or improves its performance is a part of the motherboard or connected to it through a slot or port.

2. RAM Modules: RAM stands for Random Access Memory. It is also called the main memory. RAM is a temporary data storage device in computers and other devices. SRAM, DRAM, SDRAM, DDR etc are the various types of RAM available. A memory module or RAM stick is a narrow-printed circuit board that holds memory chips (RAM chips).

SRAM (static RAM) is random access memory (RAM) that retains data bits in its memory as long as power is being supplied. You can lose data if your SRAM is not powered. SRAM does not offer to refresh programs. SRAM has a low storage capacity (about 1MB).

Dynamic Random Access memory (DRAM) is a type of semiconductor memory that is typically used for the data or program code needed by a computer processor to function. The advantage of a DRAM is it only requires a single transistor compared to around six in a typical static RAM, SRAM memory cell. The costs of DRAM are much lower than those for SRAM, and they are able to provide much higher levels of memory density (about 1GB).

SDRAM (synchronous DRAM) is a generic name for various kinds of DRAM that are synchronized with the clock speed that the microprocessor is optimized for. That is same external clock pulse can be used to operate both SRAM and processor. This tends to increase the number of instructions that the processor can perform in a given time.

DDR Stands for "Double Data Rate." It is an advanced version of SDRAM, DDR-SDRAM can transfer data twice as fast as regular SDRAM chips. This is because DDR memory can send and receive signals twice per clock cycle. DDR operates about 2.5 V and DDR2 averages about 1.8 V, with DDR3 the voltage is reduced to 1.5 V.DDR3 has transfer rates between 800MT/s and 1600MT/s.DDR4 is the latest generation of DDR. It has the lowest operating voltage of 1.2 V and

has higher transfer rates than previous generations. DDR5 launch speeds delivery nearly double the bandwidth of DDR4. It also enables scaling memory performance without degrading channel efficiency at higher speeds under real-world conditions. Crucial DDR5 memory will operate at 4800MT/s at launch, which is 1.5x the maximum standard DDR4 speed.

3. Daughter cards: A daughterboard (daughter card) is a type of circuit board that plugs in or is attached to the motherboard or similar expansion card to extend its features and services.

A daughterboard is connected directly to the motherboard. Like a motherboard, a daughterboard has sockets, pins, plugs and connectors to be attached to other boards. Today, these boards are not found or used in desktop computers. They were replaced with ISA card, PCI card and onboard options. With the rise of connective USB ports and other technology, it has become less necessary to upgrade devices with daughtercards or daughterboards.

4. Bus Slots: An expansion slot is a socket on the motherboard that is used to insert an expansion card, which provides additional features to a computer such as video, sound, advanced graphics, Ethernet or memory.

PATA stands for Parallel Advanced Technology Attachment and SATA stands for Serial Advanced Technology Attachment both are two bus interfaces used for connecting secondary storage devices like hard disks, optical drives.

Serial refers to the fact that data is sent one bit at a time down a single connection in each direction. There's a separate connection for data going in to and out of the device. Parallel refers to the fact that data is sent 16 bits at a time through a single 16-bit connection, which is used for data traveling in both directions.

SCSI stands for Small Computer System Interface is a set of standards for physically connecting and transferring data between computers and peripheral devices.

IDE stands for Integrated Drive Electronics is a standard interface for connecting a motherboard to storage devices such as hard drives and CD/DVD drives.

PCI (Peripheral Component Interconnect) is a bus standard that connects the computer motherboard and external devices. PCIe (peripheral component interconnect express) is an interface standard for connecting high-speed components. Every desktop PC motherboard has a number of PCIe slots.

5. SMPS: SMPS stands for Switched-Mode Power Supply. It is an electronic power supply that uses a switching regulator to convert electrical power efficiently.

It is a power supply unit (PSU) generally used in computers to convert the voltage into the computer acceptable range. Technically briefing, an SMPS in a desktop system that converts 220V AC and 50HZ into +5V, -5V, +12V and +3.3 V DC at various electrical components in the computer.

6. Internal Storage: A storage device is an integral part of the computer hardware which stores information or data to process the result of any computational work. Internal storage is a storage device that's internal (inside the case) and is not a removable storage or external storage. For example, the hard drive inside your computer is an example of internal storage.HDD (Hard Disk Drive) also known as fixed disk uses magnetic tape for the storage of data. HDD has a moving read/write head to access data from storage like a gramophone player and slower to read and write. Unlike HDD, SSD (Solid State Drive) has no moving parts, and it obtains data from storage instantly. SSD can give faster performance than traditional magnetic-based computer storage devices. Although both of them perform the same task.

7. Interfacing ports: A port is a connection or a jack provided on a computer to connect external or peripheral devices to the computer, for example, you will need a port on your device to connect a keyboard, mouse, pen-drives, etc. So, it acts as an interface or a point of attachment between computer and external devices. It is also called a communication port, as it is the point where you plug in a peripheral device to allow data transfer or communication between the device and computer. Generally, they are four to six in number and present on the back or sides of the computer.

Based on the type of protocol used for communication, computer ports can be of two types: Serial Ports and Parallel Ports.

i. Serial Ports

This type of ports provides an interface to connect to peripheral devices using a serial protocol. In this port, the rate of transmission of data is one bit at a time through a single communication line. For example, D-Subminiature or D-sub connector is a commonly used serial port, which carries RS-232 signals.

ii. Parallel Ports

As the name suggests, a parallel port is an interface that allows communication or data transfer between a computer and a device in a parallel manner through more than one communication line. For example, a printer port is a parallel port.

Some computer ports are:

PS/2

VGA

DVI

RJ-45

HDMI

8. Specifications of desktop and server class:

Desktop:

Processor	Core i5
Processor speed	3.90 GHz
Number of cores	4
Typical Memory	32GB
Cache size	L1:8KB -1MB, L2:256KB - 3MB
Memory type	DDR4

Web Server:

Processor	Intel® Xeon® Bronze 3206R
Processor (Multiple Processors)	
Processor speed	1.90 GHz
Number of cores	8
Typical Memory	512GB
Cache size	L1:1-2MB, L2:8MB, L3:32-64MB

Result:

Familiarisation of computer and hardware done successfully.

LAB CYCLE:1 **DATE: 07-03-2024**

EXPERIMENT NO:2

Aim: Familiarisation of OS installation.

Oracle Virtual Box is already installed on your Desktop.

1. Create a Virtual Machine with sufficient RAM, approximately half the RAM of your desktop.
2. Create a virtual hard disk (VMDK) of 20 GB with variable size.
3. Download Ubuntu 20.04 Desktop ISO (64-bit) from Ubuntu's official website or from http://10.2.7.253/downloads for a quicker download.
4. Attach this ISO to the Virtual Machine as a CD Drive.
5. Boot from it and execute GParted.
6. Create two primary partitions of equal size and install Ubuntu on one of them.
7. Make a copy of the VMDK file for later experiments.
8. You can also install another OS on the other partition. Write a description of VirtualBox and the VMDK format. Take screenshots of Steps 1, 2, 4, and 6 and affix them.

Solution:

VirtualBox is an open-source software for virtualizing the x86 computing architecture. It acts as a hypervisor, creating a VM (virtual machine) where the user can run another OS (operating system).

VMDK (short for Virtual Machine Disk) is a file format that describes containers for virtual hard disk drives to be used in VM.

Installation of Ubuntu using VirtualBox:

1. We have installed VirtualBox and you have downloaded the Ubuntu. Now set to install Linux in VirtualBox.
2. Start VirtualBox, and click on the New symbol. Give the virtual OS a relevant name.
3. Allocate RAM to the virtual OS.

4. Create a virtual disk. This serves as the hard disk of the virtual Linux system.
5. Choose VDI file type as file type for new virtual hard disk.

6. Choose either the “Dynamically allocated” or the “Fixed size” option for creating the virtual hard disk.
7. Allocate size for virtual hard disk (recommended size is 20GB).

8. Install Ubuntu
9.

10. Open the VM box. It shows all the details and properties of the virtual box.

Installation of another OS (Linux Mint) to the other partition:

1. Download Linux Mint OS.
2. Choose:

3.

4. Install Linux mint.

Result:

Linux installed successfully.

LAB CYCLE: 2 **DATE: 07-03-2024**

EXPERIMENT NO: 3

TEXT EDITOR

Aim: Study of a terminal-based text editor such as Vim or Emacs.

There are many ways to edit files in Unix. Editing files using the screen-oriented text editor vi is one of the best ways. This editor enables you to edit lines in context with other lines in the file. An improved version of the vi editor which is called the VIM has also been made available now. Here, VIM stands for Vi IMproved. The vi editor was developed by William Joy as a more visual version of his own command line editor program called ex, which was becoming popular as a text editor.

The vi editor uses the terminal window for editing a file. For an example, run the following command to open a new file or an existing file of the same name:

vi filename

There are two ways of working in vi:

- Insert Mode: This mode enables you to insert text into the file. Everything that's typed in this mode is interpreted as input and placed in the file. To come out of the insert mode, press the Esc key, which will take you back to the command mode.
- Command Mode: This mode enables you to perform administrative tasks such as saving the files, executing the commands, moving the cursor, cutting (yanking) and pasting the lines or words, as well as finding and replacing. In this mode, whatever you type is interpreted as a command.

Commands and descriptions:

- vi filename
  #Creates a new file if it already does not exist, otherwise opens an existing file.
- vi -R filename
  #Opens an existing file in the read-only mode.
- view filename
  #Opens an existing file in the read-only mode.

Create a new file testfile if it already does not exist in the current working directory –

The above command will generate the following output –

The tilde (~) on each line following the cursor. A tilde represents an unused line. If a line does not begin with a tilde and appears to be blank, there is a space, tab, newline, or some other non-viewable character present.

a) Cursor operations

In command mode we can position the cursor anywhere in the file. Since you begin all basic edits (changing, deleting, and copying text) by placing the cursor at the text that you want to change, you want to be able to move the cursor to that place as quickly as possible.

There are vi commands to move the cursor:

- Up, down, left, or right—one character at a time.
- Forward or backward by blocks of text such as words, sentences, or paragraphs.
- Forward or backward through a file, one screen at a time.

An underscore marks the present cursor position. Circles show movement of the cursor from its current position to the position that would result from various vi commands.

Single Movements

The keys h, j, k, and l, right under your fingertips, will move the cursor:

h: Left, one space
j: Down, one line
k: Up, one line
l: Right, one space

You can also use the cursor arrow keys (←, ↓, ↑, →), + and - to go up and down, or the ENTER and BACKSPACE keys.

b) Manipulate text

1. Open a new or existing file with vim filename.
2. Type i to switch into insert mode so that you can start editing the file. Enter or modify the text with your file.
3. Once you're done, press the escape key Esc to get out of insert mode and back to command mode.
4. Type :wq to save and exit your file.

Vim commands for editing#:

Those who use Vim tend to use the term "yank" where most people would use the terms copy and paste. Therefore, the command for copying a word is yw, which stands for yank word, and the command for pasting whatever has been copied is p, meaning put. If you look up additional commands in the future, it can be confusing if you don't know what yank and put mean when using Vim.

You also have two options for how to select text. You can either use commands like dd, which deletes a single line, and yy, which copies a single line, or you can highlight text and then copy it to the unnamed register. The paste commands work the same whether you've highlighted text or used a command to automatically copy it.

As with movement commands, putting a number in front of the command can increase the number of times a task is completed. For instance, putting a number in front of yy will increase the number of lines copied, so 5yy will copy five lines.

yy - Copies a line

yw - Copies a word

y$ - Copies from where your cursor is to the end of a line

v - Highlight one character at a time using arrow buttons or the h, k, j, l buttons

V - Highlights one line, and movement keys can allow you to highlight additional lines

p - Paste whatever has been copied to the unnamed register

d - Deletes highlighted text

dd - Deletes a line of text

dw - Deletes a word

D - Deletes everything from where your cursor is to the end of the line

. - Repeats the last action

one

Vim commands for working with multiple files#:

You can also edit more than one text file at a time. Vim gives you the ability to either split your screen to show more than one file at a time or you can switch back and forth between documents. As with other functions, commands make going between documents or buffers, as they're referred to with Vim, as simple as a few keystrokes.

:bn - Switch to next buffer

:bp - Switch to previous buffer

:bd - Close a buffer

:sp [filename] - Opens a new file and splits your screen horizontally to show more than one buffer

:vsp [filename] - Opens a new file and splits your screen vertically to show more than one buffer

:ls - Lists all open buffers

Ctrl + ws - Split windows horizontally

Ctrl + wv - Split windows vertically

Ctrl + ww - Switch between windows

Ctrl + wq - Quit a window

Ctrl + wh - Moves your cursor to the window to the left

Ctrl + wl - Moves your cursor to the window to the right

Ctrl + wj - Moves your cursor to the window below the one you're in

Ctrl + wk - Moves your cursor to the window above the one you're in

c) Search for patterns

Vim commands for searching text#:

Like many other text editors, Vim allows you to search your text and find and replace text within your document. If you opt to replace multiple instances of the same keyword or phrase, you can set Vim up to require or not require you to confirm each replacement depending on how you put in the command.

/[keyword] - Searches for text in the document where keyword is whatever keyword, phrase or string of characters you're looking for

?[keyword] - Searches previous text for your keyword, phrase or character string n - Searches your text again in whatever direction your last search was

N - Searches your text again in the opposite direction

:%s/[pattern]/[replacement]/g - This replaces all occurrences of a pattern without confirming each one

:%s/[pattern]/[replacement]/gc - Replaces all occurrences of a pattern and confirms each

We can use the slash to search for a word, and then use the dot to replace it. open the sample.txt file using the Vim editor:

$ vim sample.txt we need to press the forward-slash(/) key, then search for the word “article”:

/article

This will highlight the first occurrence of the word “article” and we can press the Enter key to jump to it.

d) Global search and replace

We can use the slash to search for a word, and then use the dot to replace it.

Open the file using the Vim editor:

$ vim filename

we need to press the forward-slash(/) key, then search for the word:

/word to be searched

This will highlight the first occurrence of the word and we can press the Enter key to jump to it.

We can then type in the cgn command combo:

cgn

This Vim editor command finds the last thing we searched for, deletes it, and then put us into insert mode.

We can then press the Esc key to return to normal mode.

Next, we need to press the “N” key to jump to the next occurrence of the word and press the Dot (.) key to auto-replace it with the word.

This is the simplest method to perform a basic search and replace in Vim editor.

Search and Replace Using the substitute Command

Basic syntax:

:s/<search_phrase>/<replace_phrase>/options

- All Occurrences

  :%s/article/tutorial/g
- Case-Insensitive

  :%s/vim/baeldung/gi
- With Confirmation

  :%s/article/tutorial/gc
- Within Specific Lines

  :start_line_number, end_line_number s/<search_term>/<replace_term>/g
- Whole Word

  :s/\<cover\>/go through/gi

Result:

Study of text editor has been done successfully.

LAB CYCLE: 2 **DATE: 14-03-2024**

EXPERIMENT NO: 4

BASIC LINUX COMMANDS

Aim: Basic Linux commands, familiarity with following commands/operations expected.

1. **man:** is used to display the user manual of any command that we can run on the terminal. It provides a detailed view of the command which includes NAME, SYNOPSIS, DESCRIPTION, OPTIONS, EXIT STATUS, RETURN VALUES, ERRORS, FILES, VERSIONS, EXAMPLES, AUTHORS and SEE ALSO.

2. **ls, echo, read**

   ls: is a Linux shell command that lists directory contents of files and directories. Some practical examples of ls command are shown below.

   Options:

   -t: It sorts the file by modification time, showing the last edited file first.

   -l: To show long listing information about the file/directory.

   -lh: To display file size in easy-to-read format.

   -a: To show all the hidden files in the directory.

```
exam@CSCC2-19:~$ ls
 cpu.class              mca-nsa                                  'Screenshot from 2023-04-05 10-36-01.png'   stringsort.java
 cpu.java               rithu.cpp                                'Screenshot from 2023-04-05 10-37-09.png'   Student.class
'cpu$processor.class'  'Screenshot from 2023-04-05 10-30-12.png' 'Screenshot from 2023-04-05 10-37-16.png'   Student.java
'cpu$ram.class'        'Screenshot from 2023-04-05 10-32-56.png'  sreeraj
 elle                  'Screenshot from 2023-04-05 10-33-03.png'  stringsort.class
exam@CSCC2-19:~$
```

echo: used for displaying lines of text or string which are passed as arguments on the command line.

```
exam@CSCC2-19:~$ echo "welcome"
welcome
exam@CSCC2-19:~$
```

read: is used to read from a file descriptor.

```
exam@CSCC2-19:~$ read -p "Enter name" name
Enter name pradnya
exam@CSCC2-19:~$ echo "$name"
pradnya
exam@CSCC2-19:~$
```

3. **more, less, cat**

more: reads files and displays the text one screen at a time.

```
exam@CSCC2-19:~/mca-nsa$ more file1.txt
welcome
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
exam@CSCC2-19:~/mca-nsa$
```

less: used for filtering and viewing text files one screen page at a time

```
exam@CSCC2-19:~/mca-nsa$ less file1.txt

[7]+  Stopped                 less file1.txt
exam@CSCC2-19:~/mca-nsa$
```

```
welcome
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
(END)
```

cat: used to display the content of a file, copy content from one file to another, concatenate the contents of multiple files, display the line number, display $ at the end of the line, etc.

```
exam@CSCC2-19:~/mca-nsa$ cat >file1.txt
welcome
exam@CSCC2-19:~/mca-nsa$ cat file1.txt
welcome
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
exam@CSCC2-19:~/mca-nsa$
```

4. **cd, mkdir, pwd, find**

cd: cd command in linux known as change directory command. It is used to change current working directory.

Syntax: cd [directory_name]

```
exam@CSCC2-19:~/mca-nsa$ cd ..
exam@CSCC2-19:~$ cd mca-nsa
exam@CSCC2-19:~/mca-nsa$
```

mkdir: mkdir command in Linux allows the user to create directories (also referred to as folders in some operating systems). This command can create multiple directories at once as well as set the permissions for the directories.

Syntax: mkdir [options...] [directories ...]

```
exam@CSCC2-19:~/sreeraj$ cd ..
exam@CSCC2-19:~$ mkdir mca-nsa
exam@CSCC2-19:~$ ls
 cpu.class              elle                                  'Screenshot from 2023-04-05 10-32-56.png'   stringsort.java
 cpu.java               mca-nsa                               'Screenshot from 2023-04-05 10-33-03.png'   Student.class
'cpu$processor.class'   rithu.cpp                              sreeraj                                    Student.java
'cpu$ram.class'         'Screenshot from 2023-04-05 10-30-12.png'   stringsort.class
exam@CSCC2-19:~$
```

pwd: pwd stands for Print Working Directory. It prints the path of the working directory, starting from the root.

Syntax: pwd -L: Prints the symbolic path.
pwd -P: Prints the actual path.

```
exam@CSCC2-19:~/mca-nsa/d1$ pwd
/home/exam/mca-nsa/d1
exam@CSCC2-19:~/mca-nsa/d1$
```

find: The find command in UNIX is a command line utility for walking a file hierarchy. It can be used to find files and directories and perform subsequent operations on them. It supports searching by file, folder, name, creation date, modification date, owner and permissions.

