Vidyavardhini's

College of Engineering & Technology

Vasai Road (W)

Department of Electronics and Telecommunication Engineering

Laboratory Manual
(Faculty Copy)

Semester	VI	**Class**	TE
Course Code	ECL604		
Course Name	Skill Laboratory: Linux and Networking & Server Configuration		

Vidyavardhini's College of Engineering & Technology

Vision

To be a premier institution of technical education; always aiming at becoming a valuable resource for industry and society.

Mission

- To provide technologically inspiring environment for learning.
- To promote creativity, innovation and professional activities.
- To inculcate ethical and moral values.
- To cater personal, professional and societal needs through quality education.

Department Vision:

To contrive educational and research environments to serve industry and society needs in the field of electronics and telecommunication engineering.

Department Mission:

- To enrich soft skills, ethical values, environmental and societal awareness.
- To develop technical proficiency through projects and laboratory work.
- To encourage students for lifelong learning through interaction with the outside world.

Program Education Objectives (PEOs):

PEO1: The graduates will exhibit knowledge of mathematics, science, electronics, and communication, and will be able to apply the same in diversified field.

PEO2: The graduates will develop a habit of continuous learning while working in multidisciplinary environment.

PEO3: The graduates will grow as an individual with proficiency in technical skills, ethical values, communication skills, teamwork and professionalism.

Program Specific Outcomes (PSOs):

At the end of the program engineering graduate will be able to

PSO1: To apply the knowledge of Electronics and Communication to analyse, design and implement application specific problems with modern tools.

PSO2: Adapt emerging technologies with continuous learning in the field of electronics and telecommunication engineering with appropriate solutions to real life problems.

Program Outcomes (POs):

Engineering Graduates will be able to:

- **PO1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- **PO2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- **PO3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with

appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

- **PO4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- **PO5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- **PO6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- **PO7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- **PO8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- **PO9. Individual and teamwork:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- **PO10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- **PO11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- **PO12. Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Sr. No.	Content
1.	Syllabus
2.	Course Objectives and Course Outcomes
3.	Mapping of Experiments with Course Outcomes
4.	Mapping of COs with POs and PSOs
5.	List of Experiments

1. Linux installation process using any one following method: CD-ROM, Network installation or Kickstart installation.

2. Explore the internal and external commands of Linux.

3. Explore System Calls in Linux.

4. Shell scripting to show various system configuration.

5.

6.

7..

8.

9.

Course Objectives

1	Install Linux and implement standard Linux commands.
2	Study the basic theory of the Linux Operating System.
3	Implement the system administrative functionality.
4	To write shell script programs to solve problems.
5	Study basic commands of networking.
6	Develop implementation skills of different servers on Linux.

Course Outcomes

At the end of the course, students will be able to:		Action verb	Bloom Level
ECL604.1	Demonstrate Installation of Linux using Debian, Ubuntu, and Kali Linux platforms and execute standard Linux commands	Demonstrate	Apply (Level 3)
ECL604.2	Write Process management, scheduling, and Inter-process communication in Linux.	Write	Apply (Level 3)
ECL604.3	Write shell script programs for common administrative tasks and managing user accounts.	Write	Apply (Level 3)
ECL604.4	Write shell script programs for conditional and looping statements in bash.	write	Apply (Level 3)
ECL604.5	Create DHCP server, DNS server, and NFS file server.	Create	Evaluate
ECL604.6	Create and Deploy Mail server, Telnet server, FTP server, and Web server.	Create	Evaluate

Mapping of Experiments with Lab Outcomes

Sr. No	Experiment	Course Outcomes					
		ECL604.1	ECL604.2	ECL604.3	ECL604.4	ECL604.5	ECL604.6
1	Linux installation process using any one following method: CD-ROM, Network installation or Kickstart installation.	3	-	-	-	-	-
2	Explore the internal and external commands of Linux.	3	-	-	-	-	-
3	Explore System Calls in Linux.	-	3	-	-	-	-
4	Shell scripting to show various system configuration.	-	-	3	-	-	-
5	Write a shell script to add user and password on Linux system.	-	-	3	-	-	-
6	Write a shell script program to check login details.	-	-	3	-	-	-
7	Write a shell script to find the factorial of a given integer.	-	-	-	3	-	-
8	Write a script that accepts the hostname and IP address as command line arguments and adds them to the /etc/hosts file.	-	-	-	3	-	-
9	Write an awk script to find the number of characters, words, lines in a file.	-	-	-	3	-	-

10	To setup and configure FTP server with VSFTPD on Ubuntu 20.04.	-	-	-	-	-	3
11	To setup and configure Linux Mail Server.	-	-	-	-	-	3
12	Configuration of DNS server with domain name in Linux.	-	-	-	-	3	-

Enter correlation level 1, 2 or 3 as defined below
1: Slight (Low) 2: Moderate (Medium) 3: Substatial (High)

If there is no correlation put “—“.

INDEX

Sr. No.	Experiment	DOP	DOC	Remark	Sign
1	Linux installation process using any one following method: CD-ROM, Network installation or Kickstart installation.				
2	Explore the internal and external commands of Linux.				
3	Explore System Calls in Linux.				
4	Shell scripting to show various system configuration				
5	Write a shell script to add user and password on Linux system.				
6	Write a shell script program to check login details.				
7	Write a shell script to find the factorial of a given integer.				
8	Write a script that accepts the hostname and IP address as command line arguments and adds them to the /etc/hosts file.				
9	Write an awk script to find the number of characters, words, lines in a file.				
10	To setup and configure FTP server with VSFTPD on Ubuntu 20.04.				
11	To setup and configure Linux Mail Server.				
12	Configuration of DNS server with domain name in Linux.				

D.O.P: Date of performance D.O.C : Date of correction

Experiment No.: 1

Linux Installation

Aim:

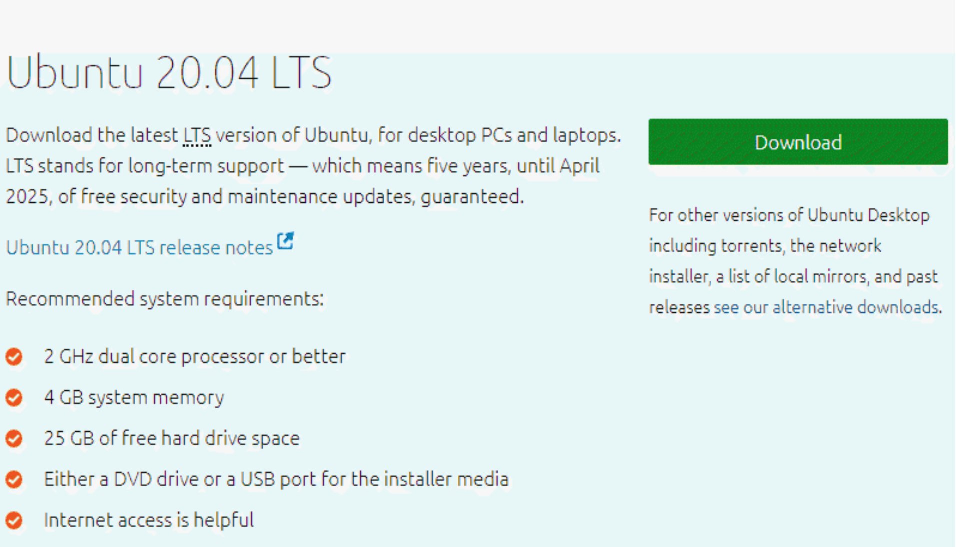
To install Linux using CD-ROM Network installation, or Kickstart installation

Theory:

Installation Steps of Ubuntu 20.04 LTS

Step:1 Download the ISO file using the following links

https://ubuntu.com/download/desktop



Ubuntu 20.04 LTS

Download the latest LTS version of Ubuntu, for desktop PCs and laptops. LTS stands for long-term support — which means five years, until April 2025, of free security and maintenance updates, guaranteed.

Ubuntu 20.04 LTS release notes

Recommended system requirements:

- 2 GHz dual core processor or better
- 4 GB system memory
- 25 GB of free hard drive space
- Either a DVD drive or a USB port for the installer media
- Internet access is helpful

Download

For other versions of Ubuntu Desktop including torrents, the network installer, a list of local mirrors, and past releases see our alternative downloads.

Once the iso file has been downloaded, burn it into DVD or USB drive and make it bootable. Save the file to a location of your choice

Step 2: Create Bootable USB

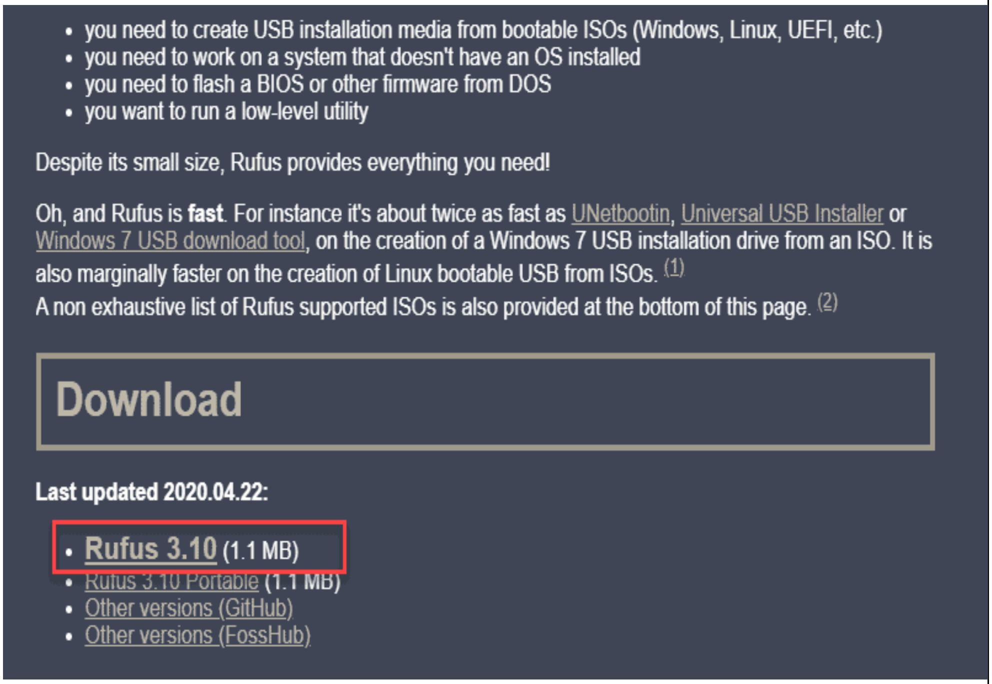
You will need a USB drive with 4GB or more. **This process will delete all data on the USB drive.** Make sure to backup any existing data on the USB drive.

Option 1: Create a Bootable USB Drive on Ubuntu

Use the **Create startup disk** tool:

1. Open a **search dialog**, and type *create startup*.
2. If it's not installed, the Software Center will offer the option to install it – choose the option for USB drive, then open the utility.
3. In the top pane, click **Other**, then browse and select the Ubuntu 20.04 .iso file you downloaded.
4. In the bottom pane, select your USB drive.

5. Click **Make startup disk.**



- you need to create USB installation media from bootable ISOs (Windows, Linux, UEFI, etc.)
- you need to work on a system that doesn't have an OS installed
- you need to flash a BIOS or other firmware from DOS
- you want to run a low-level utility

Despite its small size, Rufus provides everything you need!

Oh, and Rufus is **fast**. For instance it's about twice as fast as UNetbootin, Universal USB Installer or Windows 7 USB download tool, on the creation of a Windows 7 USB installation drive from an ISO. It is also marginally faster on the creation of Linux bootable USB from ISOs. (1)
A non exhaustive list of Rufus supported ISOs is also provided at the bottom of this page. (2)

Download

Last updated 2020.04.22:

- Rufus 3.10 (1.1 MB)
- Rufus 3.10 Portable (1.1 MB)
- Other versions (GitHub)
- Other versions (FossHub)

Bootable USB Drive on Windows

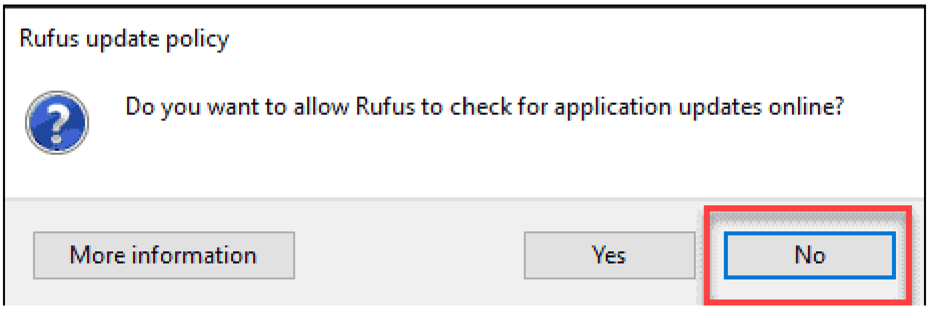
Option 2: Create

You'll need to install a third-party utility called **Rufus** to create a USB bootable drive.

1. Download the Rufus utility. Scroll down to the download section and click the link to download the latest version of Rufus.

2. Run the file once downloaded.

3. A pop-up dialog opens. You will be prompted whether you want to check for online updates. Select **No**.

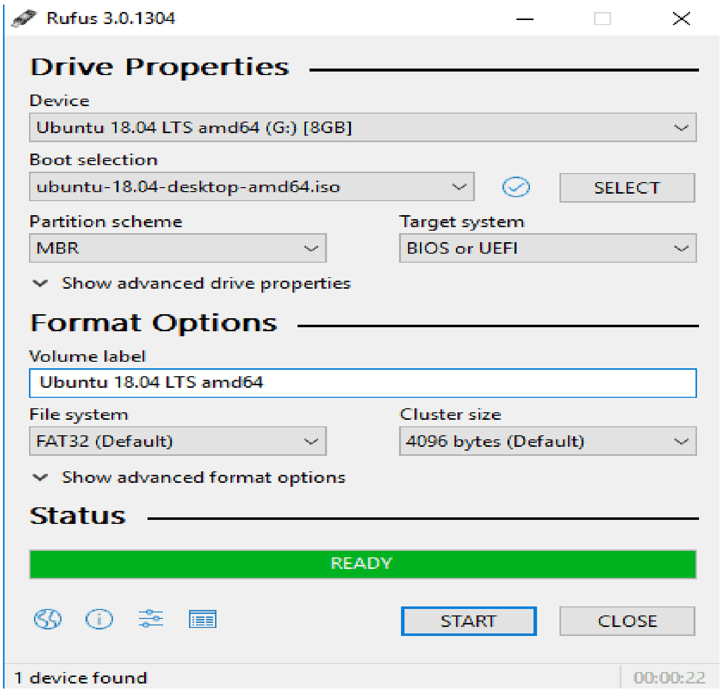


Rufus update policy

Do you want to allow Rufus to check for application updates online?

4. The Rufus utility launches. Plug in the USB drive – you should see the drive pop up in the device field.

- Set the USB as the device you wish to write to.
- In the *Boot Selection* drop-down, click **Disk or ISO Image.**
- Click the **Select** button to the right.
- Browse and select the .iso Ubuntu file you downloaded earlier.



Rufus 3.0.1304

Drive Properties

Device: Ubuntu 18.04 LTS amd64 (G:) [8GB]

Boot selection: ubuntu-18.04-desktop-amd64.iso

Partition scheme: MBR

Target system: BIOS or UEFI

Format Options

Volume label: Ubuntu 18.04 LTS amd64

File system: FAT32 (Default)

Cluster size: 4096 bytes (Default)

Status

READY

1 device found

00:00:22

6. Click **Start**.

Step 3: Boot up Ubuntu from USB

1. **Turn off your system**. Make sure you remove all other USB devices, such as printers, memory cards, etc.

2. **Insert the Ubuntu USB drive** into the system and turn on your machine. There are two possible scenarios:

- The computer boots the USB drive automatically.
- You need to manually configure USB booting in the **Boot Menu** or **BIOS/UEFI**.

3. To manually configure the boot order, tap the boot menu key about once or twice per second as soon as the computer powers on.

The boot menu key may be different depending on your computer manufacturer. Below is a list of common boot keys associated to a brand:

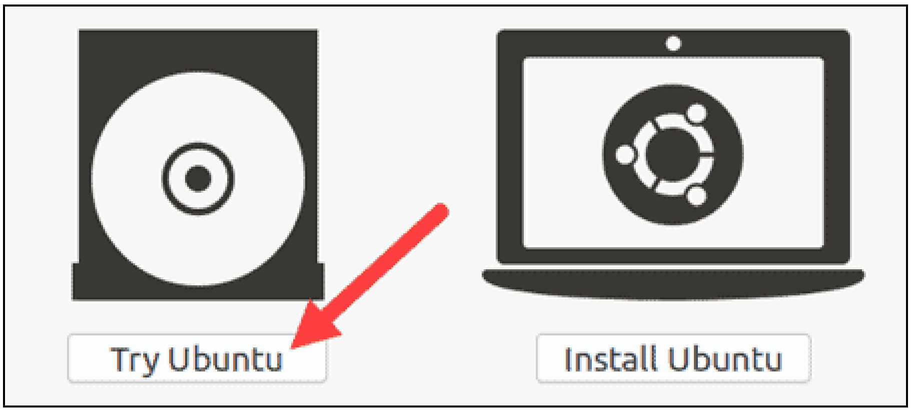
Asus	**F8** or **Esc**
Acer	**F12**, **F9** or **Esc**
Compaq	**F9** or **Esc**
Dell	**F12**
eMachines	**F12**
Fujitsu	**F12**
HP	**F9** or **Esc**
Lenovo	**F8**, **F10** or **F12**
Samsung	**F2**, **F12** or **Esc**
Toshiba	**F12**

4. Once you see your boot menu, use the arrows to pick the Ubuntu media to boot from. For a DVD, the entry will usually have DVD or Optical in the name. USB is usually labeled USB.Your system should start loading the Ubuntu live disc menu.

Step 4: Run Ubuntu

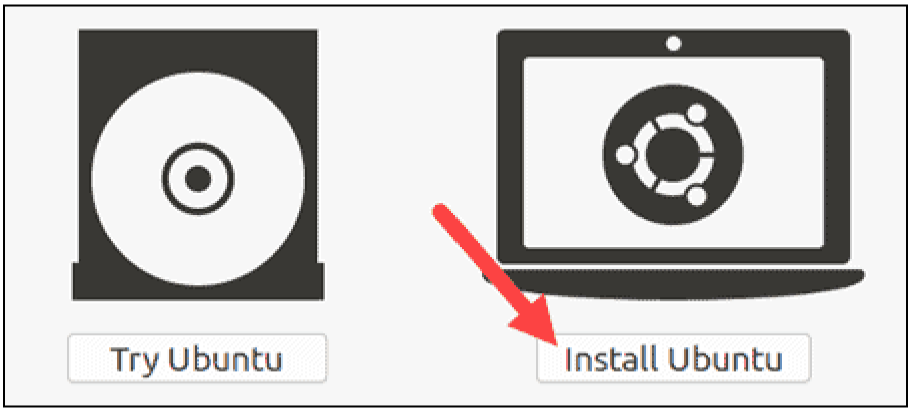
You can test Ubuntu 20.04 before you commit to installing it. The .iso includes a live mode that only runs in memory.

Launch this mode by clicking **Try Ubuntu**.



Step 5: Install Ubuntu 20.04 LTS Desktop

To begin the installation, click **Install Ubuntu**.

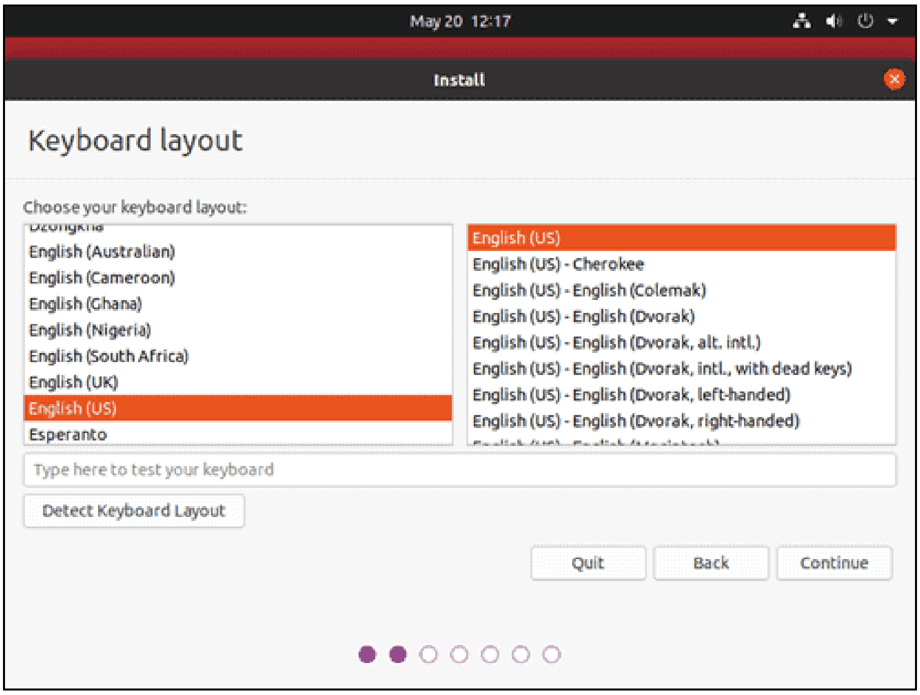


Choose Keyboard Layout

By default, the system will select English and English.

If you have a non-standard keyboard, you can select it in the list. Alternately, click **Detect Keyboard Layout** and the system will automatically choose your keyboard. If you need to test your keyboard, use the labeled field.

When you're ready, click **Continue**.



Keyboard layout

Choose your keyboard layout:

Dzongkha
English (Australian)
English (Cameroon)
English (Ghana)
English (Nigeria)
English (South Africa)
English (UK)
English (US)
Esperanto

English (US)
English (US) - Cherokee
English (US) - English (Colemak)
English (US) - English (Dvorak)
English (US) - English (Dvorak, alt. intl.)
English (US) - English (Dvorak, intl., with dead keys)
English (US) - English (Dvorak, left-handed)
English (US) - English (Dvorak, right-handed)

Type here to test your keyboard

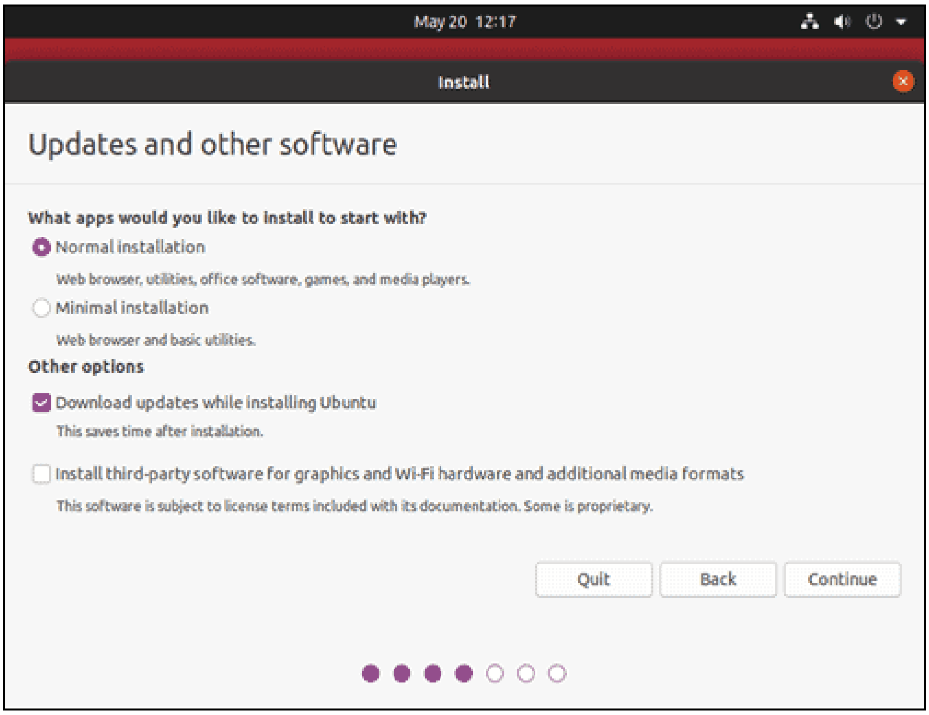
Choose Starting Applications

- **Normal Installation** – This is the full Ubuntu Desktop experience, with office software, games, and media players.