Syntax: find [where to start searching from] [expression determines what to find] [-options] [what to find]

```
exam@CSCC2-19:~/mca-nsa$ find file1.txt
file1.txt
exam@CSCC2-19:~/mca-nsa$
```

5. **mv, cp, rm, tar**

mv: mv stands for move. mv is used to move one or more files or directories from one place to another in a file system like UNIX. It has two distinct functions:

(i) It renames a file or folder.
(ii) It moves a group of files to a different directory.

Syntax: mv [Option] source destination

```
exam@CSCC2-19:~/mca-nsa$ mkdir d1
exam@CSCC2-19:~/mca-nsa$ mv file1.txt d1
exam@CSCC2-19:~/mca-nsa$ cd d1
exam@CSCC2-19:~/mca-nsa/d1$ ls
file1.txt
exam@CSCC2-19:~/mca-nsa/d1$
```

cp: cp stands for copy. This command is used to copy files or group of files or directory. It creates an exact image of a file on a disk with different file name. cp command requires at least two filenames in its arguments.

Syntax: cp [OPTION] Source Destination

```
exam@CSCC2-19:~/mca-nsa$ ls
d1  file.txt
exam@CSCC2-19:~/mca-nsa$ cp file.txt d1
exam@CSCC2-19:~/mca-nsa$ cd d1
exam@CSCC2-19:~/mca-nsa/d1$ ls
file1.txt  file.txt  mca-nsa
exam@CSCC2-19:~/mca-nsa/d1$
```

rm: rm stands for remove here. rm command is used to remove objects such as files, directories, symbolic links and so on from the file system like UNIX. To be more precise, rm removes references to objects from the filesystem, where those objects might have had multiple references.

Syntax: rm [OPTION]... FILE...

```
exam@CSCC2-19:~/mca-nsa/d1$ rm mca-nsa
exam@CSCC2-19:~/mca-nsa/d1$ ls
file1.txt  file.txt
exam@CSCC2-19:~/mca-nsa/d1$
```

tar: The Linux 'tar' stands for tape archive, is used to create Archive and extract the Archive files. tar command in Linux is one of the important commands which provides archiving functionality in Linux. We can use Linux tar command to create compressed or uncompressed Archive files and also maintain and modify them.

Syntax: tar [options] [archive-file] [file or directory to be archived]

```
lab@lab-Lenovo-IdeaPad-Z400:~/dir1$ cd
lab@lab-Lenovo-IdeaPad-Z400:~$ tar -czvf mydir.tar.gz dir1
dir1/
dir1/F3
dir1/dir2/
dir1/dir2/set.c
dir1/dir2/testfile
dir1/dir2/result
dir1/dir2/out2.txt
dir1/dir2/dir3/
dir1/dir2/dir3/testfile3
dir1/dir2/dir3/v
dir1/dir2/dir3/sample
dir1/dir2/dir3/age
dir1/dir2/dir3/marks
dir1/dir2/dir3/testfile
dir1/dir2/dir3/s
dir1/dir2/dir3/name
dir1/dir2/dir3/csvfile
dir1/dir2/dir3/file1
dir1/dir2/dir3/tesfile
dir1/dir2/dir3/testfile2
```

6. **wc, cut, paste**

wc: wc stands for word count.It is used to find out number of lines, word count, byte and characters count in the files specified in the file arguments.

Syntax: wc [OPTION]... [FILE]...

```
exam@CSCC2-19:~/mca-nsa/d1$ wc file1.txt
21 21 59 file1.txt
exam@CSCC2-19:~/mca-nsa/d1$ wc -l file1.txt
21 file1.txt
exam@CSCC2-19:~/mca-nsa/d1$ wc -c file1.txt
59 file1.txt
exam@CSCC2-19:~/mca-nsa/d1$ wc -w file1.txt
21 file1.txt
exam@CSCC2-19:~/mca-nsa/d1$
```

cut: The cut command in UNIX is a command for cutting out the sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by byte position, character and field.

Syntax: cut OPTION... [FILE]...

```
exam@CSCC2-19:~/mca-nsa/d1$ cut -c 1,3,7 file.txt
h l
exam@CSCC2-19:~/mca-nsa/d1$ cut -c 1,3,7 file1.txt
wle
1
2
3
4
5
6
7
8
9
1
1
1
1
1
1
1
1
1
1
2
exam@CSCC2-19:~/mca-nsa/d1$
```

paste: paste command is one of the useful commands in Unix or Linux operating system. It is used to join files horizontally (parallel merging) by outputting lines consisting of lines from each file specified, separated by tab as delimiter, to the standard output.

Syntax: paste [OPTION]... [FILES]...

```
exam@CSCC2-19:~/mca-nsa/d1$ paste -s number file.txt file1.txt
paste: number: No such file or directory
hi hello
welcome 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
8	19	20
exam@CSCC2-19:~/mca-nsa/d1$
```

7. **head, tail, grep, expr:**

head: it prints the first 10 lines of the specified files. If more than one file name is provided then data from each file is preceded by its file name.

Syntax: head [OPTION]... [FILE]...

```
exam@CSCC2-19:~/mca-nsa/d1$ head -10 file1.txt
welcome
1
2
3
4
5
6
7
8
9
exam@CSCC2-19:~/mca-nsa/d1$
```

tail: it prints the last 10 lines of the specified files. If more than one file name is provided then data from each file is precedes by its file name.

Syntax: tail [OPTION]... [FILE]...

```
exam@CSCC2-19:~/mca-nsa/d1$ tail -10 file1.txt
11
12
13
14
15
16
17
18
19
20
exam@CSCC2-19:~/mca-nsa/d1$
```

grep: the grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression (grep stands for global search for regular expression and print out).

Syntax: grep [options] pattern [files]

```
exam@CSCC2-19:~/mca-nsa/d1$ grep -i "wel" file1.txt
welcome
exam@CSCC2-19:~/mca-nsa/d1$
```

8. **chmod, chown:**

chmod: the chmod command is used to change the access mode of a file.The name is an abbreviation of change mode.

Syntax: chmod [reference][operator][mode] file...

```
exam@CSCC2-19:~/mca-nsa/d1$ chmod +x file1.txt
exam@CSCC2-19:~/mca-nsa/d1$ ls
file1.txt  file.txt
exam@CSCC2-19:~/mca-nsa/d1$
```

chown: it is used to set read, write, execute permission for user. To protect and secure files and directory in Linux we use permissions to control what a user can do with a file or directory.

Syntax: chown [OPTION]… [OWNER][:[GROUP]] FILE…

chown [OPTION]… –reference=RFILE FILE…

```
Number of days of warning before password expires       : 7
lab@lab-Lenovo-IdeaPad-Z400:/home$ sudo chown -R u_ser1:u_ser1 dir2
chown: cannot access 'dir2': No such file or directory
lab@lab-Lenovo-IdeaPad-Z400:/home$ sudo chown -R u_ser1:u_ser1 /home/lab/dir1/dir2
```

9. **Redirections & Piping:**

Redirection: whenever an individual runs a command, it can take input, give output, or do both. Redirection helps us redirect these input and output functionalities to the files or folders we want, and we can use special commands or characters to do so.

Append redirection
“>>” standard output
“<<” standard input

```
exam@CSCC2-19:~/mca-nsa/d1$ ls -l >>lsoutput
exam@CSCC2-19:~/mca-nsa/d1$ cat lsoutput
file1.txt
file.txt
lsoutput
total 12
-rwxrwxr-x 1 exam exam 59 Apr  5 10:43 file1.txt
-rw-rw-r-- 1 exam exam  9 Apr  5 10:54 file.txt
-rw-rw-r-- 1 exam exam 28 Apr  5 11:15 lsoutput
exam@CSCC2-19:~/mca-nsa/d1$
```

Overwrite Redirection
“>” standard output
“<” standard input

```
exam@CSCC2-19:~/mca-nsa/d1$ ls
file1.txt  file.txt
exam@CSCC2-19:~/mca-nsa/d1$ ls >lsoutput
exam@CSCC2-19:~/mca-nsa/d1$ cat lsoutput
file1.txt
file.txt
lsoutput
exam@CSCC2-19:~/mca-nsa/d1$
```

Piping: Pipe is used to combine two or more commands, and in this, the output of one command acts as input to another command, and this command’s output may act as input to the next command and so on. It can also be visualized as a temporary connection between two or more commands/ programs/ processes. The command line programs that do the further processing are referred to as filters.
Syntax: command_1 | command_2 | command_3 | .... | command_N

```
exam@CSCC2-19:~/mca-nsa/d1$ ls -l | more
total 12
-rwxrwxr-x 1 exam exam  59 Apr  5 10:43 file1.txt
-rw-rw-r-- 1 exam exam   9 Apr  5 10:54 file.txt
-rw-rw-r-- 1 exam exam 182 Apr  5 11:16 lsoutput
exam@CSCC2-19:~/mca-nsa/d1$
```

10. **useradd, usermod, userdel, passwd:**

useradd: is a command in Linux that is used to add user accounts to your system. It is just a symbolic link to adduser command in Linux and the difference between both of them is that useradd is a native binary compiled with system whereas adduser is a Perl script which uses useradd binary in the background.
Syntax: useradd [options] name_of_the_user

```
lab@lab-Lenovo-IdeaPad-Z400:/home$ sudo useradd -s /bin/bash -m -d /home/u_ser1 u_ser1
lab@lab-Lenovo-IdeaPad-Z400:/home$ sudo chage -l u_ser1
Last password change                                    : May 08, 2022
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : never
```

usermod: is a command in Linux that is used to change the properties of a user in Linux through the command line. After creating a user, we have to sometimes change their attributes like password or login directory etc. so in order to do that we use the Usermod command.
Syntax: sudo usermod -c "This is test user" test_user

```
lab@lab-Lenovo-IdeaPad-Z400:/home$ sudo useradd -s /bin/bash -m -d /home/u_ser1 u_ser1
lab@lab-Lenovo-IdeaPad-Z400:/home$ sudo chage -l u_ser1
Last password change                                    : May 08, 2022
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : never
```

userdel: is used to delete a user account and related files. This command basically modifies the system account files, deleting all the entries which refer to the username LOGIN. It is a low-level utility for removing the users.
Syntax: userdel [options] LOGIN

```
lab@lab-Lenovo-IdeaPad-Z400:~$ sudo userdel u_ser1
```

passwd: passwd command in Linux is used to change the user account passwords. The root user reserves the privilege to change the password for any user on the system, while a normal user can only change the account password for his or her own account.
Syntax: passwd [options] [username]

11. df,top, ps:

df: is used to displays the amount of disk space available on the file system containing each file name argument.
Syntax: df [OPTION]...[FILE]...

```
lab@lab-Lenovo-IdeaPad-Z400:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           580M  1.9M  578M   1% /run
/dev/sda7       281G   19G  248G   8% /
tmpfs           2.9G   93M  2.8G   4% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           4.0M     0  4.0M   0% /sys/fs/cgroup
tmpfs           580M  1.3M  579M   1% /run/user/1000
lab@lab-Lenovo-IdeaPad-Z400:~$
```

top: top command is used to show the Linux processes. It provides a dynamic real-time view of the running system. Usually, this command shows the summary information of the system and the list of processes or threads which are currently managed by the Linux Kernel.

```
top - 00:05:26 up 12:05,  1 user,  load average: 0.26, 0.32, 0.34
Tasks: 260 total,   1 running, 259 sleeping,   0 stopped,   0 zombie
%Cpu(s): 11.4 us,  2.6 sy,  0.0 ni, 83.5 id,  2.5 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   5797.5 total,    682.1 free,   2352.4 used,   2763.0 buff/cache
MiB Swap:  11443.0 total,  11443.0 free,      0.0 used.   2787.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  35342 lab       20   0  402144  44192  33592 S  15.6   0.7   0:00.47 gnome-screensho
  35333 lab       20   0   21584   4012   3336 R   0.7   0.1   0:00.12 top
  35313 root      20   0       0      0      0 I   0.0   0.0   0:00.00 kworker/0:3-cgroup_destroy
  35244 root      20   0       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0-events
  35241 root      20   0       0      0      0 I   0.0   0.0   0:00.09 kworker/u16:0-events_unbound
  35192 lab       20   0 1108.5g  63928  48524 S   0.0   1.1   0:00.04 chrome
  35181 lab       20   0 1108.5g 166884 101196 S   2.0   2.8   0:16.81 chrome
  35173 lab       20   0   20.3g  42584  31328 S   0.0   0.7   0:00.02 chrome
  35131 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/u17:0
  35112 root      20   0       0      0      0 I   0.0   0.0   0:00.09 kworker/3:2-events
  35101 root      20   0       0      0      0 I   0.3   0.0   0:00.17 kworker/u16:3-phy0
  35061 lab       20   0   17228   2556   2284 S   0.0   0.0   0:00.14 pager
  35051 lab       20   0   18492   4044   2924 S   0.0   0.1   0:00.01 man
  35043 root      20   0       0      0      0 I   0.0   0.0   0:00.18 kworker/0:2-events
  35025 root      20   0       0      0      0 I   0.0   0.0   0:00.00 kworker/1:0-events
  34983 root      20   0       0      0      0 I   0.0   0.0   0:00.00 kworker/2:0-events
  34874 root      20   0       0      0      0 I   0.0   0.0   0:00.38 kworker/u16:2-iwlwifi
  34870 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/u17:1-rb_allocator
  34836 root      20   0       0      0      0 I   0.0   0.0   0:00.11 kworker/0:1-cgroup_destroy
  34808 root      20   0       0      0      0 I   0.3   0.0   0:01.18 kworker/2:1-events
  34691 root      20   0       0      0      0 I   0.0   0.0   0:00.78 kworker/u16:1-iwlwifi
  34497 root      20   0       0      0      0 I   0.0   0.0   0:00.24 kworker/3:1-events
  34257 root       0 -20       0      0      0 I   0.7   0.0   0:02.45 kworker/u17:2-i915_flip
  32684 root      20   0       0      0      0 I   0.0   0.0   0:01.28 kworker/1:2-events
  32586 root      20   0  395836  29828  25108 S   0.0   0.5   0:00.67 fwupd
  31958 lab       20   0   18716   4264   3604 S   0.0   0.1   0:00.11 bash
  31576 lab       20   0   18716   4420   3752 S   0.0   0.1   0:00.16 bash
  24267 lab       20   0  434172  71360  45500 S   5.0   1.2   1:24.22 gnome-terminal-
  17368 lab       20   0 3001548 134608  44836 S   0.0   2.3   0:09.03 gjs
  17246 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 irq/28-mei_me
  15010 lab       20   0  919828 121340  54424 S   0.0   2.0   1:25.03 nautilus
```

ps: it allows multiple processes to operate simultaneously without interfering with each other. Process is one of the important fundamental concepts of the Linux OS.
Syntax: ps [options]

```
lab@lab-Lenovo-IdeaPad-Z400:~$ ps -T
    PID    SPID TTY          TIME CMD
  31576   31576 pts/0    00:00:00 bash
  35666   35666 pts/0    00:00:00 ps
lab@lab-Lenovo-IdeaPad-Z400:~$ ps -A
    PID TTY          TIME CMD
      1 ?        00:00:03 systemd
      2 ?        00:00:00 kthreadd
      3 ?        00:00:00 rcu_gp
      4 ?        00:00:00 rcu_par_gp
      6 ?        00:00:00 kworker/0:0H-events_highpri
      9 ?        00:00:00 mm_percpu_wq
     10 ?        00:00:00 rcu_tasks_rude_
     11 ?        00:00:00 rcu_tasks_trace
     12 ?        00:00:00 ksoftirqd/0
     13 ?        00:00:36 rcu_sched
     14 ?        00:00:00 migration/0
     15 ?        00:00:00 idle_inject/0
     16 ?        00:00:00 cpuhp/0
     17 ?        00:00:00 cpuhp/1
     18 ?        00:00:00 idle_inject/1
     19 ?        00:00:00 migration/1
     20 ?        00:00:00 ksoftirqd/1
     22 ?        00:00:00 kworker/1:0H-kblockd
     23 ?        00:00:00 cpuhp/2
     24 ?        00:00:00 idle_inject/2
     25 ?        00:00:00 migration/2
     26 ?        00:00:00 ksoftirqd/2
     28 ?        00:00:00 kworker/2:0H-events_highpri
     29 ?        00:00:00 cpuhp/3
     30 ?        00:00:00 idle_inject/3
     31 ?        00:00:00 migration/3
     32 ?        00:00:00 ksoftirqd/3
     34 ?        00:00:00 kworker/3:0H-events_highpri
     35 ?        00:00:00 kdevtmpfs
     36 ?        00:00:00 netns
     37 ?        00:00:00 inet_frag_wq
     38 ?        00:00:00 kauditd
```

12. **ssh, scp, ssh-keygen, ssh-copy-id:**

ssh: stands for “Secure Shell”. It is a protocol used to securely connect to a remote server/system. ssh is secure in the sense that it transfers the data in encrypted form between the host and the client. It transfers inputs from the client to the host and relays back the output. ssh runs at TCP/IP port 22.

Syntax: ssh user_name@host(IP/Domain_name)

```
dnyce@LinuxShellTips:~$ ssh ubuntu@18.118.208.79
ubuntu@18.118.208.79's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-1045-aws x86_64)

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

  System information as of Mon Sep 13 07:42:19 UTC 2021

  System load:  0.0                Processes:              121
  Usage of /:   37.5% of 7.69GB    Users logged in:        0
  Memory usage: 43%                IPv4 address for eth0: 172.31.0.248
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

74 updates can be applied immediately.
1 of these updates is a standard security update.
To see these additional updates run: apt list --upgradable


*** System restart required ***
Last login: Mon Sep 13 07:40:24 2021 from 154.123.208.40
ubuntu@ip-172-31-0-248:~$
```

scp: scp (secure copy) command in Linux system is used to copy file(s) between servers in a secure way. The SCP command or secure copy allows secure transferring of files in between the local host and the remote host or between two remote hosts. It uses the same

authentication and security as it is used in the Secure Shell (SSH) protocol. SCP is known for its simplicity, security and pre-installed availability.

Syntax: scp [-346BCpqrTv] [-c cipher] [-F ssh_config] [-i identity_file] [-l limit] [-o ssh_option] [-P port] [-S program] [[user@]host1:]file1 … [[user@]host2:]file2

```
pratik@Linuxbuzz:~$ scp -v /var/log/syslog root@10.4.3.201:/root/
Executing: program /usr/bin/ssh host 10.4.3.201, user root, command scp -v -t /r
oot/
OpenSSH_8.2p1 Ubuntu-4ubuntu0.2, OpenSSL 1.1.1f  31 Mar 2020
```

ssh-keygen: Secure Shell (SSH) is a cryptographic network protocol used for operating remote services securely. It is used for remote operation of devices on secure channels using a clientserver architecture that generally operates on Port 22. SSH is the successor of Telnet. SSH uses public and private keys to validate and authenticate users. ssh-keygen is used to generate these key pairs.

```
ubuntu@ubuntu-VirtualBox:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa): my_key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in my_key.
Your public key has been saved in my_key.pub.
The key fingerprint is:
5d:48:47:39:3b:fd:64:75:b3:74:26:f8:04:81:91:7b ubuntu@ubuntu-VirtualBox
The key's randomart image is:
+--[ RSA 2048]----+
|          o===   |
|         .oo+ oo=|
|          ...*.o*|
|         ...E o.o|
|        S .. . + |
|                 .|
|                  |
|                  |
+-----------------+
ubuntu@ubuntu-VirtualBox:~$
```

ssh-copy-id: is a command that copies the public key of the local host to the authorized keys file of the remote host, allowing password-less and automatic login via SSH.

```
ubuntu01@linux:~$ ssh-copy-id kali@192.168.239.134
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are promp
ted now it is to install the new keys
kali@192.168.239.134's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'kali@192.168.239.134'"
and check to make sure that only the key(s) you wanted were added.

ubuntu01@linux:~$
```

Result:
Commands executed successfully and output is verified.