- **Minimal Installation –** Choose this to save disk space, especially if you won't be using media players or productivity software.

You'll also be asked to confirm other options:

- **Download updates while installing Ubuntu** – This does the work of downloading large package files during the installation. Once the installation finishes, the packages will be ready to apply as updates.
- **Install third-party software for graphics and Wi-Fi hardware and additional media formats –** Some hardware, like graphics cards and wi-fi cards, do not have open-source driver support. Also, some media formats, such as .wmv, do not fall under the GPL license. If you need support for these, you'll need to agree to additional terms of use.



May 20 12:17

Install

Updates and other software

What apps would you like to install to start with?

● Normal installation

Web browser, utilities, office software, games, and media players.

○ Minimal installation

Web browser and basic utilities.

Other options

☑ Download updates while installing Ubuntu

This saves time after installation.

□ Install third-party software for graphics and Wi-Fi hardware and additional media formats

This software is subject to license terms included with its documentation. Some is proprietary.

Disk Partitioning

Next, you'll be presented with an **Installation Type** dialog. You can wipe the hard drive clean prior to installing Ubuntu by clicking **Erase disk and install Ubuntu**. If you go this route, skip ahead to the next step.
Advanced users may want to edit **Advanced Features**. Use this to specify your own disk partitions or set other advanced options:

- **Use LVM with the new Ubuntu installation:** LVM stands for *Logical Volume Management*. This is a tool for dynamically managing different virtual drives on your system. It’s much like an enhanced version of the **gparted** tool.

- Encrypt the new Ubuntu installation for security: This will encrypt the drive’s contents. You’ll choose a security key, which will be required to decrypt and use the drive.
- Experimental: Erase disk and use ZFS: ZFS refers to Zettabyte File System, but it has grown into a hybrid file system and volume manager. Since it’s still being tested, avoid this setting on mission-critical production systems.

If you’d rather create your own hard drive partitions, click **Something Else**. The next screen will allow you to create your own partition table and logical drives. This lets you divide a physical hard drive into different partitions. The operating system sees partitions as individual drives.

Click **Continue** to apply your changes to the drive partitions.

You’ll be asked to **Write changes to disks?** None of the options you’ve selected are permanent until you click **Continue** on this screen. Click **Continue** to proceed.

Select Time Zone

Once the system formats the disk partitions, the installer will ask **Where are you?**

Type the nearest large city into the box, and the system will **set your local time zone**.

Click **Continue**.

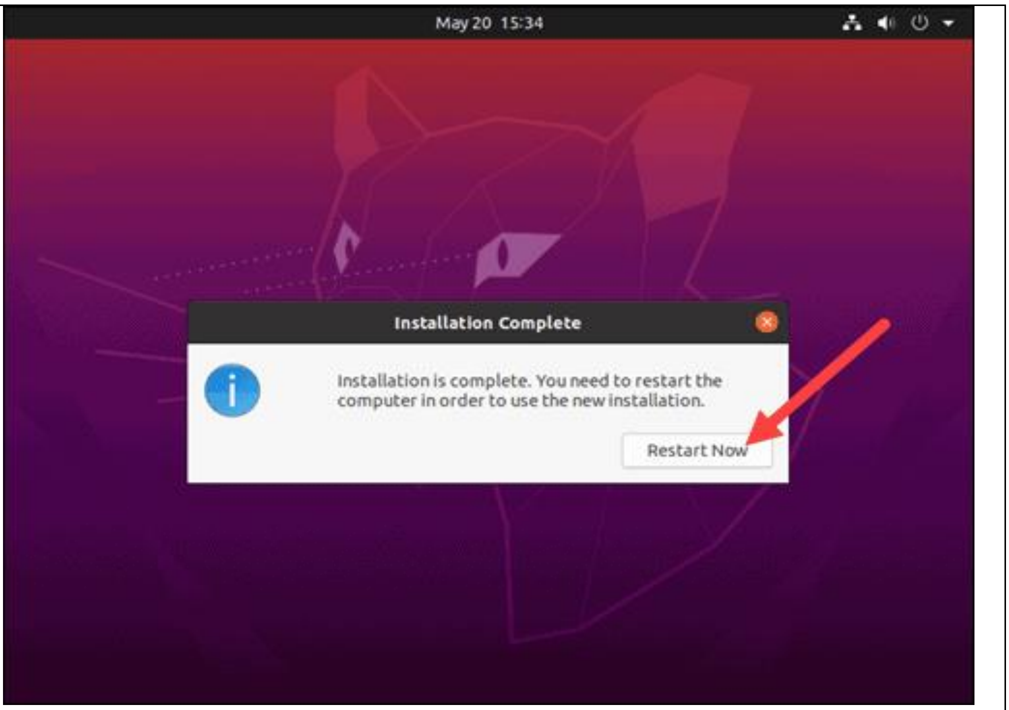
Create User Account

Next, you’ll need to configure a user account. Fill in the following fields:

- **Name:** Your actual name.
- **Computer name:** This is the hostname or network name.
- **Username:** The user account name you want to use.
- **Password:** Enter and confirm a strong password – the installer will automatically evaluate your password strength.
- **Log in automatically:** This is not recommended for publicly accessible servers.
- **Require my password to log in:** This is recommended for publicly accessible servers.

Click **Continue** to install Ubuntu.

Once the installer finishes, remove the Ubuntu installation media. You’ll be prompted to **Restart Now**.

	 Installation Complete Installation is complete. You need to restart the computer in order to use the new installation. The system should boot into your fresh install of Ubuntu 20.04.
Conclusion:	
Post Experiment questions:	1) Comment on the installation of Linux OS based on various parameters. 2) What are two types of Linux User Mode?

.

Experiment No: 2

Linux Commands

Aim:	Explore the internal and external commands of Linux.
Objective:	The Linux command is a utility of the Linux operating system. All basic and advanced tasks can be done by executing commands.
Theory:	**They are grouped into two categories:** ● **Internal Commands:** Commands which are built into the shell. For all the shell built-in commands, execution of the same is fast in the sense that the shell doesn't have to search the given path for them in the PATH variable, and also no process needs to be spawned to execute it. Examples: source, cd, fg, etc. ● **External Commands:** Commands which aren't built into the shell. When an external command has to be executed, the shell looks for its path given in the PATH variable, and also a new process has to be spawned and the command gets executed. They are usually located in /bin or /usr /bin. For example, when you execute the "cat" command, which usually is at /usr/bin, the executable /usr/bin/cat gets executed. Examples: ls, cat, etc.
pwd	It gives an absolute path to your current location i.e. current working directory. napster@napster-Veriton-Series:~$ pwd /home/napster
mkdir	It creates a new directory/ folder napster@napster-Veriton-Series:~$ mkdir TRIAL
cd	It is used to change directory. napster@napster-Veriton-Series:~$ cd TRIAL
cd..	To come back to previous directory.
**cd **	Return to the root directory
touch	To create new files. napster@napster-Veriton-Series:~/TRIAL$ touch hello.txt To verify hello.txt is created or not: napster@napster-Veriton-Series:~/TRIAL$ ls Hello.txt
Is	It lists the contents of files and directories

	napster@napster-Veriton-Series:~/TRIAL$ ls hello.txt TRIAL2
cat	1) It can also use to create new file with content as shown below. cat> hello.txt hello from Shamika 2) It can also use to append the data into existing file as shown below. cat>>hello.txt 3) It is used to concatenate files. napster@napster-Veriton-Series:~/TRIAL$ cat hello.txt hi.txt hello from Shamika How are you? Welcome to Os Lab.
b1@comp:~$ cat > a	Apple mango is also a fruit ^C
b1@comp:~$ cat >> a	APPENDING ^C
mv	To rename a file from source to destination and To move file from one location to other location. napster@napster-Veriton-Series:~/TRIAL$ mv hi.txt how.txt
To verify the file is renamed or not.	napster@napster-Veriton-Series:~/TRIAL$ ls hello.txt how.txt
To move files from one location to another location	napster@napster-Veriton-Series:~/TRIAL$ mkdir TRIAL2 napster@napster-Veriton-Series:~/TRIAL$ mv how.txt TRIAL2 napster@napster-Veriton-Series:~/TRIAL$ cd TRIAL2 napster@napster-Veriton-Series:~/TRIAL/TRIAL2$ LS napster@napster-Veriton-Series:~/TRIAL/TRIAL2$ ls How.txt
grep	It searches all text files in the current directory for lines containing "hello" napster@napster-Veriton-Series:~/TRIAL$ grep hello *.txt hello from Shamika
rm	remove/delete files. napster@napster-Veriton-Series:~/TRIAL/TRIAL2$ rm how.txt
date	Print or set the system date and time, Display the current time in the given FORMAT, or set the system date.

b1@comp:~$ date	Fri Feb 16 15:43:44 IST 2018
time:	Displays time of the system
free:	Shows the amount of RAM in use.
echo:	Echoes output on the screen.
clear:	Clears the screen.
exit:	Exit from the terminal
man :	(man command name) Gives description
gedit:	To open text editor
ps	Report a snapshot of the current processes. ps displays information about a selection of the active processes.
b1@comp:~$ ps	PID TTY TIME CMD 2227 pts/0 00:00:00 bash 2310 pts/0 00:00:00 ps
cal :	Displays a calendar
b1@comp:~$ cal	February 2018 Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
wc:	print newline, word, and byte counts for each file, Print newline, word, and byte counts for each FILE, and a total line if more than one FILE is specified.
b1@comp:~$ wc a	6 11 57 a
chmod	change file mode bits chmod changes the file mode bits of each given file according to mode, which can be either a symbolic representation of changes to make or an octal number representing the bit pattern for the new mode bits.
b1@comp:~$ chmod 777 a **b1@comp:~$ ls -l a**	-rw-rw-r-- 1 b1 b1 57 Feb 16 15:44 a
chown	change file owner and group chown changes the user and/or group ownership of each given file. If only an owner (a user name or numeric user ID) is given, that user is made the owner of each given file, and the file's group is not changed. If the owner is followed by a colon and a group

	name (or numeric group ID), with no spaces between them, the group ownership of the files is changed as well.
umask	set file mode creation mask , umask() sets the calling process's file mode creation mask (umask) to mask & 0777 (i.e., only the file permission bits of mask are used), and returns the previous value of the mask.
b1@comp:~$ umask -S	u=rwx,g=rwx,o=rx
Conclusion:	
Post Experiment questions:	**1. Comment on the basic difference between internal and external commands.** **2. What are the scheduling techniques in Linux?**

Experiment No. 3
Explore System Calls in Linux.
Date of Performance:
Date of Submission:

Experiment No. 3

Explore System Calls in Linux

Aim:	Explore the system calls open, read, write, close, getuid, getgid, getegid, geteuid o Linux.
Objective:	When a program in user mode requires access to RAM or a hardware resource, it must ask the kernel to provide access to that resource. This is done via a system call.
Theory:	• **getuid, geteuid** - get user identity • **getgid, getegid** - get group identity ● **getuid**() returns the real user ID of the calling process. ● **geteuid**() returns the e ffective user ID of the calling process. ● **getgid()** returns the real group ID of the calling process. ● **getegid()** returns the effective group ID of the calling process. All four functions shall always be successful and no return value is reserved to indicate an error. Unix-like operating systems identify a user within the kernel by a value called a **user identifier**, often abbreviated to **user ID** or **UID**. The UID, along with the group identifier (GID) and other access control criteria, is used to determine which system resources a user can access.
Effective user ID	The effective UID (euid) of a process is used for most access checks. It is also used as the owner of files created by that process. The effective GID (egid) of a process also affects access control and may also affect file creation, depending on the semantics of the specific kernel implementation in use and possibly the mount options used.
Open:	Used to Open the file for reading, writing, or both. Open() returns file descriptor **3** because when main process is created, then fd **0, 1, 2** are already taken by **stdin**, **stdout,** and **stderr**. So first unused file descriptor is **3** in the file descriptor table. int open(const char *pathname, int flags);
Parameters	➢ **Path :** file path which you want to use. • Use an absolute path begin with "/", when you are not work in the same directory of the file.

	• Use relative path which is only file name with extension, when you are working in the same directory of file. • **flags:** How you like to use • **O_RDONLY**: read only, • **O_WRONLY**: write only, • **O_RDWR**: read and write, • **O_CREAT**: create file if it doesn't exist
Close:	Tells the operating system you are done with a file descriptor and Close the file which pointed by fd. int close (int fd); **Parameter** • **fd:**file descriptor **Return** • on success. • **-1** on error.
read:	Read data from one buffer to file descriptor, Read size bytes from the file specified by fd into the memory location. size_t read (int fd, void* buf, size_t cnt); ➢ **Parameters** • fd: file descriptor • buf: buffer to read data from • cnt: length of buffer ➢ **Returns: How many bytes were read** • return the Number of bytes read on success • return 0 on reaching the end of the file • return -1 on error • return -1 on signal interrupt
Write:	Write data from a file descriptor into the buffer, Writes the bytes stored in **buf** to the file specified by **fd**. The file needs to be opened for write operations size_t write (int fd, void* buf, size_t cnt); ➢ **Parameters** • fd: file descriptor • buf: buffer to write data to

	• cnt: length of buffer ➢ **Returns: How many bytes were written** • return the Number of bytes written on success • return 0 on reaching the end of the file • return -1 on error • return -1 on signal interrupt
Conclusion:	
Post Experiment questions:	1. Comment on the kernel mode of the operating system. 2. Explain the role of the system administrator

Code:

```
#include<stdio.h>
#include <unistd.h>
#include<string.h>
#include<fcntl.h>
int main()
{
        char data[64];
        printf("getuid : %d \n",getuid());
        printf("geteuid : %d \n",geteuid());
        printf("getgid : %d \n",getgid());
        printf("getegid : %d \n",getegid());
        int fd = open("/home/shraddha/SYCALL.txt", O_CREAT | O_RDWR |
O_APPEND);
        printf("fd : %d\n", fd);
        int read1 = read(fd,data,64);
        printf("%d\n",read1);
        int w = write(fd,"smita",strlen("smita"));
        close(fd);
}
```

Code for Open() System Call

```
d9@d9-desktop:~$ touch text.txt
d9@d9-desktop:~$ cat text.txt
d9@d9-desktop:~$ cat>text.txt
Hi this is my new document
How are you?
cat > text.txt
d9@d9-desktop:~$ nano new.c

#include<fcntl.h>
#include<sys/types.h>
#include<unistd.h>
#include<sys/stat.h>
int main()
{
      int n, fd, fd1;
      char buf[30];
      fd=open("text.txt", O_RDONLY);
      n=read(fd, buf, 20);
```

```
        fd1=open("target", O_CREAT|O_WRONLY,0642);
        write(fd1, buf, n);
}

d9@d9-desktop:~$ gcc new.c
d9@d9-desktop:~$ ./a.out
d9@d9-desktop:~$ cat target
Hi this is my new dod9@d9-desktop:~$ nano new.c
d9@d9-desktop:~$ ^C
d9@d9-desktop:~$
d9@d9-desktop:~$ nano exp.c

#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main(int argc, char *argv[])
{
        printf("PID of exp.c= %d\n", getpid());
        char *args[]={"Hello","C", "Programming", NULL};
        execv("./hello",args);
        printf("Back to exp.c");
        return 0;
}

d9@d9-desktop:~$ nano hello.c

#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
        printf("We are in hello.c\n");
        printf("PID of hello.c= %d\n", getpid());
        return 0;
}
d9@d9-desktop:~$ gcc -o exp exp.c
d9@d9-desktop:~$ gcc -o hello hello.c
d9@d9-desktop:~$ ./exp
```

OUTPUT:

```
shraddha@shraddha-Inspiron:~$ gcc uid.c -o s
shraddha@shraddha-Inspiron:~$ ./u
getuid : 1000
geteuid : 1000
getgid : 1000
getegid : 1000
fd : 3
0
```

```
shraddha@shraddha-Inspiron:~$ ./s
getuid : 0
geteuid : 0
getgid : 0
getegid : 0
fd : 3
10
```

```
PID of exp.c= 5445
We are in hello.c
PID of hello.c= 5445
```

Experiment No. 4

Explore System Configurations

Aim: Write Shell Scripts to do the following:
1. Display top 10 processes in descending order.
2. Display processes with highest memory usage.
3. Display current logged in user and log name.
4. Display current shell, home directory, OS type, OS version, release number, kernel version, current path setting, current working directory.

Objective: The shell is the operating system's command-line interface (CLI) and interpreter for the set of commands that are used to communicate with the system. A shell script is usually created for command sequences in which a user has a need to use repeatedly in order to save time.

Theory:

Shell is a user program or its environment is provided for user interaction. It is a command prompt within Linux where you can type commands. It is a program that takes your commands from the keyboard and gives them to the OS to perform. Shell is not part of system KERNAL but it uses system KERNAL to execute programs, create files, etc. A Shell Script is a text file that contains a sequence of commands for a UNIX-based OS. It is called a Shell Script because it combines into a "Script" in a single file a sequence of commands, that would otherwise have to be presented to the system from a keyboard one at a time. A Shell Script is usually created for command sequences for which a user has a repeated need. You initiate the sequence of commands in Shell Script by simply entering the name of the Shell Script on a command line.

Types of Shell Script

sh	Simple Shell
bash	Bourne Again Shell
ksh	Korne Shell
csh	C Shell
ssh	Secure Shell To use a particular Shell type the Shell name at the command prompt. Eg:- $csh - It will switch the current Shell to C Shell. To view the current Shell that is being used, type echo $ SHELL at the command prompt.
Conclusion:	

Post Experiment questions:	1) Comment on the advantages of using shell scripting in Linux. 2) How to check system configuration in Linux command?

CODE:

```
#!/bin/bash
#Display current logged in user and logname.
echo  "Display current logged in user and logname."
echo "Hi,$USER! This is username."
echo "Hello, $LOGNAME! This is logname";          echo ""
```

```
#Display top 10 processes in descending order.
echo "Display top 10 processes in descending order";          echo ""
CPU1=$(top -b -n1 |tail -10)
echo "$CPU1";          echo ""
```

```
#Display processes with highest memory usage.
echo "Display processes with highest memory usage";        echo ""
CPU=$(top -b -n1 |head -10)
echo "$CPU"; echo ""
```

```
#Display Virtulmemory statistics.
echo "Display VirtulMemory statistics";                echo ""
vmstat= vmstat
echo "$vmstat";                  echo ""
```

```
#Display current shell, home directory, operating system type, current path setting
echo "Display current shell, home directory, operating system type, current path setting"
ALL=$(uname -a);     echo ""
echo "alternative for -a is:      --all:                        #prints all information";
echo "$ALL"
```

```
S=$(uname -s);          echo ""
echo "alternative for -s is:      --kernel-name:                      #prints the kernel name"
echo "$S"
```

```
R=$(uname -r);          echo ""
echo "alternative for -r is:      --kernel-release:         #prints the kernel release"
echo "$R"

V=$(uname -v);          echo ""
echo "alternative for  -v is:     --kernel-version:         #prints the kernel version"
echo "$V"
```

```
#Display current working directory.
echo "Present working directory "
echo "$PWD";echo ""
```

```
#Display current shell
echo "Display current program shell"
echo $0 $SHELL "$$"
echo "$0:        Name of currently running .sh file"
echo "$SHELL:          Folder"
echo "$$:                      PID"
```

OUTPUT:

```
co-036@co036-desktop:~$ bash shellbasic.sh
Display current logged in user and logname.
Hi,co-036! This is username.
Hello, co-036! This is logname
```

Display top 10 processes in descending order

```
 3478 root      20   0       0      0      0 S  0.0   0:00.00 kworker/u8:2
 4556 co-036    20   0  169556  29772  17000 S  1.5   0:04.60 gedit
 4947 co-036    20   0   26172   2916   1924 S  0.1   0:00.03 oosplash
 4965 co-036    20   0  284876  76880  50976 S  3.7   0:02.25 soffice.bin
 4986 co-036    20   0  136692  18356  12204 S  0.9   0:00.22 gnome-terminal
 4995 co-036    20   0    2420    704    584 S  0.0   0:00.00 gnome-pty-helpe
 4996 co-036    20   0    6924   3264   1656 S  0.2   0:00.04 bash
 5044 co-036    20   0    5300   1324   1152 S  0.1   0:00.00 bash
 5045 co-036    20   0    5300    544    368 S  0.0   0:00.00 bash
 5047 co-036    20   0    4268    600    536 S  0.0   0:00.00 tail
```

Display processes with highest memory usage

```
top - 10:16:02 up  1:03,  2 users,  load average: 0.31, 0.33, 0.27
Tasks: 162 total,   1 running, 161 sleeping,   0 stopped,   0 zombie
%Cpu(s):  5.3 us,  1.2 sy,  0.1 ni, 91.5 id,  1.9 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2052252 total, 1665232 used,   387020 free,    86024 buffers
KiB Swap:  2083836 total,        0 used,  2083836 free.   918836 cached Mem
```

```
PID USER      PR  NI    VIRT    RES    SHR S %MEM     TIME+ COMMAND
 5049 co-036    20   0    5424   1280    960 R  0.1   0:00.01 top
    1 root      20   0    4448   2516   1444 S  0.1   0:01.40 init
    2 root      20   0       0      0      0 S  0.0   0:00.00 kthreadd
```

Display VirtulMemory statistics

```
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0      0 387088  86024 918836    0    0    81    41  337  643  5  1 92  2  0
Display current shell, home directory, operating system type, current path setting
alternative for -a is:     --all:                        #prints all information
Linux co036-desktop 3.13.0-24-generic #46-Ubuntu SMP Thu Apr 10 19:08:14 UTC 2014
i686 i686 i686 GNU/Linux
```

```
alternative for -s is:     --kernel-name:                 #prints the kernel name
Linux

alternative for -r is:     --kernel-release:        #prints the kernel release
```

3.13.0-24-generic
alternative for -v is: --kernel-version: #prints the kernel version #46-Ubuntu SMP Thu Apr 10 19:08:14 UTC 2014 Present working directory /home/co-036
Display current program shell shellbasic.sh /bin/bash 5044 shellbasic.sh: Name of currently running .sh file /bin/bash: Folder 5044: PID

Experiment No. 5

Shell Script to add User

Aim:	Write a shell script program to add a user with password.
Objective:	Since Linux is a multi-user operating system, several people may be logged in and actively working on a given machine at the same time. Security-wise, it is never a good idea to allow users to share the credentials of the same account. In fact, best practices dictate the use of as many user accounts as people needing access to the machine. At the same time, it is to be expected that two or more users may need to share access to certain system resources, such as directories and files. User and group management in Linux allows us to accomplish both objectives.
Theory:	
useradd	create a new user or update default new user information, useradd is a low level utility for adding users.
userdel	delete a user account and related files
groupadd	create a new group, The groupadd command creates a new group account using the values specified on the command line plus the default values from the system. The new group will be entered into the system files as needed.
groupdel	entries that refer to GROUP. The named group must exist
who	show who is logged on, Print information about users who are currently logged in.
whoami	print effective userid
passwd	change user password The passwd command changes passwords for user accounts. A normal user may only change the password for his/her own account, while the superuser may change the password for any account. passwd also changes the account or associated password validity period.
	Linux shell script to add a user with a password
useradd -m -p	Encrypted Password Here **username** Where, -m : The user's home directory will be created if it does not exist. -p Encrypted Password Here : The encrypted password, as returned by crypt(). username: Add this user to the Linux system, **Step 1 – Create an encrypted password** You need to create an encrypted password using Perl crypt() as follows: crypt($plain, $salt)

	perl -e 'print crypt("Your-Clear-Text-Password-Here", "salt"),"\n"'
	crypt() is a one-way hash function. The PLAINTEXT ($plain) and SALT are turned into a short string, called a digest, which is returned. The same PLAINTEXT and SALT will always return the same string, but there is no (known) way to get the original PLAINTEXT from the hash. Small changes in the PLAINTEXT or SALT will result in large changes in the digest.
	Let us try out perl example: perl -e 'print crypt("2IL@ove19Pizza4_", "salt"),"\n"' **Sample output:** sa.KT9zrGYeg2
	The Perl command will display the encrypted password (sa.KT9zrGYeg2) on screen. The Perl crypt() function is a *one way encryption* method meaning, once a password has been encrypted, it cannot be decrypted. The password string is taken from the user and encrypted with the salt and displayed back on computer screen. We can store an encrypted password using the following syntax: password="1YelloDog@" pass=$(perl -e 'print crypt($ARGV[0], "password")' $password) echo "$pass" **Sample outputs** sa.KT9zrGYeg2
Conclusion:	
Post Experiment questions:	1. Comment on the role of user and group management commands of Linux. 2. Where the user details are stored in Linux?