LAB CYCLE: 2 **DATE: 28-03-2024**

EXPERIMENT NO: 4

BASIC LINUX COMMANDS

ACTIVITY QUESTIONS

1. Command to display the following message as (Use “New line).
   “God!Bless us.
   We are starting Shell Scripting”

2. Get the manual page of ‘ls’ command. Search for the word “alpha”. Find the next occurrence and then find the previous occurrence.

3. Read your name from the keyboard and display it.

4. Create the directory structure dir1/dir4 and dir1/dir2/dir3 with a single command and then change directory to dir3.

5. Create some files using Vim.
   - Creating a file using vim:
   - Start vim by typing vim filename
   - To insert text press i
   - Now start editing text. Add new text or delete unwanted text.
   - One can press Esc and type :wq to save changes to a file and exit from vim

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1/dir2/dir3
+dialog_con        +mksession         +smartindent       +writebackup
+diff              +modify_fname      +sodium            -X11
+digraphs          +mouse             -sound             -xfontset
-dnd               -mouseshape        +spell             -xim
-ebcdic            +mouse_dec         +startuptime       -xpm
+emacs_tags        +mouse_gpm         +statusline        -xsmp
+eval              -mouse_jsbterm     -sun_workshop      -xterm_clipboard
+ex_extra          +mouse_netterm     +syntax            -xterm_save
+extra_search      +mouse_sgr         +tag_binary
-farsi             -mouse_sysmouse    -tag_old_static
   system vimrc file: "$VIM/vimrc"
     user vimrc file: "$HOME/.vimrc"
 2nd user vimrc file: "~/.vim/vimrc"
      user exrc file: "$HOME/.exrc"
       defaults file: "$VIMRUNTIME/defaults.vim"
  fall-back for $VIM: "/usr/share/vim"
Compilation: gcc -c -I. -Iproto -DHAVE_CONFIG_H -Wdate-time -g -O2 -ffile-prefix-map=/build/vim-tMLXRZ/vim-8.2.3995=. -flt
o=auto -ffat-lto-objects -flto=auto -ffat-lto-objects -fstack-protector-strong -Wformat -Werror=format-security -D_REENTRA
NT -U_FORTIFY_SOURCE -D_FORTIFY_SOURCE=1
Linking: gcc -Wl,-Bsymbolic-functions -flto=auto -ffat-lto-objects -flto=auto -Wl,-z,relro -Wl,-z,now -Wl,--as-needed -o v
im -lm -ltinfo -lselinux -lsodium -lacl -lattr -lgpm -L/usr/lib/python3.10/config-3.10-x86_64-linux-gnu -lpython3.10 -lcry
pt -ldl -lm -lm
ubuntu@ubuntu:/home$ cd dir1/dir2/dir3
bash: cd: dir1/dir2/dir3: No such file or directory
ubuntu@ubuntu:/home$ cd /home/Desktop/NSA/dir1/dir2/dir3
bash: cd: /home/Desktop/NSA/dir1/dir2/dir3: No such file or directory
ubuntu@ubuntu:/home$ cd /home/ubuntu/Desktop/NSA/dir1/dir2
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2$ cd /home/ubuntu/Desktop/NSA/dir1/dir2/dir3
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ vim a1.txt
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ vim avg.c
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ vim test.py
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$
```

6. Display the current directory.

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1/dir2/dir3
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ pwd
/home/ubuntu/Desktop/NSA/dir1/dir2/dir3
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$
```

7. Listing files and folders.
   a) Listing the contents of dir1 (Qn.4) and all its descendants.

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1/dir2/dir3
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls -R -p /home/ubuntu/Desktop
/home/ubuntu/Desktop:
NSA/  ubiquity.desktop

/home/ubuntu/Desktop/NSA:
dir1/

/home/ubuntu/Desktop/NSA/dir1:
dir2/  dir4/

/home/ubuntu/Desktop/NSA/dir1/dir2:
dir3/

/home/ubuntu/Desktop/NSA/dir1/dir2/dir3:
a1.txt  avg.c  test.py

/home/ubuntu/Desktop/NSA/dir1/dir4:
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$
```

   b) Listing the contents of dir3 (Qn.4) in

   i. Alphabetical order
   ii. Sorted on Time of modification, newest first
   iii. Sorted on Size
   iv. Reverse of all above
   v. Long listing of files Sorted on Size with smallest first and size
   vi. Displayed in human readable form

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1/dir2/dir3
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls
a1.txt  avg.c  test.py
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls -lt
total 12
-rw-rw-r-- 1 ubuntu ubuntu   25 May 27 02:08 test.py
-rw-rw-r-- 1 ubuntu ubuntu  275 May 27 02:07 avg.c
-rw-rw-r-- 1 ubuntu ubuntu 1012 May 27 02:05 a1.txt
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls -S
a1.txt  avg.c  test.py
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls -r
test.py  avg.c  a1.txt
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls -lSr
total 12
-rw-rw-r-- 1 ubuntu ubuntu   25 May 27 02:08 test.py
-rw-rw-r-- 1 ubuntu ubuntu  275 May 27 02:07 avg.c
-rw-rw-r-- 1 ubuntu ubuntu 1012 May 27 02:05 a1.txt
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls -h
a1.txt  avg.c  test.py
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$
```

8. Execute ls and store the output to a file lsoutput.

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1/dir2/dir3
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls -lr > lsoutput
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cat lsoutput
total 12
-rw-rw-r-- 1 ubuntu ubuntu   25 May 27 02:08 test.py
-rw-rw-r-- 1 ubuntu ubuntu    0 May 27 02:14 lsoutput
-rw-rw-r-- 1 ubuntu ubuntu  275 May 27 02:07 avg.c
-rw-rw-r-- 1 ubuntu ubuntu 1012 May 27 02:05 a1.txt
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$
```

9. Display the file
   a. starting with the first 10 lines and

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1/dir2
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2$ more -10 cal.c
    #include <stdio.h>
    #include <conio.h>
    void main()
    {
        int n1,n2,add,sub,prod,quot,remain;
        clrscr();
        printf("\nENTER NUMBER 1: " );
        scanf("%d",&n1);
        printf("\nENTER NUMBER 2: " );
        scanf("%d",&n2);
--More--(39%)
```

   b. starting with the 10th line with provision for

      i. Scrolling Up

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1/dir2
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2$ ls
cal.c  dir3
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2$ more +10 cal.c
        scanf("%d",&n2);
        add=n1+n2;
        sub=n1-n2;
        prod=n1*n2;
        quot=n1/n2;
        remain=n1%n2;
        printf("\nADDITION OF THE NUMBERS: %d",add);
        printf("\nSUBSTRACTION OF THE NUMBERS: %d",sub);
        printf("\nPRODUCTION OF THE NUMBERS: %d",prod);
        printf("\nQUOTIENT OF THE NUMBERS: %d",quot);
        printf("\nREMAINDER OF THE NUMBERS: %d",remain);
        getch();
    }
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2$
```

      ii. Scrolling Up and Down

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1/dir2
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2$ less +10 cal.c

[3]+  Stopped                 less +10 cal.c
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2$
```

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1/dir2
        scanf("%d",&n2);
        add=n1+n2;
        sub=n1-n2;
        prod=n1*n2;
        quot=n1/n2;
        remain=n1%n2;
        printf("\nADDITION OF THE NUMBERS: %d",add);
        printf("\nSUBSTRACTION OF THE NUMBERS: %d",sub);
        printf("\nPRODUCTION OF THE NUMBERS: %d",prod);
        printf("\nQUOTIENT OF THE NUMBERS: %d",quot);
        printf("\nREMAINDER OF THE NUMBERS: %d",remain);
        getch();
    }
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
cal.c (END)
```

10. Execute **ls -l** and add the output to lsoutput, at the end.

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1/dir2/dir3
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2$ cd dir3
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls -l >> lsoutput
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cat lsoutput
total 12
-rw-rw-r-- 1 ubuntu ubuntu   25 May 27 02:08 test.py
-rw-rw-r-- 1 ubuntu ubuntu    0 May 27 02:14 lsoutput
-rw-rw-r-- 1 ubuntu ubuntu  275 May 27 02:07 avg.c
-rw-rw-r-- 1 ubuntu ubuntu 1012 May 27 02:05 a1.txt
total 16
-rw-rw-r-- 1 ubuntu ubuntu 1012 May 27 02:05 a1.txt
-rw-rw-r-- 1 ubuntu ubuntu  275 May 27 02:07 avg.c
-rw-rw-r-- 1 ubuntu ubuntu  219 May 27 02:14 lsoutput
-rw-rw-r-- 1 ubuntu ubuntu   25 May 27 02:08 test.py
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$
```

11. Execute ls -l and feed the result to less command, to scroll through the directory listing.

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1/dir2/dir3
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls -l |less

[4]+  Stopped                 ls --color=auto -l | less
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$
```

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1/dir2/dir3
total 16
-rw-rw-r-- 1 ubuntu ubuntu 1012 May 27 02:05 a1.txt
-rw-rw-r-- 1 ubuntu ubuntu  275 May 27 02:07 avg.c
-rw-rw-r-- 1 ubuntu ubuntu  438 May 27 02:26 lsoutput
-rw-rw-r-- 1 ubuntu ubuntu   25 May 27 02:08 test.py
(END)
```

12. Copy the file file1 to newfile.
    - If newfile already exists, it should be replaced.
    - If newfile already exists, it should not be replaced.

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1/dir2/dir3
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cd /home
ubuntu@ubuntu:/home$ cd /home/ubuntu/Desktop/NSA/dir1/dir2/dir3
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls
a1.txt  avg.c  file1  file2  lsoutput  test.py
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cat file1
Hello,
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cp file1 newfile
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cat newfile
Hello,
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cat >> file1
Good Morning
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cat file1
Hello,
Good Morning
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cp -n file1 newfile
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cat newfile
Hello,
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$
```

- If newfile already exists, it should be replaced, but only with the consent of the user.
- If newfile already exists, it should be replaced only if its contents is older than that of file1.

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1/dir2/dir3
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cat file1
Hello,
Good Morning
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cp -i file1 newfile
cp: overwrite 'newfile'? y
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cat newfile
Hello,
Good Morning
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cat >> file1
 to all
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cp -u file1 newfile
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cat newfile
Hello,
Good Morning
 to all
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$
```

- Even if newfile is read only.
- Create a link instead of copying.

```
ubuntu@ubuntu: ~
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ chmod 444 newfile
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cat >> file1
Students
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cp -f file1 newfile
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cat newfile
Hello,
Good Morning
 to all
Students
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cp -s file1 newfile
cp: cannot create symbolic link 'newfile' to 'file1': File exists
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cp -S file1 newfile
cp: missing destination file operand after 'newfile'
Try 'cp --help' for more information.
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cp -s file1 newf
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls
a1.txt  avg.c  file1  file2  lsoutput  newf  newfile  test.py
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls -l
total 28
-rw-rw-r-- 1 ubuntu ubuntu 1012 May 27 02:05 a1.txt
-rw-rw-r-- 1 ubuntu ubuntu  275 May 27 02:07 avg.c
-rw-rw-r-- 1 ubuntu ubuntu   37 May 27 02:43 file1
-rw-rw-r-- 1 ubuntu ubuntu   11 May 27 02:31 file2
-rw-rw-r-- 1 ubuntu ubuntu  438 May 27 02:26 lsoutput
lrwxrwxrwx 1 ubuntu ubuntu    5 May 27 02:46 newf -> file1
-rw-rw-r-- 1 ubuntu ubuntu   37 May 27 02:43 newfile
-rw-rw-r-- 1 ubuntu ubuntu   25 May 27 02:08 test.py
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cd
ubuntu@ubuntu: $ cp -R dir
```

- Copy the entire directory tree from dir1 of Qn.4 to a new directory dir5

```
ubuntu@ubuntu: ~/Desktop/NSA
ubuntu@ubuntu:~$ cd /home/ubuntu/Desktop/NSA/dir1/dir2/dir3
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ cd
ubuntu@ubuntu:~$ cp -R dir1 dir5
cp: cannot stat 'dir1': No such file or directory
ubuntu@ubuntu:~$ cd /home/ubuntu/Desktop
ubuntu@ubuntu:~/Desktop$ cd NSA
ubuntu@ubuntu:~/Desktop/NSA$ cp -R dir1 dir5
ubuntu@ubuntu:~/Desktop/NSA$ ls
dir1  dir5
ubuntu@ubuntu:~/Desktop/NSA$
```

13. Create a new directory, dir6 inside dir1
    a) Move all files in dir5 into it.
    b) Delete all files where the name starts with a vowel character, upper or lower case.

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1/dir2/dir3
ubuntu@ubuntu:~/Desktop/NSA$ mkdir -p dir1/dir6
ubuntu@ubuntu:~/Desktop/NSA$ mv dir5 dir1/dir6
ubuntu@ubuntu:~/Desktop/NSA$ cd dir1/dir6
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir6$ ls
dir5
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir6$ cd ..
ubuntu@ubuntu:~/Desktop/NSA/dir1$ cd dir2/dir3
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ mv newfile oldfile
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls
a1.txt  avg.c  file1  file2  lsoutput  newf  oldfile  test.py
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$
```

    c) Delete all files where the name is at least 3 characters long.
    d) Delete all hidden folders, and files.

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1/dir2/dir3
ubuntu@ubuntu:~/Desktop/NSA$ mv file1 dir1/dir6/file3
mv: cannot stat 'file1': No such file or directory
ubuntu@ubuntu:~/Desktop/NSA$ mv file1 dir1/dir6/file3
ubuntu@ubuntu:~/Desktop/NSA$ cd dir1/dir2/dir3
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ rm [a,e,i,o,u,A,E,I,O,U]*
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls
file1  file2  lsoutput  newf  test.py
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ rm ???*
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls -A
cal.c
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ lls -a
Command 'lls' not found, did you mean:
  command 'ls' from deb coreutils (8.32-4.1ubuntu1)
  command 'hls' from deb hfsutils (3.2.6-15build2)
  command 'lvs' from deb lvm2 (2.03.11-2.1ubuntu4)
Try: sudo apt install <deb name>
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls -a
.  ..  cal.c
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ rm -rf .*
rm: refusing to remove '.' or '..' directory: skipping '.'
rm: refusing to remove '.' or '..' directory: skipping '..'
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$ ls -a
.  ..  cal.c
ubuntu@ubuntu:~/Desktop/NSA/dir1/dir2/dir3$
```

14. Create a file testfile1 using Vim
    vim testfile1
    i. Set line number vim testfile1 Press esc
       Type :set number

    ii. Type your name and address with district and pincode

    iii. Copy paste the contents 10 times.

        1. Place the cursor in the desired location(at beginning) .
        2. Press Esc key then press ‘v’ [To change to visual mode] .
        3. Press the ‘y’ key followed by movement command ,‘$’ .
        4. Move the cursor to the location where you want to paste the contents.
        5. Press ‘10p’ to paste the contents.

    iv. Replace all occurrence of your district with a neighbouring district.
        1. Press Esc key [To change to normal mode].
        2. Type “:%s/Erode/Dindigul/gi”

15. Create 2 files testfile2 and testfile3 using Vim.
    a. Modify the permissions of testfile2 using symbolic mode.
        i. Add read permission to others
        ii. Revoke write from owner
        iii. Set only execute to Group.
        iv. Add write to owner, revoke read from others and set read only to group.
        v. Set read and write to all.

    b. Modify the permissions of testfile3 using numeric mode
        i. Set read and write to all.
        ii. set read,write and execute to owner, read and execute to group and read only to others.

    c. Set the permissions of testfile2 the same as that of testfile3

16. Use head and tail piped with cat /etc/passwd to display the details of

a. The first 12 users in the system.

b. The last 7 users in the system.

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1
ubuntu@ubuntu:~/Desktop/NSA/dir1$ cat /etc/passwd|head -12
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
ubuntu@ubuntu:~/Desktop/NSA/dir1$ cat /etc/passwd|tail -7
colord:x:123:130:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:124:131::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:125:132:PulseAudio daemon,,,:/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:126:65534::/run/gnome-initial-setup/:/bin/false
hplip:x:127:7:HPLIP system user,,,:/run/hplip:/bin/false
gdm:x:128:134:Gnome Display Manager:/var/lib/gdm3:/bin/false
ubuntu:x:999:999:Live session user,,,:/home/ubuntu:/bin/bash
ubuntu@ubuntu:~/Desktop/NSA/dir1$
```

c. All but the first 3.

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1
ubuntu@ubuntu:~/Desktop/NSA/dir1$ cat /etc/passwd | tail +4
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:102:105::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:103:106:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
syslog:x:104:111::/home/syslog:/usr/sbin/nologin
```

d. All but the last 5.