COMMANDS:

```
b1@comp:~$ sudo useradd shr
[sudo] password for b1:
```

```
b1@comp:~$ sudo passwd shr
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

```
b1@comp:~$ sudo groupadd shr1
```

```
b1@comp:~$ cat /etc/group | tail -2
shr:x:1001:
shr1:x:1002:
```

```
b1@comp:~$ compgen -u | tail -2
guest-dR7woG
shr
```

```
b1@comp:~$ compgen -g | tail -2
shr
shr1
```

```
b1@comp:~$ sudo chgrp shr1 a
b1@comp:~$ ls -l a
-rwxrwxrwx 1 b1 shr1 72 Feb 16 16:09 a
```

```
b1@comp:~$ sudo userdel shr
```

```
b1@comp:~$ sudo groupdel shr1
```

```
b1@comp:~$ compgen -u | tail -2
guest-KdgnnY
guest-dR7woG
```

```
b1@comp:~$ compgen -g | tail -2
guest-KdgnnY
guest-dR7woG
```

```
b1@comp:~$ who
b1       :0          2018-02-16 15:37 (:0)
b1       pts/0        2018-02-16 16:16 (:0)
```

CODE:
Shell script to add a user and password on Linux

```
#!/bin/bash
if [ $(id -u) -eq 0 ]; then
        read -p "Enter username : " username
        read -s -p "Enter password : " password
        egrep "^$username" /etc/passwd >/dev/null
        if [ $? -eq 0 ]; then
echo "$username exists!"
                exit 1
        else
                pass=$(perl -e 'print crypt($ARGV[0], "password")' $password)
                useradd -m -p $pass $username
                [ $? -eq 0 ] && echo "User has been added to system!" || echo "Failed to
        add a user!"
        fi
else
        echo "Only root may add a user to the system"
        exit 2
fi
```

Grant executable permissions

Executable permissions must be granted to the files to make them run or execute on the system. We could also use “777” instead of “+x” in the chmod command. Also please run the script as root to

chmod +x file_name.sh

Run the script

For root owners

./file_name.sh

Example 1:

./file_name.sh

For non-root owners

sudo ./file_name.sh

Example:

sudo ./file_name.sh

Delete a user

Syntax: deluser username

Example:

deluser Smita

Experiment No. 6

Shell Script to check login details

Aim:	Shell script to print login details of a user.
Objective:	Linux is a multi-user operating system, several people may be logged in and actively working on a given machine at the same time. It is to be expected that two or more users may need to share access to certain system resources, such as directories and files. We can identify the number of users logged in, past users on the Linux system.
Theory:	Every operating system provides a feature of multiple user accounts. Linux-based operating systems have some commands or functionalities to 1 them. This ability is mainly used by the admin account user that is the root user, to provide permissions and access to different users. The admin can also check how many users are currently logged in, how many are logged out, and the login time. Here in this article, we will explore all these ways and also write a shell script to complete these tasks efficiently.
	Command to get user-related information
Id	The id command is used to print the user and group information for the specified USER. -a ignore, for compatibility with other versions -Z, --context print only the security context of the process -g, --group print only the effective group ID This id command has produced all the user identifiers, group identifiers, and groups. If you want only a group identifier, use the below command.
groups	This will print the group to which the specified user belongs. If no specific username is given, it will search for the current users. Use the below command for the current user.
getent	This command displays entries from the databases.
Syntax:	getent database [key ...] -i, --no-idn disable IDN encoding
Example:	getent -i ahosts --no-idn The following example lists the entire contents of the protocols database.

getent protocols getent passwd getent -V	Fetch the list of user accounts on a Linux system (stored in a database known as 'passwd'). This will show all the user accounts, regardless of the type of name service being used. The databases it usually searches in are: ahosts, ahostsv4, ahostsv6, aliases, ethers (Ethernet addresses), group, gshadow, hosts, netgroup, networks, passwd, protocols, rpc, services, and shadow.
lslogins:	To see all the usernames and user ids. This provides a list of several features like UID, USER, LAST-LOGIN, etc. -a, --acc-expiration display info about passwords expiration -c, --colon-separate display data in a format similar to /etc/pas
Users:	This command will print the usernames of the logged-in to the current host. This is the only user logged in currently.
who :	To show who is logged-on. This lists the users with id and the time and date of user login. -a, --all same as –b, -d, --login, -p, -r, -t, -T, -u
w: **-h,** **-u,**	w command shows the logged-on user accounts and also shows what they are doing. --no-header do not print header --no-current ignore current process username
last or lastb:	The commands last and lastb shows a listing of last logged in users -<number> how many lines to show
-a, --hostlast	display hostnames in the last column
-d, --dns	translate the IP number back into a hostname
lastlog:	This provides all the login details of several users according to date and time.last. This tells about the latest log of the users.
-b, --before DAYS **-C, --clear**	This will produce a report of all the recent login users. This can also create a single-user report if specified. print only lastlog records older than DAYS clear lastlog record of a user (usable on

Shell Script	Now we will create a shell script using some above-mentioned commands to get user details. We are approaching the solution in a way that the user is asked for input by given suggestions. That input will be then used to check against the available cases, and then the matched case will be allowed to run.
Open gedit file:	Open any editor according to your preferences, we have used gedit editor because of its simple user interface and the color combination present. gedit userAccounts.sh
Code:	Here in the userAccounts.sh we will write our code, and use switch cases to compare the user input. We have used commands like lslogins, who, groups, etc. which will help us to satisfy the user requirements. You could find the use of these commands more extended above. So, let us begin the script.
Conclusion:	
Post Experiment questions:	1. How do you change user information in Linux? 2. What is the command to check user account status in Linux?

CODE:

```
#!/bin/bash
#here we are going to develop a script for various options on user accounts
echo -e "\n
[ 1 ] for listing all the user accounts name \n
[ 2 ] for counting the number of logged-in user accounts  \n
[ 3 ] for listing the names of currently logged-in users\n
[ 4 ] for checking the groups to which the current user belong \n"
#Now take user input
read userInput
#Now we will use switch cases for various input operations
case $userInput in
   1)
   #syntax lslogins <option[=output field]>
   lslogins -o USER
   ;;
   2)
   #syntax who <option> <user optional>
   #grep used to filter
   who --count | grep users
   ;;
   3)
   #-q option is to count the number of users and print the logged-in users.
   # instead of -q, --count can also be used.
   # -v is used to exclude any pattern
   who -q | grep -v users
   ;;
   4)
```

```
    #syntax groups <option> [USERNAME]
    groups
    ;;
    *)
    echo -e "Please Enter Correct Input \n"
    ;;
esac
```

Grant executable permissions
Executable permissions must be granted to the files to make them run or execute on the system. We could also use “777” instead of “+x” in the chmod command. Also please run the script as root to

```
chmod +x userAccounts.sh
```

Run the script

```
sudo ./userAccounts.sh
```

Example 1:

```
sodu ./userAccounts.sh
```

Experiment No. 7

Shell Script to calculate factorial of a given number

Aim:	Shell script to calculate factorial of a given number.
Objective:	Linux bash shell is an interpreter. Shell program provides access to an operating system's components. The shell gives users (or other programs) a way to get "inside" the system; the shell defines the boundary between inside and outside. Here we can calculate a factorial of a given number using three methods. 1. Using recursive function 2. Using for loop 3. Using a do-while loop
Theory:	The factorial of a number is the function that multiplies the number by every natural number below it. Symbolically, factorials can be represented as "!". So, n factorial is the product of the first n natural numbers and is represented as n! Here we are going to calculate the factorial of a number. The Factorial of a non-negative integer is the multiplication of all integers smaller than or equal to n. For example factorial of 5 is 5*4*3*2*1 which is 120.
Conclusion:	
Post Experiment questions:	1. Comment on shell script program for calculating a factorial of a given number. 2. What is bash used for in Linux?

CODE:

Method 1: Using Recursion

Factorial can be calculated using the following recursive formula.

Implementation of factorial:

```
#!/bin/bash
# Recursive factorial function
          factorial()
{
   product=$1
   # Defining a function to calculate factorial using recursion
   if((product <= 2)); then
      echo $product
   else
      f=$((product -1))
# Recursive call
f=$(factorial $f)
f=$((f*product))
echo $f
fi
}
```

```
# main program
# reading the input from user
echo "Enter the number:"
read num

# defining a special case for 0! = 1
if((num == 0)); then
echo 1
else
#calling the function
factorial $num
fi
```

Method 2: Using for loop

Approach:

- Get a number
- Use for loop to compute the factorial by using the below formula
- fact(n) = n * n-1 * n-2 * …
- Display the result

Below is the Implementation using for loop:

```
# shell script for factorial of a number

# factorial using for loop

echo "Enter a number"

# Read the number

read num

fact=1

for((i=2;i<=num;i++))

{

  fact=$((fact * i))

}

echo $fact
```

Method 3: using do-while loop

- Get a number
- Use do-while loop to compute the factorial by using the below formula
- fact(n) = n * n-1 * n-2 * .. 1
- Display the result.

Implementation using a do-while loop.

```
# shell script for factorial of a number

# factorial using while loop

echo "Enter a number"

# Read the number
```

```
read num
fact=1
# -gt is used for '>' Greater than sign
while [ $num -gt 1 ]
do
  fact=$((fact * num))
  num=$((num - 1))
done
# Printing the value of the factorial
echo $fact
```

Experiment No. 8

Shell Script to check host configuration

Aim:	Shell Script that accepts the hostname and IP address as command line arguments and adds them to the /etc/hosts file.
Theory:	The hosts file (also referred to as etc\hosts) is a text file used by operating systems including windows to map IP addresses to host names/domain names. This file acts as a local DNS service for your local machine and it overrides the mappings from the DNS server to which your machine is connected over the network. The location of the hosts file depends on the operating system being used. The Application will not start if the IP address cannot be retrieved from a locally installed server or if the IP address cannot be resolved by the DNS.
ifconfig	The command ifconfig stands for interface configurator. This command enables us to initialize an interface, assign IP address, enable or disable an interface. It display route and network interface. You can view IP address, MAC address and MTU (Maximum Transmission Unit) with ifconfig command. A newer version of ifconfig is ip command. ifconfig command works for all the versions. **Syntax**: ifconfig Examples: n_g@cloudshell:~ (alien-bricolage-340809)$ ifconfig n_g @cloudshell:~ (alien-bricolage-340809)$ /usr/sbin/ifconfig -a
	1. **To get details of specific interface use commands** 1. ifconfig eth0 2. ifconfig lo **3.** ifconfig wlan0
	2. Host Command Linux host command displays domain name for given IP address or vice-versa. It also performs DNS lookups related to the DNS query. The host command's default behavior displays a summary of its command-line arguments and supported options.

Syntax:

The host command supports various command-line arguments and options. The basic syntax for the host command is as follows:

```
host <name>
```

where,

name: The name can be a domain name or an IP address (Ipv4 or Ipv6). It will lookup for the given name.

Example:

```
n_g@cloudshell:~ (alien-bricolage-340809)$ host www.google.co.in
```

# Display host name

Output:

```
www.google.co.in has address 142.251.10.94

www.google.co.in has IPv6 address 2404:6800:4003:c0f::5e
```

```
n_g@cloudshell:~ (alien-bricolage-340809)$ host -t ns www.vcet.edu.in
```

# Display domain name

Output:

```
www.vcet.edu.in is an alias for vcet.edu.in.

vcet.edu.in name server ns1.bluehost.com.

vcet.edu.in name server ns2.bluehost.com.
```

Conclusion:

Post Experiment question:	1. Explain the process of configuring a network interface in Linux. 2. Explain Process Management System Calls in Linux

CODE:

Sample script

```
#!/bin/bash
read -p "Enter host: " hostname
read -p "Enter IP address of the host: " ip
sed -i.bkp "$ a $hostname $ip" /etc/hosts
```

Output:

```
[root@nglinux ~]# chmod +x sample1.sh
[root@nglinux ~]# ./sample1.sh
Enter host: testhost1
Enter IP address of the host: 192.168.1.2
```

Check the host file:

```
[root@nglinux ~]# tail /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4 nglinux
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
testhost1 192.168.1.2
```

Final Script:

```
#!/bin/sh
    # PATH TO YOUR HOSTS FILE
    ETC_HOSTS=/etc/hosts
    # DEFAULT IP FOR HOSTNAME
    IP="127.0.0.1"
    # Hostname to add/remove.
    HOSTNAME=$1
        function removehost()
        {
            if [ -n "$(grep $HOSTNAME /etc/hosts)" ]
            then
                echo "$HOSTNAME Found in your $ETC_HOSTS, Removing now...";
                sudo sed -i ".bak" "/$HOSTNAME/d" $ETC_HOSTS
            else
                echo "$HOSTNAME was not found in your $ETC_HOSTS";
            fi
        }
            function addhost()
            {
                HOSTNAME=$1
                HOSTS_LINE="$IP\t$HOSTNAME"
                if [ -n "$(grep $HOSTNAME /etc/hosts)" ]
                then
                    echo "$HOSTNAME already exists: $(grep $HOSTNAME $ETC_HOSTS)"
                else
                    echo "Adding $HOSTNAME to your $ETC_HOSTS";
                    sudo -- sh -c -e "echo '$HOSTS_LINE' >> /etc/hosts";
                    if [ -n "$(grep $HOSTNAME /etc/hosts)" ]
                    then
                        echo "$HOSTNAME was added successfully \n $(grep $HOSTNAME /etc/hosts)";
                        else
                            echo "Failed to Add $HOSTNAME, Try again!";
                        fi
                    fi
            }
```

Experiment No. 9

Shell Script to count characters and words in a file

Aim:	Write awk script to find the number of characters, words, and lines in a file.
Theory:	Awk is a scripting language used for manipulating data and generating reports. The awk command programming language requires no compiling and allows the user to use variables, numeric functions, string functions, and logical operators. Awk is a utility that enables a programmer to write tiny but effective programs in the form of statements that define text patterns that are to be searched for in each line of a document and the action that is to be taken when a match is found within a line. Awk is mostly used for pattern scanning and processing. It searches one or more files to see if they contain lines that match the specified patterns and then performs the associated actions. Awk is abbreviated from the names of the developers – Aho, Weinberger, and Kernighan. WHAT CAN WE DO WITH AWK? 1. AWK Operations: (a) Scans a file line by line (b) Splits each input line into fields (c) Compares input line/fields to pattern (d) Performs action(s) on matched lines 2. Useful For: (a) Transform data files (b) Produce formatted reports 3. Programming Constructs: (a) Format output lines (b) Arithmetic and string operations

(c) Conditionals and loops

Syntax:

```
awk options 'selection _criteria {action }' input-file > output-file
```

Options:

```
-f program-file : Reads the AWK program source from the file
                  program-file, instead of from the
                  first command line argument.
-F fs           : Use fs for the input field separator
```

Examples:

Consider the following text file as the input file for all cases below:

```
n_g@cloudshell:~ (alien-bricolage-340809)$ cat>newdata.txt
LinuxLab Practical Termwork 50
DBMS Theory Oral 25
Datastructure Theory Oral 25
Maths Tutorial termwork 25
CS Oral Practical 25
^Z
```

1. Default behavior of Awk: By default Awk prints every line of data from the specified file.

$ awk '{print}' newdata.txt

```
n_g@cloudshell:~ (alien-bricolage-340809)$ awk '{print}' newdata.txt
```

Output:

```
LinuxLab Practical Termwork 50
DBMS Theory Oral 25
Datastructure Theory Oral 25
Maths Tutorial termwork 25
```

	CS Oral Practical 25
	2. Print the lines which match the given pattern. $ awk '/oral/ {print}' newdata.txt n_g@cloudshell:~ (alien-bricolage-340809)$ awk '/Theory/ {print}' newdata.txt Output: DBMS Theory Oral 25 Datastructure Theory Oral 25
	3. Splitting a Line Into Fields: For each record i.e. line, the awk command splits the record delimited by whitespace character by default and stores it in the $n variables. If the line has 4 words, it will be stored in $1, $2, $3 and $4 respectively. Also, $0 represents the whole line. $ awk '{print $1,$4}' newdata.txt n_g@cloudshell:~ (alien-bricolage-340809)$ awk '{print $1,$4}' newdata.txt Output: LinuxLab 50 DBMS 25 Datastructure 25 Maths 25 CS 25
Conclusion:	

Post Experiment questions	1.What are some important features provided by AWK? 2. What's the difference between using single quotes and double quotes when executing an AWK command?

CODE:

```
BEGIN{print "record.\t characters \t words"}
        #BODY section
        {
                len=length($0)
                total_len =total_len+len
                print(NR,":\t",len,":\t",NF,$0)
                words =NF
        }
          END{
                print("\n total")
                print("characters :\t" total_len)
                print("lines :\t" NR)
        }
```

OUTPUT:

```
n_g@cloudshell:~ (alien-bricolage-340809)$ awk -f countwords.awk sample.sh
record.  characters     words
1:     11:    1 #!/bin/bash
2:     32:    6 read -p "Enter host: " hostname
3:     43:    10 read -p "Enter IP address of the host: " ip
4:     46:    8 sudo sed -i.bkp "$ a $hostname $ip" /etc/hosts
5:     0:     0

 Total characters:    132
lines: 5
```

Experiment No.10
To setup and configure FTP server with VSFTPD on Ubuntu 20.04
Date of Performance:
Date of Submission:

Aim: To setup and configure FTP Server with VSFTPD on Ubuntu 20.04

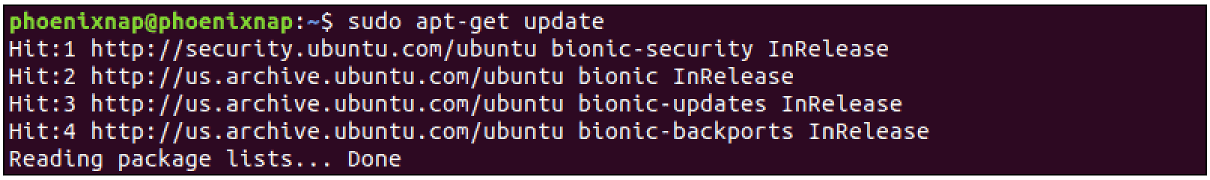
Theory: FTP (File Transfer Protocol) is a standard network protocol used to transfer files to and from a remote network. There are several open-source FTP servers available for Linux.

The most known and widely used are PureFTPd , ProFTPD , and vsftpd . We’ll be installing vsftpd (Very Secure Ftp Daemon), a stable, secure, and fast FTP server. Although FTP is a very popular protocol, for more secure and faster data transfers, we can use SCP or SFTP .

Installing vsftpd on Ubuntu 20.04

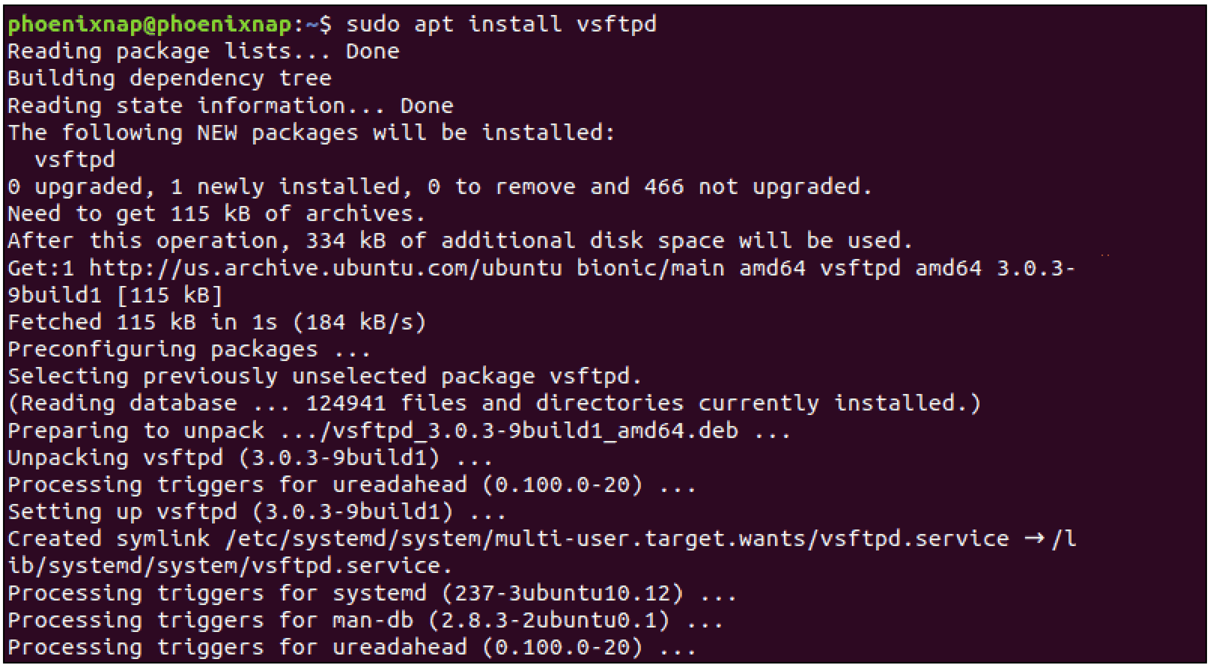
The vsftpd package is available in the Ubuntu repositories. To install it, execute the following commands:

$ sudo apt update



```
phoenixnap@phoenixnap:~$ sudo apt-get update
Hit:1 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease
Reading package lists... Done
```

$sudo apt install vsftpd