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1
ubuntu@ubuntu:~/Desktop/NSA/dir1$ cat /etc/passwd | head -n -5
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:102:105::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:103:106:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
syslog:x:104:111::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:112:TPM software stack,,,:/var/lib/tpm:/bin/false
```

e. Only the 9th.

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1
ubuntu@ubuntu:~/Desktop/NSA/dir1$ cat /etc/passwd | head -9|tail -1
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
ubuntu@ubuntu:~/Desktop/NSA/dir1$
```

17. Use grep to
    a. Display all lines in a file that contains the string “abc”
    b. Display all lines in a file that does not contain the string “abc”

```
ubuntu@ubuntu: ~/Desktop/NSA/dir1
ubuntu@ubuntu:~/Desktop/NSA/dir1$ cat > txtfile
abc
abc is a string
these are english alphabetsubuntu@ubuntu:~/Desktop/NSA/dir1$ grep abc txtfile
abc
abc is a string
ubuntu@ubuntu:~/Desktop/NSA/dir1$ grep -v abc txtfile
these are english alphabets
```

18. Using expr
    a. Read two integers X and Y. Display the sum, difference, product, quotient and remainder of these variables.
    b. Read a string, S, a position, p and a length l. Display the substring of length l starting at position p from the string S.

```
lab@lab-Lenovo-IdeaPad-Z400:~/dir1/dir6/dir5/dir2$ read -p "enter two numbers: " x y
enter two numbers: 12 5
lab@lab-Lenovo-IdeaPad-Z400:~/dir1/dir6/dir5/dir2$ echo "sum is : `expr $x + $y`"
sum is : 17
lab@lab-Lenovo-IdeaPad-Z400:~/dir1/dir6/dir5/dir2$ echo "Difference is : `expr $x - $y`"
Difference is : 7
lab@lab-Lenovo-IdeaPad-Z400:~/dir1/dir6/dir5/dir2$ echo "Product is : `expr $x \* $y`"
Product is : 60
lab@lab-Lenovo-IdeaPad-Z400:~/dir1/dir6/dir5/dir2$ echo "Quotient is : `expr $x / $y`"
Quotient is : 2
lab@lab-Lenovo-IdeaPad-Z400:~/dir1/dir6/dir5/dir2$ echo "Reminder is : `expr $x % $y`"
Reminder is : 2
lab@lab-Lenovo-IdeaPad-Z400:~/dir1/dir6/dir5/dir2$ read -p "enter a string: " s
enter a string: notebook
lab@lab-Lenovo-IdeaPad-Z400:~/dir1/dir6/dir5/dir2$ read -p "enter the position: " p
enter the position: 5
lab@lab-Lenovo-IdeaPad-Z400:~/dir1/dir6/dir5/dir2$ read -p "enter the length: " l
enter the length: 4
lab@lab-Lenovo-IdeaPad-Z400:~/dir1/dir6/dir5/dir2$ echo "Substring: `expr substr $s $p $l`"
Substring: book
lab@lab-Lenovo-IdeaPad-Z400:~/dir1/dir6/dir5/dir2$
```

19. a. Add a normal user, user1. Create (if it does not exist) the folder /user1 and set /user1 as the home directory of user1. Also set /bin/bash as the login shell (Use a single command).

    b. Modify the user account of user1, to expire it after a specific date.

```
user@user-VirtualBox: /
user@user-VirtualBox:~$ sudo useradd -s /bin/bash -m -d /home/u_ser1 u_ser1
[sudo] password for user:
user@user-VirtualBox:~$ sudo change -l u_ser1
sudo: change: command not found
user@user-VirtualBox:~$ sudo chage -l u_ser1
Last password change                                    : Jun 06, 2023
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
user@user-VirtualBox:~$ sudo change -E 2022-06-9 u_ser1
sudo: change: command not found
user@user-VirtualBox:~$ sudo chage -E 2022-06-9 u_ser1
user@user-VirtualBox:~$ sudo chage -l u_ser1
Last password change                                    : Jun 06, 2023
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : Jun 09, 2022
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
user@user-VirtualBox:~$ sudo chown -R u_ser1:u_ser1 /home/Documents/dir1/dir2
chown: cannot access '/home/Documents/dir1/dir2': No such file or directory
user@user-VirtualBox:~$ sudo chown -R u_ser1:u_ser1 /Documents/dir1/dir2
chown: cannot access '/Documents/dir1/dir2': No such file or directory
user@user-VirtualBox:~$ sudo chown -R u_ser1:u_ser1 /home/user/Documents/dir1/
user@user-VirtualBox:~$ cd ..
user@user-VirtualBox:/home$ ls -l
total 8
drwxr-x--- 18 user   user   4096 Jun  6 21:22 user
drwxr-x---  2 u_ser1 u_ser1 4096 Jun  6 22:06 u_ser1
user@user-VirtualBox:/home$ cd Documents
bash: cd: Documents: No such file or directory
user@user-VirtualBox:/home$ cd ..
user@user-VirtualBox:/$ cd Documents
bash: cd: Documents: No such file or directory
user@user-VirtualBox:/$
```

c. Change the owner and group of the directory tree from dir2 and all its contents to user1.

d. Delete the user account user1

i. By retaining the home folder

ii. By deleting the home folder

20. Miscellaneous

a. Using tar create a tar.gz file of the folder dir1 of Qn.4 with the name mydir.tar.gz.

b. Extract the contents of mydir.tar.gz to dir6 of Qn.14.

c. Use top to display processes sorted on

i. Process id

1. Type top
2. Press N

```
top - 21:34:09 up 49 min,  2 users,  load average: 0.09, 0.09, 0.09
Tasks: 231 total,   1 running, 230 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.3 us,  0.0 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   4685.0 total,   1654.5 free,   1713.1 used,   1613.8 buff/cache
MiB Swap:   4096.0 total,   4096.0 free,      0.0 used.   2971.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1642 user      20   0 5081956 525256 168664 S   1.7  10.9   2:36.78 gnome-shell
   2259 user      39  19  784244  32700  18944 S   0.3   0.7   0:01.05 tracker-miner-f
   2782 user      20   0  643092  57804  46032 S   0.3   1.2   0:05.48 gnome-terminal-
   5375 user      20   0   23024   5504   3328 R   0.3   0.1   0:00.02 top
      1 root      20   0  168440  12724   8884 S   0.0   0.3   0:01.77 systemd
      2 root      20   0       0       0      0 S   0.0   0.0   0:00.00 kthreadd
      3 root       0 -20       0       0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0       0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      5 root       0 -20       0       0      0 I   0.0   0.0   0:00.00 slub_flushwq
      6 root       0 -20       0       0      0 I   0.0   0.0   0:00.00 netns
      8 root       0 -20       0       0      0 I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri
     10 root       0 -20       0       0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
     11 root      20   0       0       0      0 I   0.0   0.0   0:00.00 rcu_tasks_kthread
     12 root      20   0       0       0      0 I   0.0   0.0   0:00.00 rcu_tasks_rude_kthread
     13 root      20   0       0       0      0 I   0.0   0.0   0:00.00 rcu_tasks_trace_kthread
     14 root      20   0       0       0      0 S   0.0   0.0   0:00.03 ksoftirqd/0
     15 root      20   0       0       0      0 I   0.0   0.0   0:00.78 rcu_preempt
     16 root      rt   0       0       0      0 S   0.0   0.0   0:00.02 migration/0
     17 root     -51   0       0       0      0 S   0.0   0.0   0:00.00 idle_inject/0
     18 root      20   0       0       0      0 I   0.0   0.0   0:00.18 kworker/0:1-events
     19 root      20   0       0       0      0 S   0.0   0.0   0:00.00 cpuhp/0
     20 root      20   0       0       0      0 S   0.0   0.0   0:00.00 cpuhp/1
     21 root     -51   0       0       0      0 S   0.0   0.0   0:00.00 idle_inject/1
     22 root      rt   0       0       0      0 S   0.0   0.0   0:00.26 migration/1
     23 root      20   0       0       0      0 S   0.0   0.0   0:00.03 ksoftirqd/1
     24 root      20   0       0       0      0 I   0.0   0.0   0:00.22 kworker/1:0-events
     25 root       0 -20       0       0      0 I   0.0   0.0   0:00.00 kworker/1:0H-events_highpri
     26 root      20   0       0       0      0 S   0.0   0.0   0:00.00 cpuhp/2
     27 root     -51   0       0       0      0 S   0.0   0.0   0:00.00 idle_inject/2
     28 root      rt   0       0       0      0 S   0.0   0.0   0:00.26 migration/2
     29 root      20   0       0       0      0 S   0.0   0.0   0:00.03 ksoftirqd/2
     31 root       0 -20       0       0      0 I   0.0   0.0   0:00.00 kworker/2:0H-events_highpri
     32 root      20   0       0       0      0 S   0.0   0.0   0:00.00 cpuhp/3
     33 root     -51   0       0       0      0 S   0.0   0.0   0:00.00 idle_inject/3
```

ii.CPU%

1. Type top
2. Press P

```
top - 21:33:24 up 48 min,  2 users,  load average: 0.09, 0.09, 0.10
Tasks: 229 total,   1 running, 228 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.3 us,  0.0 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   4685.0 total,   1652.0 free,   1716.6 used,   1611.8 buff/cache
MiB Swap:   4096.0 total,   4096.0 free,      0.0 used.   2968.3 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   5351 user      20   0   23024   5632   3456 R   0.3   0.1   0:00.03 top
   5350 root      20   0   16884   2176   2048 S   0.0   0.0   0:00.00 anacron
   5295 root      20   0       0       0      0 I   0.0   0.0   0:00.00 kworker/u12:4-ext4-rsv-conversion
   5294 root      20   0       0       0      0 I   0.0   0.0   0:00.00 kworker/u12:1-flush-8:0
   5258 root      20   0       0       0      0 I   0.0   0.0   0:00.00 kworker/4:1-events
   5250 root      20   0       0       0      0 I   0.0   0.0   0:00.01 kworker/3:1-events
   5248 root      20   0       0       0      0 I   0.0   0.0   0:00.00 kworker/1:2-cgroup_destroy
   5191 root      20   0       0       0      0 I   0.0   0.0   0:00.00 kworker/u12:5-flush-8:0
   5163 root      20   0       0       0      0 I   0.0   0.0   0:00.10 kworker/u12:0-events_power_efficient
   5153 root      20   0       0       0      0 I   0.0   0.0   0:00.00 kworker/0:0-cgroup_destroy
   5107 root      20   0       0       0      0 I   0.0   0.0   0:00.00 kworker/5:1-events
   4230 user      20   0 2519536 242452 107860 S   0.0   5.1   0:24.80 gnome-text-edit
   4219 root      20   0       0       0      0 I   0.0   0.0   0:00.18 kworker/u12:2-events_power_efficient
   3539 user      20   0  868160  26368  14848 S   0.0   0.5   0:00.06 snap
   3269 root      20   0       0       0      0 I   0.0   0.0   0:00.16 kworker/u12:3-events_unbound
   3137 root      20   0       0       0      0 I   0.0   0.0   0:00.42 kworker/5:2-events
   3133 root      20   0       0       0      0 I   0.0   0.0   0:00.00 kworker/2:0
   2801 user      20   0   19604   4864   3584 S   0.0   0.1   0:00.06 bash
   2782 user      20   0  642860  57676  46032 S   0.0   1.2   0:05.30 gnome-terminal-
   2751 user      20   0  396408   8064   7296 S   0.0   0.2   0:00.01 gvfsd-recent
   2672 user      20   0 3029392  64592  48508 S   0.0   1.3   0:00.87 gjs
   2661 user      20   0 2673300 319420 124752 S   0.0   6.7   0:14.34 nautilus
   2552 user      20   0  575972  31492  24196 S   0.0   0.7   0:00.31 update-notifier
   2449 root      20   0  479412  33728  28232 S   0.0   0.7   0:00.62 fwupd
   2373 user      20   0  244796   6400   5888 S   0.0   0.1   0:00.01 gvfsd-metadata
   2302 user      20   0  160652   2076   1792 S   0.0   0.0   0:00.20 VBoxClient
   2301 user      20   0   2854B   1564   1280 S   0.0   0.0   0:00.00 VBoxClient
   2299 user      20   0  227352   2824   2560 S   0.0   0.1   0:02.67 VBoxClient
   2294 user      20   0   2854B   1544   1280 S   0.0   0.0   0:00.00 VBoxClient
   2288 user      20   0  226836   3080   2688 S   0.0   0.1   0:01.50 VBoxClient
   2286 user      20   0   2854B   1544   1280 S   0.0   0.0   0:00.00 VBoxClient
   2266 user      20   0 1576112  86496  66068 S   0.0   1.8   0:00.65 mutter-x11-fram
   2259 user      39  19  784244  33000  18944 S   0.0   0.7   0:00.99 tracker-miner-f
   2251 user      20   0  275376  24700  18116 S   0.0   0.5   0:00.21 ibus-x11
```

d. Use ps to display

i. Processes associated with the current terminal.
ii. All processes in the system.

```
user@user-VirtualBox: ~
er@user-VirtualBox:~/Documents$ ps -T
  PID    SPID TTY          TIME CMD
 2801    2801 pts/0    00:00:00 bash
 5424    5424 pts/0    00:00:00 ps
er@user-VirtualBox:~/Documents$ ps -A
  PID TTY          TIME CMD
    1 ?        00:00:01 systemd
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 rcu_gp
    4 ?        00:00:00 rcu_par_gp
    5 ?        00:00:00 slub_flushwq
    6 ?        00:00:00 netns
    8 ?        00:00:00 kworker/0:0H-events_highpri
   10 ?        00:00:00 mm_percpu_wq
   11 ?        00:00:00 rcu_tasks_kthread
   12 ?        00:00:00 rcu_tasks_rude_kthread
   13 ?        00:00:00 rcu_tasks_trace_kthread
   14 ?        00:00:00 ksoftirqd/0
   15 ?        00:00:00 rcu_preempt
   16 ?        00:00:00 migration/0
   17 ?        00:00:00 idle_inject/0
   18 ?        00:00:00 kworker/0:1-events
   19 ?        00:00:00 cpuhp/0
   20 ?        00:00:00 cpuhp/1
   21 ?        00:00:00 idle_inject/1
   22 ?        00:00:00 migration/1
   23 ?        00:00:00 ksoftirqd/1
   24 ?        00:00:00 kworker/1:0-events
   25 ?        00:00:00 kworker/1:0H-events_highpri
   26 ?        00:00:00 cpuhp/2
   27 ?        00:00:00 idle_inject/2
   28 ?        00:00:00 migration/2
   29 ?        00:00:00 ksoftirqd/2
   31 ?        00:00:00 kworker/2:0H-events_highpri
   32 ?        00:00:00 cpuhp/3
   33 ?        00:00:00 idle_inject/3
   34 ?        00:00:00 migration/3
   35 ?        00:00:00 ksoftirqd/3
   37 ?        00:00:00 kworker/3:0H-kblockd
   38 ?        00:00:00 cpuhp/4
   39 ?        00:00:00 idle_inject/4
```

e. Use df to display the storage available in each partition in human readable form.

```
user@user-VirtualBox:~/Documents$ df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           469M  1.6M  467M   1% /run
/dev/sda2        25G   13G   11G  57% /
tmpfs           2.3G     0  2.3G   0% /dev/shm
tmpfs           5.0M  8.0K  5.0M   1% /run/lock
tmpfs           469M  116K  469M   1% /run/user/1000
user@user-VirtualBox:~/Documents$
```

Result:
Commands executed successfully and output is verified.

LAB CYCLE: 3 **DATE: 04-04-2024**

EXPERIMENT NO: 5

FILE SYSTEM HIERARCHY, PERMISSION, CONFIGURATION FILES AND LOG FILES

AIM

a. File system hierarchy in a common Linux distribution
b. File and device permissions,
c. Study of system configuration files in /etc,
d. Familiarizing log files for system events, user activity, network events.

a) File system hierarchy in a common Linux distribution

The Linux File Hierarchy Structure or the File System 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.

1. The Root Directory

- Every single file and directory starts from the root directory.
- The only root user has the right to write under this directory.
- /root is the root user's home directory, which is not the same as /.

2. /bin- User binaries

- 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, cat, ls.

3. /boot – Static Boot Files

- Boot loader files
- Kernel initrd, vmlinux, grub files are located under /boot

Example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic

4. /dev-Device Files

- Contains device files
- These include terminal devices, usb, or any device attached to the system.

- 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.

5. /etc- Configuration Files

- Contains configuration files required by all programs.
- This also contains start-up and shutdown shell scripts used to start/stop individual programs.

Example: /etc/resolv.conf

b) File and device permissions

Since Ubuntu is a multi-user operating system, so it has security to prevent people from accessing each other's confidential files. When you do so, each file will be listed on a separate line in a long format. $ls -l

1. There's a lot of information in those lines.
2. The first character = '-', which means it's a file
   'd', which means it's a directory.
3. The next nine characters = (rw-r–r–) show the security
4. The next column shows the owner of the file. (Here it is `root`)

5. The next column shows the group owner of the file.
6. The next column shows the size of the file in bytes.
7. The next column shows the date and time the file was last modified.
8. Last Column = File_name or Directory_name.

```
ubuntu@ubuntu:~/Desktop/NSA$ ls -l
total 0
drw-rw-r-- 5 ubuntu ubuntu 100 May 27 02:53 dir1
ubuntu@ubuntu:~/Desktop/NSA$ man ls chmod
```

Security Permissions

rwx rwx rwx

user group other

Letters Definition

‘r’: “read” the file’s contents.

‘w’: “write”, or modify, the file’s contents.

‘x’ : “execute” the file. This permission is given only if the file is a program.

Operators Definition

`+`: Add permissions

`-`: Remove permissions

`=`: Set the permissions to the specified values

Reference Class Description

`u`(user): The user permissions apply only to the owner of the file or directory, they will not impact the actions of other users.

`g`(group): The group permissions apply only to the group that has been assigned to the file or directory, they will not affect the actions of other users.

`o`(others): The other permissions apply to all other users on the system, this is the permission group that you want to watch the most.

`a`(All three): All three (owner, groups, others)

Changing security permissions

The command you use to change the security permissions on files is called “chmod“, which stands for “change mode” because the nine security characters are collectively called the security “mode” of the file.

Adding execute permission

chmod o+x filename

Change multiple permissions at once

chmod ugo-rwx filename

Revoke execute(x) permission from others(o)

chmod ug+rw,o-x filename

Octal values

When Linux file permissions are represented by numbers, it's called numeric mode. In numeric mode, a three-digit value represents specific file permissions (for example, 744.) These are called octal values. The first digit is for owner permissions, the second digit is for group permissions, and the third is for other users. Each permission has a numeric value assigned to it:

r (read): 4

w (write): 2

x (execute): 1

In the permission value 744, the first digit corresponds to the user, the second digit to the group, and the third digit to others. By adding up the value of each user classification, you can find the file permissions.

For example, a file might have read, write, and execute permissions for its owner, and only read permission for all other users. That looks like this:

Owner: rwx = 4+2+1 = 7

Group: r-- = 4+0+0 = 4

Others: r-- = 4+0+0 = 4

The results produce the three-digit value 744.

c) Study of system configuration files in /etc

Linux configuration files enable the Kernel to know about users, their login state, and manage file permissions and user groups. Most of the configuration files in Linux are usually under the /etc folder./etc(et-see) directory is where a Linux configuration file live.

The configuration files are static and cannot be executable. They are used to control the operation of various Linux programs.

Types of Configuration Files in /etc

1. Access files
2. Booting and login/logout
3. File system
4. System administration
5. Networking
6. System commands
7. Daemons

1. Access files

Access files tell the network domain how to look up host names.

Access files include:

- /etc/hosts.config
- /etc/hosts
- /etc/hosts.allow
- /etc/hosts.deny

2. Booting and login/logout

These files contain configuration information for booting up the system.

/etc/issue & /etc/issue.net

/etc/rc.d/rc

/etc/rc.d/rc.sysinit

/etc/rc.d/rc/rc X.d

3. File system

Linux provides /proc, a virtual file system that can be used to display various system data structures and parameters.

Many programs access this file system to gain statistical information about the system, such as the devices mounted on the system, the memory usage, etc.

/etc/mtab

/etc/fstab

/etc/mtools.conf

4. System Administration

This group of files contains information about the users and user groups, as well as the file permissions and credentials of all users. These files include the following configuration files:

/etc/group

/etc/nologin

/etc/passwd

/etc/securetty

/etc/shadow

5. Networking

/etc/networks

Lists names and addresses of networks accessible from the network to which the machine is connected. Used by route command. Allows use of name for network.

/etc/protocols

Lists the currently available protocols

/etc/services

Translates network service names to port number/protocol.

6. System commands

System commands are meant to be used exclusively by the system. Programs like login or bash are all system commands. These are important files containing information about the system commands and include the following files:

/etc/lilo.conf

/etc/logrotate.conf

/etc/ld.so.conf

/etc/inittab

/etc/termcap

7. Daemons

Daemons are programs that run in the background without user interference.