```
phoenixnap@phoenixnap:~$ sudo apt install vsftpd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  vsftpd
0 upgraded, 1 newly installed, 0 to remove and 466 not upgraded.
Need to get 115 kB of archives.
After this operation, 334 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 vsftpd amd64 3.0.3-
9build1 [115 kB]
Fetched 115 kB in 1s (184 kB/s)
Preconfiguring packages ...
Selecting previously unselected package vsftpd.
(Reading database ... 124941 files and directories currently installed.)
Preparing to unpack .../vsftpd_3.0.3-9build1_amd64.deb ...
Unpacking vsftpd (3.0.3-9build1) ...
Processing triggers for ureadahead (0.100.0-20) ...
Setting up vsftpd (3.0.3-9build1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/vsftpd.service → /l
ib/systemd/system/vsftpd.service.
Processing triggers for systemd (237-3ubuntu10.12) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ureadahead (0.100.0-20) ...
```

The ftp service will automatically start once the installation process is complete. To verify it, print the service status:

2. To launch the service and enable it at startup, run the commands:

sudo systemctl start vsftpd

sudo systemctl enable vsftpd

Step 3: Backup Configuration Files
Before making any changes, make sure to back up your configuration files.

1. Create a backup copy of the default configuration file by entering the following:

sudo cp /etc/vsftpd.conf /etc/vsftpd.conf_default

Step 4: Create FTP User
Create a new FTP user with the following commands:

sudo useradd -m testuser

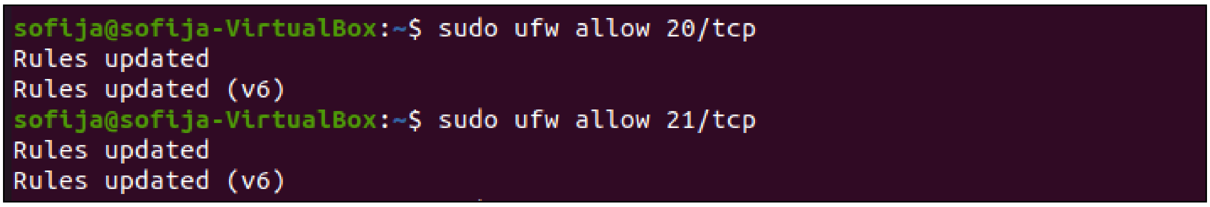
sudo passwd testuser

The system should ask you to create a password for the new testuser account.

Step 5: Configure Firewall to Allow FTP Traffic
If you are using UFW that comes standard with Ubuntu, it will block FTP traffic by default. Enter the following commands to open Ports 20 and 21 for FTP traffic:

sudo ufw allow 20/tcp

sudo ufw allow 21/tcp



```
sofija@sofija-VirtualBox:~$ sudo ufw allow 20/tcp
Rules updated
Rules updated (v6)
sofija@sofija-VirtualBox:~$ sudo ufw allow 21/tcp
Rules updated
Rules updated (v6)
```

Step 6: Connect to Ubuntu FTP Server
Connect to the FTP server with the following command:

sudo ftp ubuntu-ftp

Replace *ubuntu-ftp* with the name of your system (taken from the command line).

Log in using the testuser account and password you just set. You should now be successfully logged in to your FTP server.

Configuring and Securing Ubuntu vsftpd Server
Change Default Directory

By default, the FTP server uses the /srv/ftp directory as the default directory. You can change this by creating a new directory and changing the FTP user home directory.

To change the FTP home directory, enter the following:

```
sudo mkdir /srv/ftp/new_location
```

```
sudo usermod -d /srv/ftp/new_location ftp
```

Restart the vsftpd service to apply the changes:

```
sudo systemctl restart vsftpd.service
```

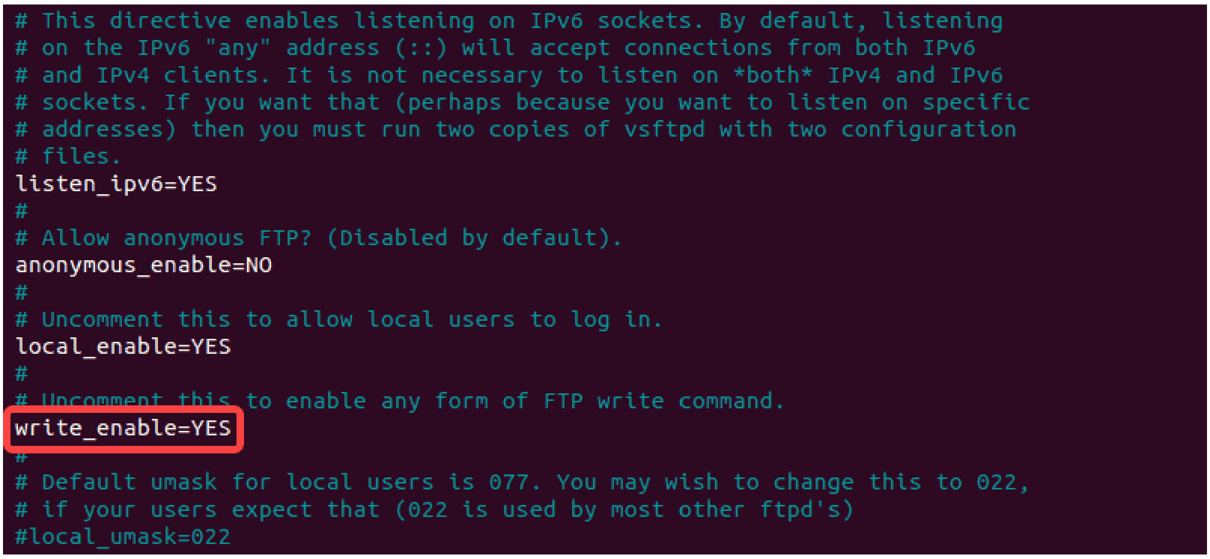
Now, you can put any files you want to share via FTP into the /srv/ftp folder (if you left it as the default), or the /srv/ftp/*new_location*/ directory (if you changed it).

Authenticate FTP Users

If you want to let authenticated users upload files, edit the vsftpd.conf file by entering the following:

```
sudo nano /etc/vsftpd.conf
```

Find the entry labeled *write_enable=NO*, and change the value to "YES."



```
# This directive enables listening on IPv6 sockets. By default, listening
# on the IPv6 "any" address (::) will accept connections from both IPv6
# and IPv4 clients. It is not necessary to listen on *both* IPv4 and IPv6
# sockets. If you want that (perhaps because you want to listen on specific
# addresses) then you must run two copies of vsftpd with two configuration
# files.
listen_ipv6=YES
#
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=NO
#
# Uncomment this to allow local users to log in.
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
write_enable=YES
#
# Default umask for local users is 077. You may wish to change this to 022,
# if your users expect that (022 is used by most other ftpd's)
#local_umask=022
```

Save the file, exit, then restart the FTP service with the following:

```
sudo systemctl restart vsftpd.service
```

This allows the user to make changes inside their home directory.

Securing FTP

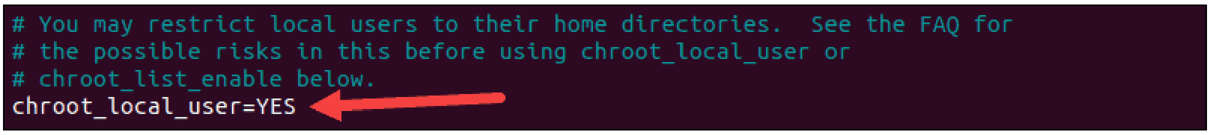
Numerous exploits take advantage of unsecured FTP servers. In response, there are several configuration options in vsftpd.conf that can help secure your FTP server.

Limit User Access

One method is to limit users to their home directory. Open vsftpd.conf in an editor and uncomment the following command:

```
chroot_local_user=YES
```

This is an example of the file in nano:



```
# You may restrict local users to their home directories.  See the FAQ for
# the possible risks in this before using chroot_local_user or
# chroot_list_enable below.
chroot_local_user=YES
```

Create a User List File

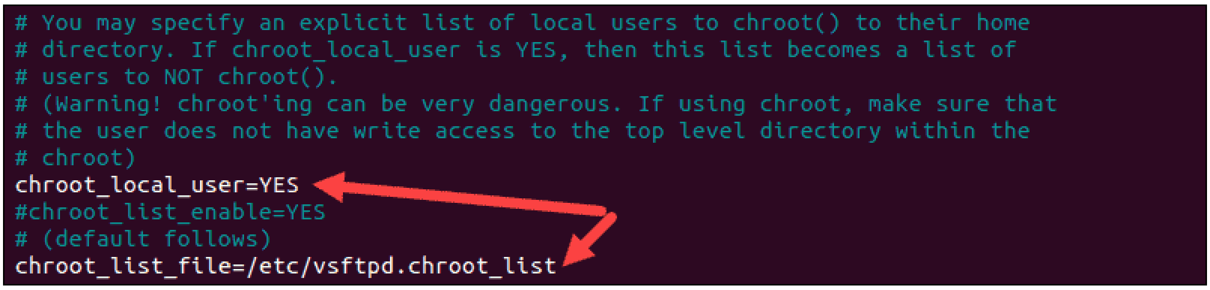
To create a list file, edit /etc/vsftpd.chroot_list, and add one user per line.

Instruct your FTP server to limit this list of users to their own home directories by editing vsftpd.conf:

```
chroot_local_user=YES
```

```
chroot_list_file=/etc/vsftpd.chroot_list
```

The image illustrates the edits that were made:



```
# You may specify an explicit list of local users to chroot() to their home
# directory. If chroot_local_user is YES, then this list becomes a list of
# users to NOT chroot().
# (Warning! chroot'ing can be very dangerous. If using chroot, make sure that
# the user does not have write access to the top level directory within the
# chroot)
chroot_local_user=YES
#chroot_list_enable=YES
# (default follows)
chroot_list_file=/etc/vsftpd.chroot_list
```

Restart the vsftpd service:

```
sudo systemctl restart vsftpd.service
```

By default, the list of blocked users from FTP access is stored in /etc/ftpusers. To add blocked users, edit this file and add one user per line.

Encrypt Traffic With FTPS

Another method to secure your FTP server is to encrypt the traffic. This is done by using FTPS – File Transfer Protocol over SSL (Secure Socket Layer).

For this to work, users need to be set up with a shell account on the FTP server. This will add a layer of secure encryption to your FTP traffic.

1. Start by creating a new certificate with openssl. To do so, run the command:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/ssl/private/vsftpd.pem -out /etc/ssl/private/vsftpd.pem
```

2. Provide the required information when prompted or keep the default configuration by pressing Enter.

3. Next, open your vsftpd.conf file in an editor and change the line `ssl_enable=NO` to `ssl_enable=YES`:

```
ssl_enable=YES
```

4. Then, add the following lines:

```
rsa_cert_file=/etc/ssl/private/vsftpd.pem

rsa_private_key_file=/etc/ssl/private/vsftpd.pem

allow_anon_ssl=NO

force_local_data_ssl=YES

force_local_logins_ssl=YES

ssl_tlsv1=YES

ssl_sslv2=NO

ssl_sslv3=NO

require_ssl_reuse=NO

ssl_ciphers=HIGH

pasv_min_port=40000
```

```
pasv_max_port=50000
```

5. Save the changes and exit the file.

6. Finally, restart the service to apply the changes:

```
sudo systemctl restart vsftpd.service
```

CONCLUSION:

Experiment No.11
To setting up and configuring Linux Mail server
Date of Performance:
Date of Submission:

Aim: To setting up and configuring Linux Mail Server.

Theory:

Linux Email Server Components

There are three components to a mail service on a Linux email server:

- Mail user agent (MUA) is the GUI, the part that lets you write and send emails, like Thunderbird or Outlook.
- Mail transport agent (MTA) is the bit that moves the mail (as the name suggests). MTAs like Sendmail and Postfix are the parts that waft your communications from place to place through the ether.
- Mail delivery agent (MDA) is the component that sends out messages sent to you on your local machine, so they get to the appropriate user mailbox. Postfix-maildrop and Procmail are examples.

Setup Linux Email Server

In order to configure a Linux mail server, you'll first need to check if Postfix is already installed. It's the default mail server on the lion's share of Linux distributions these days, which is good because server admins like it a lot. Here's how to check if it's already on the system:

```
$ rpm -qa | grep postfix
```

If not, this is how you install it on Red Hat distributions:

```
$ dnf -y install postfix
```

Next, run it and activate it on system start-up:

```
$ systemctl start postfix
$ systemctl activate postfix
```

For distributions based on Debian, like Ubuntu, you'd install them like this:

```
$ apt-get -y install postfix
```

As you configure Linux mail server you will receive a prompt to choose how you want to configure your Postfix mail server.

You'll be presented with these choices:

- No configuration
- Internet site
- Internet with smarthost
- Satellite system and Local only

Let's go with the No configuration option for our Linux email server.