These are often related to networking stacks, where they wait for connections to arrive so that they can provide services through them.

/etc/syslogd.conf

/etc/httpd.conf

/etc/conf.modules

d) Familiarizing log files for system events, user activity,network events.

Log files are a set of records that Linux maintains for the administrators to keep track of important events.
They contain messages about the server, including the kernel, services and applications running on it.
Linux provides a centralized repository of log files that can be located under the /var/log directory.
When issues arise, analyzing log files is the first thing an administrator needs to do.
How can Linux Log files help

1. Troubleshooting

When something goes wrong on a Linux system, logs can help pinpoint the issue.

By examining system logs, application logs, and service logs, it’s possible to identify errors, warnings, and other messages that indicate what went wrong.

2. Diagnosing Performance Issues

System logs can help identify performance issues like memory leaks or disk I/O bottlenecks.

Examining application logs can also help identify performance issues with specific applications.

3. Monitoring System Health

Linux logs can be used to monitor system health and detect issues before they become critical.
By monitoring system logs, administrators can identify trends and patterns that could indicate a problem is brewing.

4. Compliance and Auditing

Many organizations are required to maintain logs for compliance and auditing purposes. Linux logs can help organizations meet these requirements by providing a record of system activity.

5. Security

Linux logs are an essential tool for monitoring and detecting security issues.

System logs can be used to detect unauthorized access attempts, while application logs can help identify suspicious activity within specific applications.

By monitoring logs, administrators can quickly identify and respond to security incidents.
The log files generated in a Linux environment can typically be classified into four different categories:

- Application Logs
- Event Logs
- Service Logs
- System Logs

1. System Logs

These logs contain information about the system's operation, such as boot messages, kernel messages, and hardware events.
System logs are essential for troubleshooting system issues, and monitoring system performance.

2. Application Logs

These logs contain information about the behaviour of an application, including errors, warnings, and other messages.
Application logs are used to diagnose problems with applications and to analyze application performance.

3. Service Logs

These logs contain information about services running on the system, including network services and daemons.
Service logs are used to monitor service activity, and optimize service performance.

4.Event Logs

These logs contain information about events on the system, such as user logins, system shutdowns, and security events.

Event logs are used to audit system activity, track user activity, and investigate security incidents

Which Linux log files to monitor?

Monitoring and analyzing all of them can be a challenging task.

The sheer volume of logs can sometimes make it frustrating just to drill down and find the right file that contains the desired information.

1. /var/log/messages
2. /var/log/auth.log
3. /var/log/secure
4. /var/log/boot.log
5. /var/log/dmesg
6. /var/log/kern.log
7. /var/log/faillog
8. /var/log/cron
9. /var/log/yum.log
10. /var/log/maillog or /var/log/mail.log
11. var/log/httpd/
12. /var/log/mysqld.log or /var/log/mysql.log

Result:

Familiarisation with file system hierarchy, permission, configuration files and log files has been done successfully.

LAB CYCLE: 4 **DATE: 11-04-2024**

EXPERIMENT NO: 6

SHELL SCRIPTING

AIM

a) 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.
b) Study of startup scripts, login and logout scripts, familiarity with systemd and system 5 init scripts is expected

a) 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.

Shell scripting refers to the process of writing and executing scripts using a shell, which is a command-line interface or command interpreter on Unix-like operating systems. The most commonly used shell scripting languages are Bash (Bourne Again Shell) and sh (Bourne Shell). Shell scripts can automate tasks, execute commands, and perform various operations on a Unix or Linux system.

Here are some key concepts and features of shell scripting:

1. Script file extension: Shell scripts typically have file extensions like ".sh" to indicate that they are shell scripts.

2. Shebang line: The first line of a shell script starts with a shebang (#!) followed by the path to the shell interpreter. For example, #!/bin/bash specifies that the script should be interpreted using Bash.

3. Variables: Shell scripts can define and use variables. Variables are typically assigned values using the assignment operator (=).

For example :

```
name="John"
age=25
```

4. Command execution: Shell scripts can execute commands using various command-line utilities. Commands can be executed directly or stored in variables for later use. For example:

```
ls

result=$(ls)
```

5. Control structures: Shell scripting supports control structures like conditionals (if-else statements) and loops (for loops, while loops). These structures allow you to control the flow of execution based on conditions or iterate over a set of values.

6. Input/output redirection: Shell scripts can redirect input and output streams. For example, you can redirect the output of a command to a file using the ">" operator or read input from a file using the "<" operator.

7. Functions: Shell scripts can define functions to encapsulate reusable code blocks. Functions help in modularizing the script and improving code organization.

8. Command-line arguments: Shell scripts can accept command-line arguments. These arguments can be accessed using special variables like $1, $2, etc., where $1 represents the first argument, $2 represents the second argument, and so on.

9. Environment variables: Shell scripts can access and modify environment variables, which are system-wide variables that hold information about the environment in which the script is running.

10. Error handling: Shell scripts can handle errors using conditional statements and error codes returned by commands. This allows scripts to take appropriate actions based on the success or failure of commands.

Shell scripting is a powerful tool for automation, system administration, and repetitive tasks in Unix-like operating systems. It provides a flexible and efficient way to execute commands, manipulate data, and control the behaviour of the shell.

Environment Variables

Environment variables are an essential part of shell scripting as they allow you to store and retrieve information that can be used by various processes and scripts running in the shell. Here are some key points about environment variables in shell scripting:

1. Defining Environment Variables: Environment variables are typically defined using uppercase letters. To set an environment variable, you can use the export command followed by the variable name and its value. For example:

```
export MY_VAR="Hello, World!"
```

2. Accessing Environment Variables: To access the value of an environment variable, you can use the $ symbol followed by the variable name. For example:

```
echo $MY_VAR
```

3. System-defined Environment Variables: The shell automatically sets some environment variables that contain useful information. Examples include:

$HOME: The current user's home directory.

$PATH: A list of directories where the shell looks for executable files.

$USER: The username of the current user.

$PWD: The present working directory.

4. Shell Script-defined Environment Variables: You can define your own environment variables within shell scripts. These variables are local to the script by default, but you can export them to make them available to other processes. For example:

```
# Define a local variable

MY_LOCAL_VAR="Local variable"

# Export the variable to make it available to other processes

export MY_LOCAL_VAR
```

5. Using Environment Variables in Scripts: Environment variables can be used within shell scripts just like any other variables. Here's an example:

```
#!/bin/bash

echo "The value of MY_VAR is: $MY_VAR"
```

6. Unsetting Environment Variables: To remove an environment variable, you can use the unset command followed by the variable name. For example:

```
unset MY_VAR
```

7. Persistent Environment Variables: Environment variables defined in a shell session are not persistent and will be lost when the session ends. To make them persistent, you can define them in shell startup files such as .bashrc or .bash_profile (for Bash) or in the system-wide /etc/environment file.

Variables

Variables in shell scripting are used to store and manipulate data within a script. Here are some important points about variables in shell scripting:

1. Variable Declaration: In shell scripting, variables are typically declared without specifying a data type. You can assign a value to a variable using the syntax variable_name=value. For example:

```
name="John"

age=25
```

2. Variable Naming: Variable names in shell scripting can contain letters (a-z, A-Z), digits (09), and underscores (_). They must start with a letter or an underscore. Variable names are casesensitive, so my_var and MY_VAR are considered different variables.

3. Accessing Variable Values: To access the value of a variable, you can use the $ symbol followed by the variable name. For example:

```
echo $name
```

4. Variable Scope: By default, variables in shell scripts have a global scope and can be accessed from anywhere within the script. If you want to limit the scope of a variable to a specific block of code, you can use functions.

5. Special Variables: Shell scripting provides some predefined variables that have special meanings. Examples include:

$0: The name of the script itself.

$1, $2, etc.: Command-line arguments passed to the script.

$#: The number of command-line arguments.

$?: The exit status of the last command.

$$: The process ID of the current script.

6. Variable Manipulation: You can perform various operations on variables, such as concatenation, substring extraction, and arithmetic calculations. Some commonly used operators and syntax include:

Concatenation: result=$var1$var2

Substring extraction: substring=${string:position:length}

Arithmetic calculations: result=$((num1 + num2))

7. Quoting Variables: When accessing variable values, it's a good practice to enclose them in double quotes ("") to handle cases where the value contains spaces or special characters. For example:

```
echo "$name is $age years old."
```

8. Variable Assignment from Command Output: You can assign the output of a command to a variable using command substitution. The syntax is variable=$(command). For example:

```
date=$(date +%Y-%m-%d)
```

These are some fundamental aspects of variables in shell scripting. They allow you to store and manipulate data, perform calculations, and pass information to and from your script.

Control Structures

In shell scripting, control structures are used to control the flow of execution based on certain conditions or to repeat a set of instructions. The commonly used control structures in shell scripting include:

1. if-else: The if-else statement allows you to perform different actions based on a condition. It has the following syntax:

```
if condition
then
  # statements to execute if condition is true
else
  # statements to execute if condition is false
fi
```

2. for loop: The for loop allows you to iterate over a list of values or elements. It has the following syntax:

```
for variable in list
do
  # statements to execute for each iteration
done
```

The variable takes the value of each item in the list on each iteration.

3. while loop: The while loop allows you to repeat a set of statements as long as a condition is true. It has the following syntax:

```
while condition
do
# statements to execute while condition is true
done
```

The loop continues until the condition evaluates to false.

4. until loop: The until loop is similar to the while loop, but it continues until a condition becomes true. It has the following syntax:

```
until condition
do
# statements to execute until condition becomes true
```

```
done
```

The loop continues until the condition evaluates to true.

5. case statement: The case statement allows you to perform different actions based on the value of a variable. It has the following syntax:

```
case variable in
pattern1)
    # statements to execute for pattern1
    ;;
pattern2)
  # statements to execute for pattern2
  ;;
pattern3)
  # statements to execute for pattern3
  ;;
 *)
# statements to execute for all other patterns
;;
esac
```

The value of variable is compared against each pattern, and the corresponding statements are executed for the matching pattern.

These control structures provide you with the flexibility to conditionally execute code, iterate over a set of values, and make decisions based on different conditions in shell scripting.

Aliasing

In Linux, an alias is a shortcut that references a command. An alias replaces a string that invokes a command in the Linux shell with another user-defined string. Aliases are mostly used to replace long commands, improving efficiency and avoiding potential spelling errors.

The alias command provides a string value that replaces a command name when it is encountered. The alias command lets you create shortcuts for long commands, making them easier to remember and use. It will have the same functionality as if the whole command is run.

How to Create Your Own Linux Commands?

Using the alias command, you'll be able to create your own commands. It's so simple to create your own command.

Here's the syntax for the alias command:

alias [alias-name[=string]...]

Let's look at an example of creating your own command.

Let's assume you want to create a command called cdv, and entering the command in the terminal should take you to the Videos directory.

Usually, to navigate to a directory, we use cd command. To navigate to Videos we need to use cd Videos as shown in the below screenshot:

```
arunachalamb@ArunachalamB:~$ ls
'2FA Codes'            'Backup from Samsung'   Documents      'Git Project'    Music         Pictures     scripts     Test       'Wonders of the World'
 Android               'Backup Mobile'         Downloads      IdeaProjects     my-alias.sh   Projects     snap        Videos      yarn.lock
 AndroidStudioProjects  Books                   Dropbox        install_git.sh   Nextcloud     Public       SSH         wget-log    z-commands
 Apps                   Desktop                 'Easy Zip.zip'  install_goi.sh  notes.txt     Recordings   Templates   Wonderla
arunachalamb@ArunachalamB:~$ cd Videos/
arunachalamb@ArunachalamB:~/Videos$
```

Let's create our command called cdv to navigate to the Videos directory. To achieve that, you have to enter the following command in your terminal:

alias cdv="cd Videos“

```
arunachalamb@ArunachalamB:~$ alias cdv="cd Videos"
arunachalamb@ArunachalamB:~$
```

We have created our command. From the above screenshot, you can see that it does not return anything.

Run the cdv command on your terminal to see what happens:

```
arunachalamb@ArunachalamB:~$ cdv
arunachalamb@ArunachalamB:~/Videos$
```

How to View Created Alias Commands

You can view all your alias commands by appending the -p flag to the alias command like this:

alias -p

```
arunachalamb@ArunachalamB:~$ alias -p
alias a='xdotool key ctrl+shift+t'
alias ad='~/Android/Sdk/emulator/emulator -list-avds'
alias adbr='adb reverse tcp:8081 tcp:8081'
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo error)"
'\'')"'
alias b='cd ..'
alias c='code ./'
alias cdb='cd -'
alias cdv='cd Videos'
```

How to Remove an Alias Command in Linux

Pass your alias name to the unalias command as an argument to remove the alias command.
unalias alias_name

```
arunachalamb@ArunachalamB:~$ unalias cdv
arunachalamb@ArunachalamB:~$ alias -p
alias a='xdotool key ctrl+shift+t'
alias ad='~/Android/Sdk/emulator/emulator -list-avds'
alias adbr='adb reverse tcp:8081 tcp:8081'
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo error)"
'\'')"'
alias b='cd ..'
alias c='code ./'
alias cdb='cd -'
alias clear_goi='rm -rf ~/.nvm/versions/node/v14.17.6/bin/goi; rm -rf ~/.nvm/versions/nod
```

How to Remove All Alias Commands in Linux

We have a command to achieve that:

unalias –a

```
arunachalamb@ArunachalamB:~$ alias -p
alias cdd='cd Downloads'
alias cddu='cd Documents'
alias cdv='cd Videos'
arunachalamb@ArunachalamB:~$ unalias -a
arunachalamb@ArunachalamB:~$ alias -p
arunachalamb@ArunachalamB:~$
```

If you create an alias command, it'll be active only for the particular instance of the terminal. It'll not be created permanently, so you won't be able to access it in two different terminal windows unless you run the alias command on both terminals.

b) Study of startup scripts, login and logout scripts, familiarity with systemd and system 5 init scripts is expected

Startup scripts, login scripts, and logout scripts are important components of the initialization process in a Unix-like operating system. They allow administrators to automate tasks and configure system behavior during system startup, user login, and user logout. Familiarity with systemd and System V init scripts is crucial for managing these processes efficiently. Let's explore each of these topics in more detail:

1. Startup Scripts:

Startup scripts are executed during the boot process to initialize the system. In modern Linux distributions that use systemd as the init system, startup scripts are typically managed through systemd unit files. These unit files define services, targets, and other units that control the behavior of the system. The main directory for systemd unit files is /etc/systemd/system/.

2. Login Scripts:

Login scripts are executed when a user logs into a system. These scripts provide a way to set environment variables, configure user-specific settings, and perform additional tasks upon login. The specific login script executed depends on the user's shell and the system configuration. For example, in the Bash shell, the login script is usually ~/.bash_profile or ~/.bash_login for login shells, and ~/.bashrc for non-login shells.

3. Logout Scripts:

Logout scripts are executed when a user logs out of a system. These scripts allow for cleanup tasks, such as removing temporary files or logging session statistics. The specific logout script executed depends on the user's shell and the system configuration. In Bash, the logout script is typically ~/.bash_logout.

4. Systemd:

Systemd is a modern init system and service manager that has replaced the traditional System V init system in many Linux distributions. Systemd provides enhanced features for managing services, including parallel startup, dependency-based service control, and centralized logging. Systemd unit files, such as service unit files (.service), target unit files (.target), and timer unit files (.timer), are used to define and control services.

5. System V Init Scripts:

System V init scripts were the traditional initialization mechanism used in Unix-like systems before the introduction of systemd. Init scripts are stored in /etc/init.d/ directory and are typically written in shell scripting languages (e.g., Bash). These scripts are responsible for starting, stopping, and managing system services and can be run manually or automatically during the system startup.

Familiarity with both systemd and System V init scripts is important for understanding and managing the initialization process in different Linux distributions. The choice of which init system to use depends on the specific distribution and its configuration. It's beneficial to have knowledge of both to be able to work with various systems effectively.

Result:

The scripting familiarization process equipped a solid understanding of Bash syntax, environment variables, variable manipulation, control constructs (such as if-else statements, for and while loops), aliases and functions, and accessing command line arguments passed to shell scripts.

Shell script program questions.

1. Write a script to show current date,time and current directory.

SOURCE CODE:

```
#!/bin/bash

echo "Current date is `date`"

echo "Current directory is `pwd`"
```

OUTPUT:

```
Current date is Sun May 19 05:13:36 PM UTC 2024
Current directory is /home
```

2. Write shell script to find reverse of a number.

SOURCE CODE:

```
#!/bin/bash
echo "Enter a number:"
read n
# Check if the input is a valid number
if ! [[ "$n" =~ ^[0-9]+$ ]]; then
    echo "Invalid input. Please enter a valid number."
    exit 1
fi
reversed=0
while [ $n -gt 0 ]; do
    reversed=$((reversed * 10))
    digit=$((n % 10))
    reversed=$((reversed + digit))
    n=$((n / 10))
done
echo "Reversed number is $reversed"
```

OUTPUT:

```
Enter a number:
563
Reversed number is 365
```

3. Write a script to find largest among three numbers.

SOURCE CODE:

```
#!/bin/bash
echo "Enter the 1st number:"
read m
# Validate input for the first number
if [ -z "$m" ] || ! [[ "$m" =~ ^[0-9]+$ ]]; then
    echo "Error: '$m' is not a valid number."
    exit 1
fi
echo "Enter the 2nd number:"
read n
# Validate input for the second number
if [ -z "$n" ] || ! [[ "$n" =~ ^[0-9]+$ ]]; then
    echo "Error: '$n' is not a valid number."
     exit 1
fi
echo "Enter the 3rd number:"
read o
# Validate input for the third number
if [ -z "$o" ] || ! [[ "$o" =~ ^[0-9]+$ ]]; then
    echo "Error: '$o' is not a valid number."
    exit 1
fi
# Find the largest number
if [ $m -gt $n ] && [ $m -gt $o ]; then
    echo "Largest number is $m"
```

```
elif [ $n -gt $m ] && [ $n -gt $o ]; then
    echo "Largest number is $n"
else
    echo "Largest number is $o"
fi
```

OUTPUT:

```
Enter the 1st number:
55
Enter the 2nd number:
96
Enter the 3rd number:
77
Largest number is 96
```

4. Write a script to check whether a number is armstrong or not.

SOURCE CODE:

```
#!/bin/bash
# Prompt the user to enter a number
echo "Enter a number: "
read c
# Store the original number in a variable
x=$c
sum=0
r=0
n=0
# Loop through each digit of the number
while [ $x -gt 0 ]; do
   r=$(( $x % 10 ))
   n=$(( $r * $r * $r ))
   sum=$(( $sum + $n ))
   x=$(( $x / 10 ))
done
# Check if the sum of cubes of digits is equal to the original number
if [ $sum -eq $c ]; then
   echo "It is an Armstrong Number."
else
```

```
    echo "It is not an Armstrong Number."
fi
```

OUTPUT:

```
Enter a number:
153
It is an Armstrong Number.
```

5. Write a script to check password and login

SOURCE CODE:

```
#!/bin/bash

# Prompt the user to enter a username

read -p 'Username: ' user

# Prompt the user to enter a password (and hide input)

read -sp 'Password: ' pass

echo

# Check if the entered username and password match predefined values

if [[ $user == "Admin" && $pass == "admin123" ]]; then

    echo -e "Welcome! You are successfully logged in\n"

else

    echo -e "Unsuccessful login\n"

fi
```

OUTPUT:

```
Username: Admin
Password: admin123
Welcome! You are successfully logged in
```

6. Write a script to count the prime numbers in specific range

SOURCE CODE:

```
#!/bin/bash
echo "Enter a lower limit"
read i
echo "Enter a upper limit"
read limit
echo "prime numbers upto $limit are :"
while [ $i -le $limit ]
do
   flag=1
   j=2
while [ $j -lt $i ]
 do
   rem=$(( $i % $j ))
   if [ $rem -eq 0 ]
   then
      flag=0
   break
    fi
    j=$(( $j+1 ))
done
if [ $flag -eq 1 ]
then
echo "$i"
fi
i=$(( $i+1 ))
done
```

OUTPUT:

```
Enter a lower limit
5
Enter a upper limit
50
prime numbers upto 50 are :
5
7
11
13
17
19
23
29
31
37
41
43
47
```

7. Write a script to convert the contents of agiven file from uppercase to lowercase and also count the number of lines,words and characters of the resultant file. Also display the resultant file in descending order.

SOURCE CODE:

```
#!/bin/bash
getFile() {
# Reading txtFileName to convert its content
echo -n "Enter File Name: "
read -r filename
# Checking if file exists
if [ ! -f "$filename" ]; then
    echo "File '$filename' does not exist."
     exit 1
fi
```

```
}
echo "1. Uppercase to Lowercase"
echo "2. Count the number of characters, words, lines"
echo "3. Exit"
echo -n "Enter your Choice (1-3): "
read -r ch

case "$ch" in
   1)
      # Function Call to get File
      getFile
      # Converting to lowercase if user chooses 1
      echo "Converting Upper-case to Lower-Case"
      tr '[:upper:]' '[:lower:]' < "$filename"
   ;;
 2)
      echo "Enter the filename: "
      read -r file
      # Check if file exists
      if [ ! -f "$file" ]; then
         echo "File '$file' does not exist."
         exit 1
      fi
     # Counting characters, words, and lines
     c=$(wc -c < "$file")
     w=$(wc -w < "$file")
     l=$(wc -l < "$file")
     echo "Number of characters in $file is $c"
     echo "Number of words in $file is $w"
   echo "Number of lines in $file is $l"
  ;;
```

```
    *)
        # Exiting for all other cases
        echo "Exiting..."
         exit
        ;;
esac
```

OUTPUT:

```
Enter your Choice(1-3):^Cexam@cec-H110M-S2:~/mca_nsa$ ./fileop.sh
1. Uppercase to Lowercase
2. Count the number of characters,words,lines
3. Exit
Enter your Choice(1-3):1
Enter File Name:sample.txt
Converting Upper-case to Lower-Case
welcome to linux operating system.
this is second line.
this is third line.
exam@cec-H110M-S2:~/mca_nsa$ ./fileop.sh
1. Uppercase to Lowercase
2. Count the number of characters,words,lines
3. Exit
Enter your Choice(1-3):2
Enter the filename
sample.txt
Number of characters in sample.txt is 76
Number of words in sample.txt is 13
Number of lines in sample.txt is 3
exam@cec-H110M-S2:~/mca_nsa$ ./fileop.sh
1. Uppercase to Lowercase
2. Count the number of characters,words,lines
3. Exit
Enter your Choice(1-3):3
Exiting...
exam@cec-H110M-S2:~/mca_nsa$
```

8. Write a script to perform following basic math operation as:

Addition, subtraction, multiplication, division

SOURCE CODE:

```
#!/bin/sh
echo "Enter the two numbers to perform arithmetic operations"
read a b
val=`expr $a + $b`
echo "a + b : $val"
val=`expr $a - $b`
echo "a - b : $val"
val=`expr $a \* $b`
echo "a * b : $val"
```

```
if [ $a -gt $b ]
then
   val=`expr $a / $b`
   echo "a / b : $val"
else
   val=`expr $b / $a`
   echo "b / a : $val"
fi
```

OUTPUT:

```
Enter the two numbers to perform arithmetic operations
5 6
a + b : 11
a - b : -1
a * b : 30
b / a : 1
```

9. Read 3 marks of a student and find the average. Display the grade of the student based on the average. (if..then..elif..fi)

S >= 90%

A < 90%, but >= 80%

B < 80%, but >= 60%

P < 80%, but >= 40%

F < 40%

SOURCE CODE:

```
#! /bin/bash
read -p "Enter three marks out of 100 each : " m1 m2 m3
s=$(($m1+$m2+$m3))
avg=$(($s / 3|bc))
echo -e "Average: $avg"
if [ $avg -ge 90 ]
then
```

```
echo "Grade: S"
elif [[ $avg -lt 90 && $avg -ge 80 ]]
then
echo "Grade: A"
elif [[ $avg -lt 80 && $avg -ge 60 ]]
then
echo "Grade: B"
elif [[ $avg -lt 80 && $avg -ge 40 ]]
then
echo "Grade: P"
else
echo "Grade: F"
fi
```

OUTPUT:

```
Enter three marks out of 100 each : 88 66 99
Average: 84
Grade: A
```

10. Read the name of an Indian state and display the main language according to the table.

For other states, the output may be "Unknown". Use "|" to separate states with same language *(case..esac)*

State Main	Language
Andhra Pradesh	Telugu
Assam	Assamese
Bihar	Hindi
Himachal Pradesh	Hindi
Karnataka	Kannada
Kerala	Malayalam
Lakshadweep	Malayalam
Tamil Nadu	Tamil

SOURCE CODE:

```
#!/bin/bash

echo -e "1. Andhra Pradesh \n2. Assam \n3. Bihar \n4. Karnataka \n5. Kerala \n6. Tamil Nadu \n7. Exit"

read -p "Enter the Indian state number: " state


case $state in

1)

   echo "Language: Telugu"

;;

2)

   echo "Language: Assamese"

;;

3)

   echo "Language: Hindi"

;;

4)

   echo "Language: Kannada"

;;

5)

   echo "Language: Malayalam"

;;

6)

   echo "Language: Tamil"

;;

7)

    echo "Exiting..."

    exit 0

;;

*)

   echo "Language: Unknown"

;;

esac
```

OUTPUT:

```
1. Andhra Pradesh
2. Assam
3. Bihar
4. Karnataka
5. Kerala
6. Tamil Nadu
7. Exit
Enter the Indian state number: 5
Language: Malayalam
```

11. Change the home folder of all users whose name start with stud from /home/username to/usr/username. Also change the password of username to username123 (e.g., /home/stud25 changes to /usr/stud25 and his/her password changes to stud25123) - (Use for ..in)

SOURCE CODE:

```
#!/bin/bash
result=$(grep -E 'stud' /etc/passwd)
result=$(echo "${result}" | cut -d: -f1)
echo "$result"
for f in $result; do
   p="${f}123"
   echo "$f:$p" | sudo chpasswd
   sudo usermod -m -d /usr "$f"
done
```

OUTPUT:

```
stud:x:1003:1003::/home:/bin/sh
students:x:1006:1006::/home:/bin/sh
```

```
user@user-VirtualBox:~/shellpg$ bash usermod.sh
stud students
usermod: directory /usr exists
usermod: directory /usr exists
```

```
stud:x:1003:1003::/usr:/bin/sh
students:x:1006:1006::/usr:/bin/sh
```

12. Read a number and display the multiplication table of the number up to 10 lines. - (Use for((..)))

SOURCE CODE:

```
#!/bin/bash
read -p "Enter a number: " num
echo "Multiplication table of $num : "
for (( i=1; i<=10; i++ ))
do
   val=$(( num * i ))
    echo "$i * $num = $val"
done
```

OUTPUT:

```
Enter a number: 6
Multiplication table of 6 :
1 * 6 = 6
2 * 6 = 12
3 * 6 = 18
4 * 6 = 24
5 * 6 = 30
6 * 6 = 36
7 * 6 = 42
8 * 6 = 48
9 * 6 = 54
10 * 6 = 60
```

13. Read a Decimal number. Convert it to Binary and display the result. - (Use while)

SOURCE CODE:

```
#! /bin/bash
read -p "Enter a decimal number: " n
val=0
power=1
while [ $n -ne 0 ]
do
```

```
    r=`expr $n % 2`
    val=`expr $r \* $power + $val`
    power=`expr $power \* 10`
    n=`expr $n \/ 2`
done
echo "Binary equivalent : $val"
```

OUTPUT:

```
Enter a decimal number: 55
Binary equivalent : 110111
```

RESULT:

Shell script programs have executed successfully, and the output has been verified.

LAB CYCLE: 5 **DATE: 18-04-2024**

EXPERIMENT NO: 7

NETWORKING

AIM

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.

IPv4 networking

IP stands for Internet Protocol and v4 stands for Version Four (IPv4). IPv4 was the primary version brought into action for production within the ARPANET in 1983. IP version four addresses are 32-bit integers which will be expressed in decimal notation.

Parts of IPv4

- Network part: The network part indicates the distinctive variety that's appointed to the network. The network part conjointly identifies the category of the network that's assigned.
- Host Part: The host part uniquely identifies the machine on your network. This part of the IPv4 address is assigned to every host. For each host on the network, the network part is the same, however, the host half must vary.
- Subnet number: This is the nonobligatory part of IPv4. Local networks that have massive numbers of hosts are divided into subnets and subnet numbers are appointed to that.

Characteristics

- IPv4 could be a 32-Bit IP Address.
- IPv4 could be a numeric address, and its bits are separated by a dot.
- The number of header fields is twelve and the length of the header field is twenty.
- It has Unicast, broadcast, and multicast style of addresses.
- IPv4 supports VLSM (Virtual Length Subnet Mask).
- IPv4 uses the Post Address Resolution Protocol to map to the MAC address.
- RIP may be a routing protocol supported by the routed daemon.

- Networks ought to be designed either manually or with DHCP.
- Packet fragmentation permits from routers and causing host.

Some key points about IPv4 networking:

1. Addressing: IPv4 addresses are expressed as four sets of numbers separated by periods, such as 192.168.0.1. Each set represents 8 bits, resulting in a range of 0 to 255 for each set. This gives a total of 2^32 (around 4.3 billion) unique addresses.

2. Classes: IPv4 initially defined three address classes: A, B, and C. Each class had a different range of available addresses, with Class A providing a large number of network addresses but fewer host addresses, and Class C offering a smaller number of network addresses but more host addresses. Later, Classless Inter-Domain Routing (CIDR) was introduced to allow more flexible allocation of IP addresses.

3. Subnetting: Subnetting is the practice of dividing a network into smaller subnetworks or subnets. It allows organizations to efficiently allocate IP addresses by creating smaller networks within a larger network. Subnetting involves borrowing bits from the host portion of an IP address to create a separate network identifier and subnet identifier.

4. Private IP Addresses: Certain IP address ranges are reserved for private networks, which are not directly accessible from the public Internet. The most commonly used private IP address ranges are:

Class A: 10.0.0.0 to 10.255.255.255

Class B: 172.16.0.0 to 172.31.255.255

Class C: 192.168.0.0 to 192.168.255.255

NAT (Network Address Translation): Due to the limited number of available IPv4 addresses, NAT is commonly used to allow multiple devices within a private network to share a single public IP address. NAT translates private IP addresses to a public IP address when communicating with devices on the Internet, and vice versa.

5. IPv6 Transition: With the depletion of available IPv4 addresses, IPv6 (Internet Protocol version 6) was developed to provide a much larger address space. IPv6 uses 128-bit addresses

and supports approximately 3.4 x 10^38 unique addresses. However, IPv4 continues to be used extensively, and the transition to IPv6 is an ongoing process.

6. IPv4 Limitations: The limited number of IPv4 addresses is one of the significant limitations of IPv4 networking. The explosive growth of the Internet and the increasing number of devices connected to it have led to address exhaustion. This issue, along with other limitations of IPv4, such as security vulnerabilities and inefficient routing, prompted the development and adoption of IPv6.

Network Commands

1) Ping:

The ping command is used to test network connectivity between your device and a specific destination IP address or domain name. It sends ICMP (Internet Control Message Protocol) echo requests to the destination and waits for ICMP echo replies. It measures the round-trip time (RTT) and checks for packet loss.

2)Route:

The route command displays and manages the IP routing table on your device. It shows the network destinations and the corresponding gateway addresses used for routing network traffic.

```
$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG    100    0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
```

3)Traceroute:

The traceroute command helps identify the path that network packets take from your device to a specific destination. It displays the IP addresses of intermediate routers or hops along the route and measures the time it takes for packets to reach each hop.

```
prabhakar@Inspiron-3542:~$ traceroute -n google.com
traceroute to google.com (216.58.197.78), 30 hops max, 60 byte packets
 1  192.168.43.45  1.276 ms  2.567 ms  2.894 ms
 2  * * *
 3  10.45.1.230  213.144 ms  208.579 ms  208.552 ms
 4  10.45.8.178  208.533 ms  213.026 ms  224.356 ms
 5  10.45.8.187  224.303 ms  236.195 ms  320.501 ms
 6  * * *
 7  218.248.235.141  337.190 ms  136.746 ms  136.789 ms
 8  218.248.235.142  142.816 ms  138.503 ms  143.977 ms
 9  72.14.211.114  132.807 ms  138.627 ms  126.861 ms
10  * 108.170.253.97  145.860 ms 108.170.253.113  138.521 ms
11  108.170.237.95  138.488 ms 108.170.236.197  132.544 ms 108.170.237.95  132.564 ms
12  216.58.197.78  128.691 ms  128.919 ms  122.617 ms
```

3)nslookup:

The nslookup command is used to query DNS (Domain Name System) servers to retrieve information about a specific domain. It can be used to find the IP address associated with a domain name or obtain other DNS-related information.

```
File  Edit  View  Search  Terminal  Help
student@Comp9:~$ nslookup -type=a google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 172.217.166.174

student@Comp9:~$
```

4)IP:

The ip command is a versatile networking tool available in Linux-based operating systems. It allows you to view and modify network interfaces, routing tables, and other network-related settings.

```
raghav@techie-inspiron-15-3567: ~
File  Edit  View  Search  Terminal  Help
raghav@techie-inspiron-15-3567:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp3s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN grou
p default qlen 1000
    link/ether [redacted] brd ff:ff:ff:ff:ff:ff
3: wlp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default
qlen 1000
    link/ether [redacted] brd ff:ff:ff:ff:ff:ff
    inet [redacted] brd 192.168.43.255 scope global dynamic noprefixroute wlp2s
0
       valid_lft 3521sec preferred_lft 3521sec
    inet6 2405:205:158a:f65e:a578:ec04:e6b:5a36/64 scope global temporary dynamic
       valid_lft 86257sec preferred_lft 14257sec
    inet6 2405:205:158a:f65e:50b7:ad6f:d120:749f/64 scope global dynamic mngtmpaddr no
prefixroute
       valid_lft 86257sec preferred_lft 14257sec
    inet6 fe80::d33:8b10:5be3:afb5/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
raghav@techie-inspiron-15-3567:~$
```

Setting up static and Dynamic IP addresses:

Static IP Address:

1. Identify Network Interface: Open Terminal and use the ip addr command to find the name of your network interface (e.g., eth0, enp0s3).
2. Edit the Network Configuration File: Use a text editor like nano or vim to open the Netplan configuration file located at /etc/netplan/.

```bash
sudo nano /etc/netplan/01-network-manager-all.yaml
```

3. Configure Static IP: Add the following lines to the configuration file, replacing placeholders with your actual network details.

```yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    <interface>:
      addresses: [192.168.1.100/24]  # Your static IP address and subnet mask
      gateway4: 192.168.1.1          # Your gateway IP address
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]  # DNS server IP addresses
```

4. Apply Configuration Changes: Save the changes and exit the text editor. Then, apply the changes.

5. Verify Configuration: Check that the static IP configuration has been applied to your network interface:

```bash
ip addr show <interface>
```

Dynamic IP Address (DHCP):
Dynamic IP addresses are assigned automatically using the DHCP (Dynamic Host Configuration Protocol) server on the network.

To configure dynamic IP addressing, you need to enable DHCP in your device's network settings. It will automatically obtain an IP address, subnet mask, gateway, and DNS server information from the DHCP server.

Concept of Subnets

In computer networking, a subnet refers to a logical subdivision of an IP network. It is a technique used to partition a single large network into smaller, more manageable networks called subnetworks or subnets. The purpose of subnetting is to improve network performance, enhance security, and simplify network administration.

Subnetting involves dividing an IP network into smaller network segments by manipulating the host portion of the IP address. An IP address consists of two parts: the network portion and the host portion. The network portion identifies the network to which a device belongs, while the host portion identifies the specific device within that network.
By using subnetting, you can allocate IP addresses more efficiently and group devices based on their requirements and purposes. Each subnet has its own unique network address, and devices within the same subnet share a common network prefix.

Subnetting also allows for more efficient use of network resources. For example, if you have a large network with thousands of devices, it may not be practical or efficient to

broadcast network traffic to all devices. Subnetting enables you to divide the network into smaller segments, reducing the broadcast domain and limiting the scope of network traffic.

Moreover, subnetting enhances network security by creating isolated segments. Devices within the same subnet can communicate directly with each other, but communication between devices in different subnets requires routing. This isolation helps contain network issues, restrict unauthorized access, and minimize the impact of potential security breaches.

Subnets are identified by their subnet mask, which determines the network and host portions of an IP address. The subnet mask is represented using a series of ones (1) followed by a series of zeros (0). The ones indicate the network portion, while the zeros represent the host portion.

For example, in the IP address 192.168.0.0 with a subnet mask of 255.255.255.0, the first three octets (192.168.0) represent the network portion, and the last octet (0) represents the host portion. This subnet mask allows for 256 possible host addresses within the network.

Subnetting is commonly used in large networks, such as organizations, universities, and internet service providers (ISPs), to efficiently manage IP address allocation and optimize network performance.

CIDR address schemes

CIDR (Classless Inter-Domain Routing) is a method used for allocating IP addresses and routing Internet Protocol (IP) packets. It is an improved addressing scheme that replaces the older classful addressing scheme used in IPv4.

In CIDR, IP addresses are represented in a format that includes both the IP address and a prefix length, separated by a slash ("/") character. The prefix length indicates the number of bits in the IP address that are used as the network prefix. The remaining bits are used to identify hosts within the network.

CIDR allows for more flexible and efficient allocation of IP addresses by allowing variablelength subnet masks. This means that networks can be divided into smaller subnets, which reduces address wastage and allows for more efficient use of available IP addresses.

CIDR notation follows the format: <IP address>/<prefix length>. For example:

192.168.0.0/24 represents a network with an IP address of 192.168.0.0 and a prefix length of 24, indicating that the first 24 bits are used for the network prefix and the remaining 8 bits are available for host addresses. This corresponds to a subnet mask of 255.255.255.0.

10.0.0.0/16 represents a network with an IP address of 10.0.0.0 and a prefix length of 16, indicating that the first 16 bits are used for the network prefix and the remaining 16 bits are available for host addresses. This corresponds to a subnet mask of 255.255.0.0.

Subnet masks

A subnet mask is a 32-bit value used in conjunction with an IP address to determine the network and host portions of the address. It is used in IPv4 networks to divide an IP address into a network address and a host address. Subnet masks are typically represented in dotted decimal notation, similar to IP addresses.

In subnetting, the subnet mask contains a series of consecutive 1s followed by a series of consecutive 0s. The 1s represent the network portion of the address, and the 0s represent the host portion. The length of the consecutive 1s in the subnet mask is known as the prefix length or subnet prefix, which indicates the number of bits in the subnet mask that are used for the network portion.

For example, a subnet mask of 255.255.255.0 (or /24 in CIDR notation) has 24 consecutive 1s, representing the network portion, followed by 8 consecutive 0s for the host portion. This means that the first three octets of the IP address are used for the network, and the last octet is available for host addresses.

Subnet masks are used for various purposes, including:

1. Network addressing: They help identify the network portion of an IP address and determine which devices are on the same network.
2. Routing: Subnet masks are used by routers to determine the best path for forwarding IP packets.
3. Subnetting: Subnet masks are used to divide IP address ranges into smaller subnets for more efficient addressing and routing

iptables

In iptables, you can define rules based on IP addresses, including subnet addresses, to filter or manipulate network traffic. Here are a few examples of how you can use iptables with subnet addresses:

Allow incoming traffic from a specific subnet:

iptables -A INPUT -s 192.168.0.0/24 -j ACCEPT

This rule allows incoming traffic from the subnet 192.168.0.0/24. You can replace 192.168.0.0/24 with the desired subnet address.

Block incoming traffic from a specific subnet:

iptables -A INPUT -s 10.0.0.0/16 -j DROP

This rule blocks incoming traffic from the subnet 10.0.0.0/16. Adjust the subnet address as per your requirement.

Forward traffic from one subnet to another:

iptables -A FORWARD -s 192.168.1.0/24 -d 192.168.2.0/24 -j ACCEPT

This rule allows forwarding of traffic from subnet 192.168.1.0/24 to subnet 192.168.2.0/24. Make sure IP forwarding is enabled on your system for this to work.

Source NAT for a specific subnet:

iptables -t nat -A POSTROUTING -s 172.16.0.0/12 -o eth0 -j MASQUERADE

This rule performs source Network Address Translation (NAT) for the subnet 172.16.0.0/12 on the eth0 interface. Adjust the subnet and interface names accordingly.

These are just a few examples to illustrate how you can use iptables with subnet addresses. You can combine subnet addresses with other matching criteria like ports, protocols, or interface names to create more specific rules to suit your network security or routing requirements.

Remember to save your iptables rules and apply them persistently to ensure they survive system reboots. The exact method for saving and applying iptables rules may vary depending on your Linux distribution.

Setting up a firewall for LAN

To set up a firewall for a local area network (LAN), you can use iptables to define rules that

control incoming and outgoing traffic between your LAN and the internet. Here's a basic approach to setting up a firewall for your LAN:

Define default policies:

Start by setting default policies for the INPUT, FORWARD, and OUTPUT chains to control how traffic is handled if no specific rules match. For example, you can set the default policy to DROP to deny all incoming and forwarded traffic unless explicitly allowed by rules.

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT
```

Allow LAN traffic:

Allow traffic within the LAN by permitting traffic from the LAN subnet on the INPUT and FORWARD chains. Adjust the subnet address based on your LAN's configuration.

```
iptables -A INPUT -s <LAN_subnet> -j ACCEPT
iptables -A FORWARD -s <LAN_subnet> -j ACCEPT
```

Allow specific incoming services:

If you need to allow specific services or ports to be accessible from the internet, such as HTTP (port 80) or SSH (port 22), you can add rules to permit incoming connections for those services.

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT # Allow incoming HTTP traffic
iptables -A INPUT -p tcp --dport 22 -j ACCEPT # Allow incoming SSH traffic
```

Be cautious when opening up ports to the internet and ensure that you understand the security implications.

Enable Network Address Translation (NAT):

If your LAN is using private IP addresses and requires internet access, you'll need to enable NAT to translate the private IP addresses to your public IP address when outgoing traffic goes to the internet. Here's an example:

```
iptables -t nat -A POSTROUTING -s <LAN_subnet> -o <internet_interface> -j
```

MASQUERADE

Replace <LAN_subnet> with your LAN's subnet and <internet_interface> with the network interface connected to the internet (e.g., eth0).

Save and apply rules:

Once you have defined your firewall rules, save them to a file and apply them on system startup or when necessary. The process for saving and applying rules may differ based on your Linux distribution. For example, on Ubuntu, you can use the iptables-persistent package to save and restore rules automatically.

```
# Save rules to a file
iptables-save > /etc/iptables/rules.v4
# Apply rules on system startup
iptables-restore < /etc/iptables/rules.v4
```

Application layer (L7) proxies

Application layer (Layer 7) proxies are network devices or software applications that operate at the highest layer of the OSI model, the application layer. These proxies are designed to handle and manage network traffic based on application-specific protocols and perform various functions such as load balancing, caching, content filtering, and security enforcement. They provide an additional layer of control and visibility over network traffic flowing between clients and servers.

Here are some key features and use cases of Layer 7 proxies:

1. Load balancing: Layer 7 proxies distribute incoming client requests across multiple backend servers based on advanced criteria such as URL, session information, or content type. This helps evenly distribute the workload and improve application performance and scalability.

2. Caching: Proxies can cache frequently accessed content from backend servers, reducing the response time and network bandwidth usage. By serving cached content, they can offload requests from the backend servers and improve overall system performance.

3. SSL/TLS termination: Layer 7 proxies can handle SSL/TLS encryption and decryption, relieving the backend servers from the computational overhead of cryptographic operations. This enables secure communication between clients and servers while offloading the SSL/TLS processing to the proxy.

4. Content filtering and security: Proxies can inspect and filter network traffic based on application-layer attributes, allowing for the enforcement of security policies, content filtering, or access control rules. They can block or allow traffic based on URL, file type, keywords, or other application-specific criteria.
5. Protocol transformation: Layer 7 proxies can translate or transform network protocols to bridge communication gaps between clients and servers that use different protocols or versions. For example, they can convert between HTTP and HTTPS, perform protocol conversions for backend applications, or even bridge different application protocols.
6. Application acceleration: Proxies can optimize and accelerate application performance by implementing techniques such as compression, protocol optimization, and connection pooling. These optimizations reduce bandwidth usage, minimize latency, and improve user experience.
7. API gateway: Layer 7 proxies can act as API gateways, providing a centralized entry point for accessing and managing APIs. They can handle API authentication, rate limiting, request/response transformations, and traffic analytics.

   Popular Layer 7 proxy solutions include Nginx, HA Proxy, Apache Traffic Server, and Envoy. These proxies can be deployed as standalone devices, virtual appliances, or as software applications running on dedicated servers or in cloud environments.

Layer 7 proxies provide granular control and advanced functionality for managing and securing network traffic at the application layer, making them essential components in modern network architectures.

Result:

Familiarisation done successfully.

LAB CYCLE: 5 **DATE: 18-04-2024**

EXPERIMENT NO: 8

NETWORK AND PACKETS ANALYSIS

AIM

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

Network and packet analysis are fundamental techniques used in computer networking and cybersecurity to examine and understand the behavior and communication patterns of networked devices. This analysis provides insights into how data flows across a network, identifies potential issues, and helps in diagnosing and troubleshooting network problems. Here's an overview of network and packet analysis:

Network Analysis:

Network analysis involves studying the overall behaviour, performance, and health of a network. This can include examining factors such as network traffic, latency, bandwidth utilization, and response times. Network analysis tools and techniques help network administrators and engineers ensure that the network is functioning optimally and identify any anomalies or bottlenecks.

Packet Analysis:

Packet analysis is a more granular approach that involves inspecting individual packets of data that travel across a network. Each packet contains information about the source, destination, payload, and other relevant metadata. Packet analysis helps in understanding the details of network communication, diagnosing issues, and detecting potential security threats.

Key Concepts in Packet Analysis:

• Packets: Data sent over a network is divided into packets, which are small units of data that include headers (containing routing information) and payloads (actual data being transmitted).

• Protocols: Networks use communication protocols (such as TCP, UDP, HTTP, etc.) to define how data is formatted, transmitted, and received. Analysing protocols helps in understanding how different devices communicate.

• Capture: To perform packet analysis, you need to capture network traffic using tools like Wireshark, tcpdump, or Microsoft Network Monitor. These tools capture packets in real-time or from packet capture files.

• Inspection: Once packets are captured, analysts can inspect individual packets to gather information about the communication. This includes examining header details, payload contents, and timing information.

• Filters and Display Filters: Packet analysis tools allow you to filter and focus on specific packets of interest based on criteria like source IP, destination port, protocol, etc. This is helpful for narrowing down the analysis to relevant packets.

• Anomalies and Patterns: Analysts look for anomalies in packet behavior, such as unusually high traffic volume, unusual communication patterns, or unexpected packet types. These anomalies can indicate network issues or potential security breaches.

• Troubleshooting: Packet analysis is invaluable for diagnosing network problems. By examining packets, analysts can identify issues like latency, dropped packets, misconfigurations, and more.

• Security Analysis: Packet analysis is a crucial tool for detecting security threats like intrusion attempts, malware infections, data breaches, and more. Security analysts can identify patterns that indicate unauthorized or suspicious activity.

Challenges:

• Complexity: Network traffic can be vast and complex, making it challenging to analyze all packets effectively.

• Encryption: Encrypted traffic makes it difficult to inspect packet contents. While headers can still provide useful information, payload analysis is limited.

• Packet Overload: High network traffic can lead to a massive volume of captured packets, requiring effective filtering and analysis techniques.

• False Positives: Identifying genuine threats among a large volume of packets can be challenging, as false positives can waste time and resources.

Both network and packet analysis play a crucial role in maintaining network performance, ensuring security, and troubleshooting issues in complex network environments.

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

1. tcpdump:

tcpdump is a command-line packet analyzer that allows you to capture and display network packets flowing through a network interface. It's often used for diagnosing network issues, analyzing network traffic, and debugging network protocols. Here's a basic command to capture packets on a specific interface:

sudo tcpdump -i eth0 -n -s 0 -w capture.pcap

-i eth0: Specifies the network interface to capture packets from.

-n: Displays numerical addresses instead of resolving hostnames.

-s 0: Captures the entire packet.

-w capture.pcap: Writes the captured packets to a file in pcap format.

```
vishnu@Ubuntu:~$ tcpdump -D
1.enp0s3 [Up, Running, Connected]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.bluetooth-monitor (Bluetooth Linux Monitor) [Wireless]
5.nflog (Linux netfilter log (NFLOG) interface) [none]
6.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
7.dbus-system (D-Bus system bus) [none]
8.dbus-session (D-Bus session bus) [none]
vishnu@Ubuntu:~$
```

2. Wireshark:

Wireshark is a popular open-source network protocol analyzer used for network analysis, troubleshooting, and packet capturing. It allows network administrators, security professionals, and developers to capture and inspect the packets that flow over a computer network.

1. Packet Capture: Wireshark captures network packets from the network interface or from a saved packet capture file. It can capture packets from various sources, including wired and wireless connections.

2. Packet Inspection: Once packets are captured, Wireshark displays them in a graphical interface, providing a detailed view of the packet contents. It can dissect and interpret a wide range of network protocols, from common ones like TCP, UDP, and HTTP to more specialized protocols used in various applications.

3. Protocol Analysis: Wireshark can help in analyzing protocol behaviors and interactions. By examining the sequence of packets and their content, network analysts can identify issues such as misconfigurations, performance bottlenecks, or security vulnerabilities.

4. Troubleshooting: When network issues arise, Wireshark can assist in diagnosing the problem by showing the actual data being transmitted. It can help identify issues like dropped packets, high latency, incorrect protocol usage, or unexpected traffic patterns.

5. Security Analysis: Wireshark is also valuable for security professionals in detecting malicious activities or potential breaches. It allows for the inspection of network traffic to identify signs of unauthorized access, data exfiltration, or other suspicious behaviors.

6. Performance Optimization: Wireshark can help in optimizing network performance by analyzing the flow of data. By examining packet delays, retransmissions, and other performance metrics, network engineers can identify areas where improvements can be made.

7. Baseline Creation: Network administrators often use Wireshark to establish baselines for network behavior. This involves capturing and analyzing normal network traffic patterns, which can then be used for comparison when troubleshooting abnormal behavior.

8. Protocol Development and Testing: Developers use Wireshark to validate their protocol implementations. By examining how their software communicates over the network, they can ensure that their protocols adhere to specifications and standards.

9. Educational Purposes: Wireshark is widely used in educational settings to teach networking concepts, protocols, and network analysis techniques. Students can gain hands-on experience by capturing and analyzing real network traffic.

3. Netcat (nc):

netcat (often referred to as nc) is a versatile networking utility that can be used for various tasks, including creating network connections, transferring data, and performing service tests. For service testing, you can use it to send and receive data to check the availability of a specific service on a target host.

Here's an example:

To test a TCP service on a specific host and port:

nc <host> <port>

To test a UDP service:

echo "Test data" | nc -u -w1 <host> <port>

-u: Use UDP instead of the default TCP.

-w1: Set a timeout of 1 second.

Remember to replace <host> and <port> with the actual target host and port

Result:

Familiarized with network packet stream analysis conducted using tcpdump and Wireshark.

EXPERIMENT NO: 9 **DATE: 23-04-2024**

LAMP INSTALLATION

Aim:

Setting up the LAMP Stack.

Illustrate the steps involved in installing the LAMP Stack on a Linux machine. Deploy phpMyadmin.

Steps to install LAMP Stack:

A “LAMP” stack is a group of open-source software that is typically installed together to enable a server to host dynamic websites and web apps. This term is actually an acronym which represents the Linux operating system, with the Apache web server. The site data is stored in a MySQL database, and dynamic content is processed by PHP.

Step 1: Installing Apache and Updating the Firewall

Command: sudo apt update

Once the cache has been updated, you can install Apache:

Command: sudo apt install apache2

After entering this command, apt will tell you which packages it plans to install and how much extra disk space they’ll take up. Press Y and hit ENTER to confirm, and the installation will proceed.

To know the active status of apache server:

Command: sudo service apache2 status

```
Enabling conf serve-cgi-bin.
Enabling site 000-default.
Processing triggers for libc-bin (2.23-0ubuntu11.3) ...
Processing triggers for systemd (229-4ubuntu21.27) ...
Processing triggers for ureadahead (0.100.0-19) ...
Processing triggers for ufw (0.35-0ubuntu2) ...
cec@cec-H110M-S2:~$ sudo service apache2 status
● apache2.service - LSB: Apache2 web server
   Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since Wed 2023-07-26 10:00:24 IST; 24s ago
     Docs: man:systemd-sysv-generator(8)
   CGroup: /system.slice/apache2.service
           ├─4117 /usr/sbin/apache2 -k start
           ├─4120 /usr/sbin/apache2 -k start
           └─4121 /usr/sbin/apache2 -k start

Jul 26 10:00:23 cec-H110M-S2 systemd[1]: Starting LSB: Apache2 web server...
Jul 26 10:00:23 cec-H110M-S2 apache2[4095]:  * Starting Apache httpd web server
Jul 26 10:00:23 cec-H110M-S2 apache2[4095]: AH00558: apache2: Could not reliably
Jul 26 10:00:24 cec-H110M-S2 apache2[4095]:  *
Jul 26 10:00:24 cec-H110M-S2 systemd[1]: Started LSB: Apache2 web server.
lines 1-16/16 (END)
cec@cec-H110M-S2:~$ sudo ufw app list
Available applications:
```

Adjust the Firewall to Allow Web Traffic

Make sure that your firewall allows HTTP and HTTPS traffic. You can check that UFW has an application profile for Apache like so:

Command: sudo ufw app list

```
cec@cec-H110M-S2:~$ sudo ufw app list
Available applications:
  Apache
  Apache Full
  Apache Secure
  CUPS
  OpenSSH
```

If you look at the Apache Full profile details, you'll see that it enables traffic to ports 80 and 443:

Command: sudo ufw app info "Apache Full"

```
cec@cec-H110M-S2:~$ sudo ufw app info "Apache Full"
Profile: Apache Full
Title: Web Server (HTTP,HTTPS)
Description: Apache v2 is the next generation of the omnipresent Apache web
server.

Ports:
  80,443/tcp
```

To allow incoming HTTP and HTTPS traffic for this server, run:

Command: sudo ufw allow "Apache Full"

You can do a spot check right away to verify that everything went as planned by visiting your server's public IP address in your web browser.

Step 2: <u>Installing MySQL</u>

MySQL is a database management system. Basically, it will organize and provide access to databases where your site can store information.

Command: sudo apt install mysql-server

Enter Y to continue.

When the installation is complete, run a simple security script that comes pre-installed with MySQL which will remove some dangerous defaults and lock down access to your database system. Start the interactive script by running:

Command: sudo mysql_secure_installation

Command: sudo mysql

Step 3: Installing PHP

PHP is the component of your setup that will process code to display dynamic content. It can run scripts, connect to your MySQL databases to get information, and hand the processed content over to your web server so that it can display the results to your visitors.

Once again, leverage the apt system to install PHP. In addition to the php package, you'll also need libapache2-mod-php to integrate PHP into Apache, and the php-mysql package to allow PHP to connect to MySQL databases. Run the following command to install all three packages and their dependencies:

Command: sudo apt install php libapache2-mod-php php-mysql

Changing Apache's Directory Index (Optional)

Command: sudo nano /etc/apache2/mods-enabled/dir.conf

After this, restart the Apache web server in order for your changes to be recognized. You can do that with the following command:

Command: sudo systemctl restart apache2

check on the status of the apache2 service using systemctl:

Command: sudo systemctl status apache2

Installing PHP Extensions

To extend the functionality of PHP, you have the option to install some additional modules. To see the available options for PHP modules and libraries, pipe the results of apt search into less, a pager which lets you scroll through the output of other commands:

Command: apt search php- | less

Step4: Testing PHP Processing on your Web Server

In order to test that your system is properly configured for PHP, create a PHP script called info.php. In order for Apache to find this file and serve it correctly, it must be saved to your web root directory.

Create the file at the web root you created in the previous step by running:

Command: sudo nano /var/www/your_domain/info.php

This will open a blank file. Add the following text, which is valid PHP code:

When you are finished, save and close the file.

Result

Installed LAMP and deployed phpMyAdmin successfully.

EXPERIMENT NO: 10 **DATE: 23-04-2024**

LARAVEL INSTALLATION

Aim:

Installation and configuration of Laravel

Laravel is a powerful and elegant PHP web application framework that streamlines web application development through an MVC (Model-View-Controller) architecture. Created by Taylor Otwell, Laravel has gained immense popularity due to its simplicity and ease of use, making it a top choice for web developers worldwide.

Steps to install Laravel:

1. Install Apache Web Server

First, update your package index and install Apache by running the following commands in the Terminal:

sudo apt update sudo apt install apache2

After installation, enable the Apache service and start it:

sudo systemctl enable apache2 sudo systemctl start apache2

2. Install PHP

To install PHP, use the following command in the Terminal:

sudo apt install php libapache2-mod-php php-mbstring php-xmlrpc php-soap php-gd php-xml php-cli php-zip php-bcmath php-tokenizer php-json php-pear

3. Install MariaDB

MariaDB is an open-source relational database management system. Install it by running the following command:

sudo apt install mariadb-server

After installation, secure your MariaDB installation by running:

sudo mysql_secure_installation

Follow the on-screen instructions to set up a root password and other security settings.

4. Install Composer

Composer is a dependency management tool for PHP. Download and install Composer by executing the following commands in the Terminal:

curl -sS https://getcomposer.org/installer | php

sudo mv composer.phar

sudo chmod +x /usr/local/bin/composer

5. Verify PHP, MariaDB, and Composer Installations

Ensure that PHP, MariaDB, and Composer are installed correctly by checking their versions with the following commands:

php -v mysql --version composer -V

6. Install Laravel Using Composer

Now, install Laravel by running the following command in the Terminal:

composer create-project --prefer-dist laravel/laravel app-name

Replace app-name above with whatever you wish.

Then go to the newly created project directory:

cd app-name

7. Run the Laravel Server

To launch the development server for the project, go to the project directory and run the specified command.

php artisan serve

To access your Laravel application, run this command to launch a local development server on port 8000. Then, open your web browser and navigate to:

http://127.0.0.1:8000

If everything has been set up correctly, you will be able to view the default Laravel welcome page which confirms that your application is up and running. Now Laravel is installed on your Linux system!

Result

Installed Laravel successfully.

EXPERIMENT NO: 11 **DATE: 25-04-2024**

SOFTWARE FROM SOURCE CODE

AIM

Installing software from source code.

- Source code software must be compiled and installed.
- Usually comes in a compressed archive, called a tarball with .tar or .tar.gz ending.
- Archive includes source, configure script, makefile and install scripts.

Package Managers

- Automate the installation. removal, and management of the software applications.
- Only track software installed using the package manager.
- Similar to Add/Remove programs control panel in MS Windows

> Configure Script:

- Inspects system for requirements and configures the “makefile”.

>Make:

- Automates the compilation of programming source code for the target system.
- "makefile" defines the necessary steps to build the application.
- They are far from perfect.
- There is no central database to track applications installed with make.
- Removal of applications may or may not be supported by the make file.
- "makefile" contains installation parameters, variables, and setup instructions.
- "make" and "make install" commands are run to compile and install software.

>Make Command:

- Source code distributed as "gzipped tarballs"
- After unpacking the code, you must check the README file specific install instructions.

$ configure

$ make

$ make install

Installation

1. Open Linux terminal and enter

   sudo apt update

   This command is used to install the latest versions of the packages currently installed on the user's system from the sources enumerated in /etc/apt/sources. The installed packages which have new packages available are retrieved and installed.

2. Now Enter:

   sudo apt install build-essential

   This command installs the build-essential package, which is a meta-package including essential tools required for compiling software written in C and C++. These tools include gcc, g++, make, and other necessary libraries and header files.

3. Change Directory
   Navigate to the directory where you've downloaded and extracted the No-IP DUC source code. If you've followed the previous steps, you're already in the correct directory, which is /usr/local/src.

   cd /usr/local/src

4. Download No-IP DUC Source Code
   Use the wget command to download the No-IP DUC source code archive from the provided URL:

   sudo wget http://www.noip.com/client/linux/noip-duc-linux.tar.gz

5. Extract the No-IP DUC Source Code and Navigate to the Extracted Directory
   Use the tar command to extract the contents of the downloaded No-IP DUC source code archive, and then navigate to the extracted directory:

   sudo tar xf noip-duc-linux.tar.gz && cd noip-2.1.9-1/

6. Execute *sudo make install*

The make install command is used to copy the built program, its libraries, and documentation to the correct locations on your system. This step is crucial in the installation process as it ensures that the software is properly installed and accessible.

Here's how you execute it:

sudo make install

By running this command with elevated privileges (sudo), you allow the installation process to copy the necessary files to system directories. Once completed, the software should be installed and ready for use on your system.

Result

The software was successfully built and installed using CMake.

EXPERIMENT NO: 12 **DATE: 25-04-2024**

HYPERVISORS AND VIRTUAL MACHINE

Aim:

Familiarisation with Hypervisors and Virtual Machines

VIRTUAL MACHINES

Virtual Machines (VMs) are digital emulations of physical computers that enable running multiple operating systems on a single physical machine. Each VM operates independently, with its own virtualized hardware, including CPU, memory, storage, and network interfaces. Managed by a hypervisor software, VMs offer strong isolation, allowing activities within one VM to remain unaffected by others, enhancing security and stability. They are widely used for various purposes, including server consolidation, software testing, and creating development environments. VMs provide flexibility and efficiency, enabling easy resource management, snapshotting, and migration between physical hosts, making them a cornerstone technology in modern computing infrastructures.

HYPERVISORS

Hypervisors are software platforms that enable the creation and management of virtual machines (VMs) on physical hardware. They play a crucial role in virtualization by abstracting physical hardware resources and providing a virtualized environment for running multiple operating systems simultaneously. There are two main types of hypervisors: Type 1, also known as bare-metal hypervisors, run directly on the host's hardware without the need for a separate underlying operating system. They offer high performance and efficiency and are commonly used in enterprise environments. Type 2 hypervisors, on the other hand, run as an application within a conventional operating system. While they are easier to set up and manage, they introduce some performance overhead due to the additional layer of abstraction. Hypervisors are essential components of modern data centers, cloud computing environments, and virtualization solutions, enabling organizations to optimize resource utilization, improve scalability, and reduce costs.

Popular hypervisor platforms include VMware vSphere/ESXi, Microsoft Hyper-V, KVM, and Xen.

HYPERVISOR TYPES

- Type 1 Hypervisor: Runs directly on hardware without an underlying OS, offering high performance and resource efficiency, commonly used in enterprise environments.
  Examples include VMware vSphere/ESXi, Microsoft Hyper-V (when installed in standalone mode), KVM, and Xen.

- Type 2 Hypervisor: Operates as an application within an existing OS, providing ease of setup and management, suitable for development and testing purposes.
  Examples encompass VMware Workstation, Oracle VirtualBox, and Microsoft Virtual PC.

BENEFITS OF HYPERVISORS

Hypervisors offer several benefits:

- **Resource Utilization:** They enable efficient utilization of physical hardware by allowing multiple virtual machines to run on a single physical server, reducing hardware costs and optimizing resource allocation.

- **Isolation**: Hypervisors provide strong isolation between virtual machines, ensuring that activities in one VM do not impact others, enhancing security and stability.

- **Flexibility**: They allow for easy provisioning and deployment of virtual machines, enabling quick scalability to meet changing demands.

- **Disaster Recovery**: Virtual machine snapshots and replication facilitate easy backup and recovery processes, enhancing resilience and minimizing downtime in case of failures.

- **Testing and Development**: Hypervisors provide isolated environments for testing and development purposes, allowing developers to experiment with software configurations without affecting production systems.

- **Legacy Application Support**: They enable running legacy applications on modern hardware by providing compatibility with older operating systems and hardware configurations through virtualization.

- **Live Migration**: Hypervisors support live migration of virtual machines between physical hosts without disrupting services, enabling workload balancing, maintenance, and hardware upgrades with minimal downtime.

- **Consolidation**: They facilitate server consolidation by running multiple virtual machines on a single physical server, reducing the overall data center footprint and energy consumption.

Overall, hypervisors play a crucial role in modern computing environments, offering increased flexibility, efficiency, and resilience through virtualization technology.

Xen

Xen is a free and open-source type-1 hypervisor, providing services that allow multiple computer operating systems to execute on the same computer hardware concurrently. It was originally developed by the University of Cambridge Computer Laboratory and is now being developed by the Linux Foundation with support from Intel, Citrix, Arm Ltd, Huawei, AWS, Alibaba Cloud, AMD, Bitdefender and epam.

The Xen Project community develops and maintains Xen Project as free and open-source software, subject to the requirements of the GNU General Public License (GPL), version 2. Xen Project is currently available for the IA-32, x86-64 and ARM instruction sets.

Xen Project runs in a more privileged CPU state than any other software on the machine, except for firmware.

KVM

Kernel-based Virtual Machine (KVM) is a cornerstone of open-source virtualization technology, seamlessly integrated into the Linux kernel. By leveraging hardware-assisted virtualization extensions like Intel VT-x and AMD-V, KVM delivers exceptional performance for virtualized workloads, nearly matching that of native execution. This high-performance capability makes KVM a preferred choice for organizations seeking efficient virtualization solutions without the overhead of proprietary software. Its inclusion within the Linux kernel ensures stability, security, and ongoing support, as it benefits from the collective contributions and scrutiny of the open-source community.

One of the key advantages of KVM is its scalability, catering to diverse deployment scenarios ranging from small-scale setups to large-scale data centers. KVM supports the simultaneous operation of numerous virtual machines on a single physical host, efficiently utilizing hardware resources while maintaining strong isolation between virtual environments. Furthermore, KVM's support for live migration facilitates seamless workload management, allowing virtual machines to be dynamically moved between physical hosts without service interruption. This flexibility, combined with its robust security features and open-source nature, positions KVM as a versatile and reliable virtualization solution for

various use cases, including server consolidation, cloud computing, and development/testing environments.

Docker

Docker is a powerful tool for developers and system administrators alike. It essentially allows you to package your applications and their dependencies into a standardized unit called a container, which can then be easily deployed across different environments. Let's break down some key concepts:

1. Docker Overview:
    - Docker is an open platform that simplifies the process of developing, shipping, and running applications.
    - It provides a way to separate applications from infrastructure, facilitating quicker software delivery.
    - Docker allows you to manage both applications and infrastructure in a consistent manner.

- By leveraging Docker's methodologies, you can reduce the time it takes to move code from development to production.

2. Container:
   - A container is a lightweight, standalone, and executable software package that includes everything needed to run an application: code, runtime, system tools, system libraries, and settings.
   - Containers provide isolation and security, enabling multiple containers to run concurrently on the same host.
   - They eliminate the need to rely on the host system's configuration, ensuring consistent behavior across different environments.
   - Docker facilitates easy sharing of containers, ensuring that collaborators can work with identical environments.
3. Docker Image:
   - A Docker image is a template used to create Docker containers.
   - It contains all the necessary instructions to build a container, including the application code, runtime, dependencies, and configuration settings.
   - Docker images serve as the starting point for creating containers.
   - They can be compared to snapshots in virtual machine environments, capturing a specific state of the application and its dependencies.

In essence, Docker images and containers provide a standardized and efficient way to package, distribute, and run applications, simplifying the process of software development and deployment.

<u>Installing Docker</u>

1. Update Package Lists:

Before installing Docker, it's good practice to ensure that your package lists are up to date. You can do this by running:

sudo apt update

2. Install Dependencies:

Docker requires a few dependencies to be installed first. Run the following command to install them:

sudo apt install -y apt-transport-https ca-certificates curl software-properties-common

3. Add Docker's Official GPG Key:

Docker uses a GPG key for package verification. Add Docker’s official GPG key with:

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –

4.Add Docker Repository:

Add the Docker repository to your system's software sources:

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

5. Update Package Lists (Again):

After adding the repository, update the package lists again to ensure that Docker packages can be installed from the newly added repository:

sudo apt update

6. Install Docker Engine:

Finally, install Docker Engine:

sudo apt install docker-ce

7. Verify Docker Installation:

Once the installation is complete, you can verify that Docker is installed correctly by running the following command, which should output Docker version information:

docker --version

8. Manage Docker as a Non-root User (Optional):

By default, Docker commands require root privileges. If you want to run Docker commands without using sudo, you can add your user to the docker group:

sudo usermod -aG docker $USER

After running this command, log out and log back in for the changes to take effect.

9. Start and Enable Docker Service:

Docker should start automatically after installation, but you can ensure it's running and set to start on boot by running:

sudo systemctl start docker

sudo systemctl enable docker

That's it! Docker is now installed on your Linux system. You can start using Docker to build, ship, and run containers. Remember to refer to Docker's documentation for further information on how to use Docker commands and manage containers.

Result

Familiarised with Hypervisors, Virtual Machines, Xen, KVM and Docker

EXPERIMENT NO: 13 **DATE: 25-04-2024**

LINUX VM USING ANSIBLE PLAYBOOK

Aim:

Deploy VM using Ansible Playbook

Why Ansible?

Ansible is an open-source IT automation engine that simplifies tasks such as application deployment, intra-service orchestration, and cloud provisioning. It facilitates automation without requiring any agents or custom security infrastructure, making it easy to deploy. Here's a detailed overview of Ansible:

Key Features and Concepts

1. **Agentless Architecture**:
    - **Ease of Deployment**: Ansible does not require any agents to be installed on the nodes it manages. This reduces complexity and overhead.
    - **Connection Methods**: By default, Ansible connects to nodes via SSH but also supports other methods like Kerberos.
2. **Automation with Playbooks**:
    - **Playbooks**: Ansible uses playbooks to define automation tasks. Playbooks are written in YAML, a straightforward and human-readable data serialization language.
    - **Human-Readable Format**: YAML's simplicity means that IT staff, including those who may not be deeply familiar with programming, can read, write, and understand playbooks, facilitating easier debugging and management.
3. **Modular Design**:
    - **Ansible Modules**: Ansible pushes small programs called modules to the nodes. These modules execute tasks and are removed after completion.
    - **Task Execution**: This approach ensures that the nodes do not require persistent software beyond what is necessary to perform the task at hand.
4. **Multi-Tier Deployment**:

- **System Modeling**: Ansible is designed to handle multi-tier deployments by modeling IT infrastructure as interconnected systems rather than managing each system individually.
- **Interrelated Systems**: This enables complex orchestration where multiple systems need to work together.

5. **Inventory Management**:
    - **Hosts File**: Ansible manages inventory through simple text files known as hosts files. These files group hosts, allowing targeted actions on specific groups within playbooks.

Benefits

- **Simplified Automation**: Ansible's use of YAML for playbooks and its agentless architecture make automation accessible and straightforward.
- **Flexibility**: With the ability to connect using various methods and support for numerous modules, Ansible offers flexibility in managing diverse IT environments.
- **Scalability**: Ansible’s approach to modeling infrastructure enables it to scale from managing a few servers to thousands of nodes across complex environments.
- **Community and Support**: As an open-source tool, Ansible has a strong community and extensive documentation, which can be valuable resources for troubleshooting and extending its capabilities.

How Ansible Playbooks work?

Ansible interacts with your networks and sends little programs, known as modules, to them. These modules are utilized to complete automated tasks as systems designed to be resource models for the functioning at the desired state. Ansible runs these modules and eliminates them after they're done. If modules weren't available, you would have to depend on ad-hoc procedures and scripting to complete tasks. Ansible's management node is the primary node overseeing the Playbook's implementation.

The management node sets up an SSH connection before executing the modules and installing the product on the host workstations. Once the modules have been deployed, it eliminates them. So that's how it works with Ansible. Python is used to create an Ansible script and connects remote hosts through SSH, specified in the inventory file. Ansible is agentless – implying that it doesn't need any program to be installed on the nodes it controls.

Ansible takes inventory data to determine which machines you wish to control. It has a default inventory file, but users can customize it to manage the servers they want to. To link to servers and conduct tasks, Ansible employs the SSH protocol. Ansible establishes a connection with the remote system and distributes the modules required for command or playbook execution. Ansible uses human-readable YAML templates to allow the automation of repetitive processes without the need to master a complex programming language.

You could use Ansible's built-in modules to automate, or you can develop your own processes. Creating Ansible modules is possible using any programming language that can return JSON, including Ruby, Python, and shell. Powershell is used to develop Windows automation modules. It also stores your application's history information, enabling you to repeal to a previous standard or update it simply.

ANSIBLE INSTALLATION AND PLAYBOOK EXECUTION

Step 1: Install Ansible on your Control Machine

Assuming you're using a Debian-based system (like Ubuntu), you can install Ansible as follows:

sudo apt update

sudo apt install software-properties-common

sudo add-apt-repository --yes --update ppa:ansible/ansible

sudo apt install ansible

Verify the installation:

ansible --version

Step 2: Create an Ansible Playbook to Deploy a Linux VM

You'll need to create a playbook and possibly a few other configuration files depending on the virtualization environment you're using. Here, I'll demonstrate how to do this using KVM/QEMU.

1. **Install necessary packages on the control machine** (assuming it's also the host):

sudo apt update

sudo apt install qemu-kvm libvirt-daemon-system libvirt-clients bridge-utils virtinst

sudo systemctl enable --now libvirtd

sudo usermod -aG libvirt $USER

newgrp libvirt

2. **Create the playbook and necessary configuration files**:

- Inventory file (hosts):

[local]

localhost ansible_connection=local

- Playbook (deploy_vm.yml):

```
---
- name: Deploy a Linux VM
  hosts: local
```

```
  become: yes
  tasks:
    - name: Install necessary packages
      apt:
        name:
          - qemu-kvm
          - libvirt-daemon-system
          - libvirt-clients
          - virtinst
        state: present
        update_cache: yes

    - name: Create a disk image for the VM
      command: qemu-img create -f qcow2 /var/lib/libvirt/images/myvm.qcow2 20G
      args:
        creates: /var/lib/libvirt/images/myvm.qcow2

    - name: Download the ISO image
      get_url:
        url: https://releases.ubuntu.com/20.04/ubuntu-20.04.5-live-server-amd64.iso
        dest: /var/lib/libvirt/images/ubuntu-20.04.5-live-server-amd64.iso
        mode: 0755
        checksum:
sha256:443511f6bf12402c12503733059269a2e10dec602916c3c83e3e0c59762e1f1
d
```

```
- name: Define and start the VM
  virt:
    name: myvm
    state: running
    command: define
    xml: |
      <domain type='kvm'>
        <name>myvm</name>
        <memory unit='MiB'>2048</memory>
        <vcpu placement='static'>2</vcpu>
        <os>
          <type arch='x86_64' machine='pc-i440fx-2.9'>hvm</type>
          <boot dev='cdrom'/>
        </os>
        <devices>
          <disk type='file' device='disk'>
            <driver name='qemu' type='qcow2'/>
            <source file='/var/lib/libvirt/images/myvm.qcow2'/>
            <target dev='vda' bus='virtio'/>
          </disk>
          <disk type='file' device='cdrom'>
            <driver name='qemu' type='raw'/>
            <source file='/var/lib/libvirt/images/ubuntu-20.04.5-live-server-
amd64.iso'/>
            <target dev='hda' bus='ide'/>
            <readonly/>
```

```
    </disk>
    <interface type='network'>
      <source network='default'/>
      <model type='virtio'/>
    </interface>
    <console type='pty'>
      <target type='serial' port='0'/>
    </console>
  </devices>
</domain>
```

Step 3: Execute the Playbook

Run the playbook with the following command:

ansible-playbook -i hosts deploy_vm.yml

This streamlined playbook covers the essential steps:

1. **Install necessary virtualization packages**: Installs **qemu-kvm**, **libvirt-daemon-system**, **libvirt-clients**, and **virtinst**.
2. **Create a disk image for the VM**: Creates a 20GB disk image.
3. **Download the Ubuntu Server ISO image**: Downloads the ISO to be used for installation.
4. **Define and start the VM**: Defines the VM with 2GB RAM, 2 vCPUs, and boot from the downloaded ISO, then starts the VM.

Result:

Installed Ansible and executed a playbook