Configure Linux Mail Server

After installing the Postfix mail server, you will need to set it up, and most of the files you'll need for this can be found inside the /etc/postfix/ directory. You can find the main configuration for Postfix Linux mail server in the /etc/postfix/main.cf file. This file contains numerous options like:

myhostname

Use this one to specify the hostname of the mail server, which is where postfix will obtain its emails. The hostnames will look something like mail.mydomain.com, smtp.mydomain.com. You incorporate the hostname this way:

myhostname = mail.mydomain.com

exampledomain.com

This option is the mail domain that you will be servicing, like mydomain.com

The syntax looks like this:

mydomaindomain.com = mydomain.com

myorigin

All emails sent from this mail server will look as though they came from the one that you specify in this option. You can set this to $exampledomain.com.

myorigin = $exampledomain.com

Use any value that you want for this option but put a dollar sign in front of it like this: $exampledomain.com.

mydestination

This option shows you which domains the Postfix server uses for incoming emails to your Linux email server. You can assign values like this:

mydestination = $myhostname, localhost.$exampledomain.com, $exampledomain.com, mail.$exampledomain.com, www.$exampledomain.com

mail_spool_directory

A Postfix Linux mail server can use two modes of delivery:

- straight to someone's mailbox.
- to a central spool directory, which means the mail will sit in /var/spool/mail with a file for every user.

mail_spool_directory = /var/spool/mail

mynetworks

This will let you arrange which servers can relay through your Postfix server. It should only take local addresses like local mail scripts on your server. If this isn't the case, then spammers can piggyback on your Linux mail server. That means your lovely shiny server will be doing the heavy lifting for some bad guys and it will also end up getting banned. Here's the syntax for this option:

mynetworks = 127.0.0.0/8, 192.168.1.0/24

smtpd_banner

This one determines what message is sent after the client connects successfully. Consider changing the banner so it doesn't give away any potentially compromising information about your server.

inet_protocols

This option designates which IP protocol version is used for server connections.

inet_protocols = ipv4

When you change any of files used to configure Linux mail server for Postfix, you must reload the service, with this directive:

```
$ systemctl reload postfix
```

Of course, we all get distracted and typing things in can often result in mistakes, but you can track down any misspellings that might compromise your Linux mail server using this command:

```
$ postfix check
```

Checking the Mail Queue
Things like network failure (and many other reasons) can mean that the mail queue on your Linux email server can end up getting full, but you can check the Postfix mail queue with this command:

```
$ mailq
```

If that reveals that its full then you can flush the queue using this command:

```
$ postfix flush
```

Look at it again and you should see that your Linux email server queue is clear.

Test Linux Mail Server
Once your configuration is done you need to test your Linux mail server. The first thing to do is use a local mail user agent such as mailx or mail which is a symlink to mailx. Send your first test to someone on the Linux mail server and if that works then send the next one to somewhere external.

```
$ echo "This is the body of the message" | mailx -s "Here we have a Subject" -r "for instance <small example@mydomain.com>" -a /path/to/attachment someone@mydomain.com
```

Then check if your Linux email server can pick up external mail. If you run into any snags, have a peek at the logs. The Red Hat log file can be found in /var/log/maillog and for Debian versions in /var/log/mail.log, or wherever else the rsyslogd configuration specifies. I would suggest you review the Linux syslog server for an in-depth clarification on logs and how to set up rsyslogd. If you run into any more difficulties, take a look at your DNS settings and use Linux network commands to check your MX records.

Fight Spam with SpamAssassin
Nobody likes spam, and SpamAssassin is probably the best free, open source spam fighting ninja that you could hope to have in your corner. Installing it is as simple as doing this:

```
$ dnf -y install spamassassin
```

Then you just start the service and activate it at start-up:

```
$ systemctl start spamassassin
```

```
$ systemctl activate spamassassin
```

Once you've done that, you can see how it's configured in the /etc/mail/spamassassin/local.cf file. SpamAssassin runs a number of scripts to test how spammy an email is. The higher the score that the scripts deliver, the more chances there are that it's spam.

In the configuration file, if the parameter required_hits is 6, this tells you that SpamAssassin will consider an email to be spam if it scores 6 or more.

The report_safe command will have values of 0, 1, or 2. A 0 tells you that email marked as spam is sent without modification, and only the headers will label it as spam. A 1 or a 2 means that a new report message will be created by SpamAssassin and delivered to the recipient. A value of 1 indicates that the spam message is coded as content message/rfc822, and if it's a 2, that means the message has been coded as text or plain content. Text or plain is less dangerous because some mail clients execute message/rfc822, which is not good if they contain any kind of malware. The next thing to do is integrate it into Postfix, and the easiest way to do that is with procmail.

We'll make a file called/etc/procmailrc, and add this to it:

```
:0 hbfw | /usr/bin/spamc
```

Then we'll edit the Postfix configuration file /etc/postfix/main.cf and alter the mailbox_command, thus:

```
mailbox_command = /usr/bin/procmail
```

Last but not least, restart Postfix and SpamAssassin services:

```
$ systemctl restart postfix
```

```
$ systemctl restart spamassassin
```

Unfortunately, SpamAssassin can't catch everything, and spam messages can still sneak through to fill up the mailboxes on your Linux email server. But never fear because you can filter messages before they even get to the Postfix server with Realtime Blackhole Lists (RBLs). Open the Postfix server configuration at /etc/postfix/main.cf and change smtpd_recipient_restrictions option by adding the following options like this:

```
strict_rfc821_envelopes = yes
relay_domains_reject_code = 554
unknown_address_reject_code = 554
unknown_client_reject_code = 554
unknown_hostname_reject_code = 554
unknown_local_recipient_reject_code = 554
unknown_relay_recipient_reject_code = 554
unverified_recipient_reject_code = 554
smtpd_recipient_restrictions =
reject_invalid_hostname,
reject_unknown_recipient_domain,
reject_unauth_pipelining,
permit_mynetworks,
permit_sasl_authenticated,
reject_unauth_destination,
reject_rbl_client dsn.rfc-ignorant.org,
reject_rbl_client dul.dnsbl.sorbs.net,
reject_rbl_client list.dsbl.org,
reject_rbl_client sbl-xbl.spamhaus.org,
reject_rbl_client bl.spamcop.net,
reject_rbl_client dnsbl.sorbs.net,
permit
```

Now, restart your postfix Linux mail server:

```
$ systemctl restart postfix
```

The above RBLs are the most common ones found, but there are plenty more on the web for you to track down and try.

POP3 and IMAP Protocol Basics

We now know how a SMTP Linux mail server sends and receives emails, but what about other user needs, like when they want local copies of emails to view off-line? mbox file format isn't supported; it's used by many mail user agents such as mailx and mutt. Due to security concerns, some mail servers restrict access to the shared mail spool directories. Another class of protocols—called mail access protocols—was introduced to deal with such situations. The commonest ones are POP and IMAP – Post Office Protocol and Internet Message Access Protocol. POP's underlying methodology is very simple: a central Linux mail server is online 24/7 for reception and storage of all user emails. When an email is sent, the email client relays it through the central Linux mail server using SMTP. Be aware that the SMTP server and POP server can easily be on the same system, and that this is a common thing to do. IMAP was developed because previously you couldn't keep a master copy of a user's email on the server. With IMAP, your Linux email server supports three kinds of access:

- online mode is like having direct access to the Linux email server file system.
- offline mode feels like POP, where the client only connects to the network to get their mail, and the server won't keep a copy.
- disconnected mode lets users keep cached copies of their emails and the server keeps one too.

There are a few different implementations for IMAP and POP, with the most prevalent being dovecot server, which offers both.

POP3, POP3S, IMAP, and IMAPS listen on ports 110, 995, 143, and 993 respectively.

Dovecot Installation

Dovecot is preinstalled on the majority of Linux distributions, and there's no problem putting it in Red Hat too:

```
$ dnf -y install dovecot
```

For Debian, a pair of packages provide the IMAP and POP3 functionality. Here's how to install them:

```
$ apt-get -y install dovecot-imapd dovecot-pop3d
```

You will be prompted to create self-signed certificates for using IMAP and POP3 over SSL/TLS. Select yes and type in the hostname of your system when asked to do so.

Then you can run the service and activate it at start-up like this:

```
$ systemctl start dovecot
```

```
$ systemctl activate dovecot
```

Configure Dovecot

The main configuration file for Dovecot is /etc/dovecot/dovecot.conf file.

Some varieties of Linux keep the configuration in the/etc/dovecot/conf.d/ directory and then have the include directive include the settings in the files.

Here are a few of the parameters used to configure dovecot:

protocols: the ones you want to support.

```
protocols = imap pop3 lmtp
```

lmtp stands for local mail transfer protocol.

listen: IP addresses to listen on.

```
listen = *, ::
```

The asterisk means all ipv4 interfaces and :: means all ipv6 interfaces

userdb: user database to authenticate users.

```
userdb { driver = pam }
```

passdb: password database two authenticate users.

```
passdb { driver = passwd }
```

mail_location: this entry is in the /etc/dovecot/conf.d/10-mail.conf file, and it's written like this:

```
mail_location = mbox:~/mail:INBOX=/var/mail/%u
```

Secure Dovecot

Dovecot features generic SSL certificates and key files used with /etc/dovecot/conf.d/10-ssl.conf

```
ssl_cert = </etc/pki/dovecot/certs/dovecot.pem
```

```
ssl_key = </etc/pki/dovecot/private/dovecot.pem
```

If you try to connect to a dovecot server and certificates haven’t been signed, then you’ll get a warning, but if you go to a certificate authority you can buy one, so no worries there.

Alternatively, you can point to them using Let’s Encrypt certificates:

```
ssl_cert = </etc/letsencrypt/live/yourdomain.com/fullchain.pem

ssl_key = </etc/letsencrypt/live/yourdomain.com/privkey.pem
```

You’ll need to open dovecot server ports in your iptables firewall by adding iptables rules for ports 110, 995, 143, 993, 25.

Do that and save the rules.

Or if you have a firewall then do this:

```
$ firewall-cmd --permanent --add-port=110/tcp --add-port=995/tcp

$ firewall-cmd --permanent --add-port=143/tcp --add-port=993/tcp

$ firewall-cmd --reload
```

Finally, for troubleshooting, check through the log files /var/log/messages, /var/log/maillog, and /var/log/mail.log files.

Linux mail server (and particularly Postfix) is one of the simplest systems you can work with.

Conclusion:

Experiment No.12
Configuration of DNS server with domain name in Linux
Date of Performance:
Date of Submission:

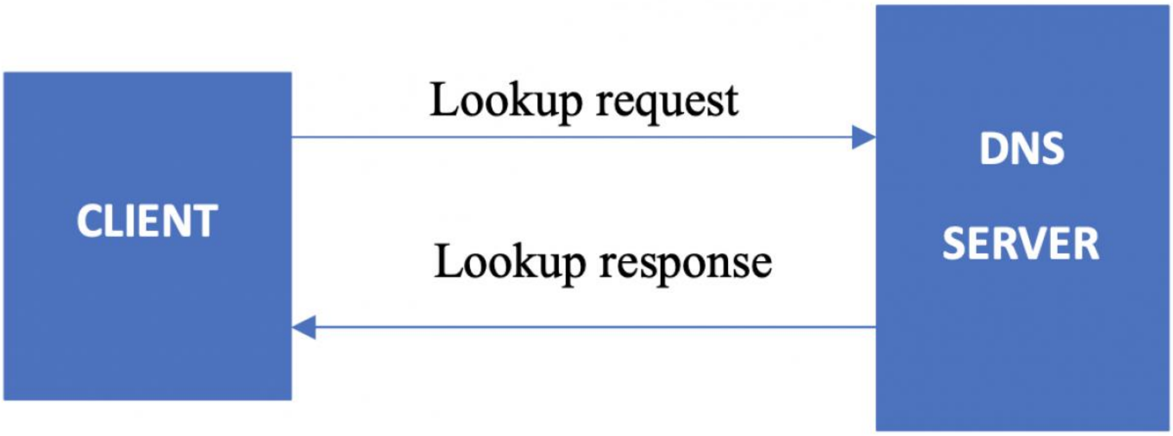
Aim: Configuration of DNS server with domain name in Linux

Theory:

The Domain Name System (DNS) is used to resolve (translate) hostnames to internet protocol (IP) addresses and vice versa. A DNS server, also known as a nameserver, maps IP addresses to hostnames or domain names.

How DNS works?

When a client requests information from a nameserver, it usually connects to port 53, and then the nameserver resolves the name requested.



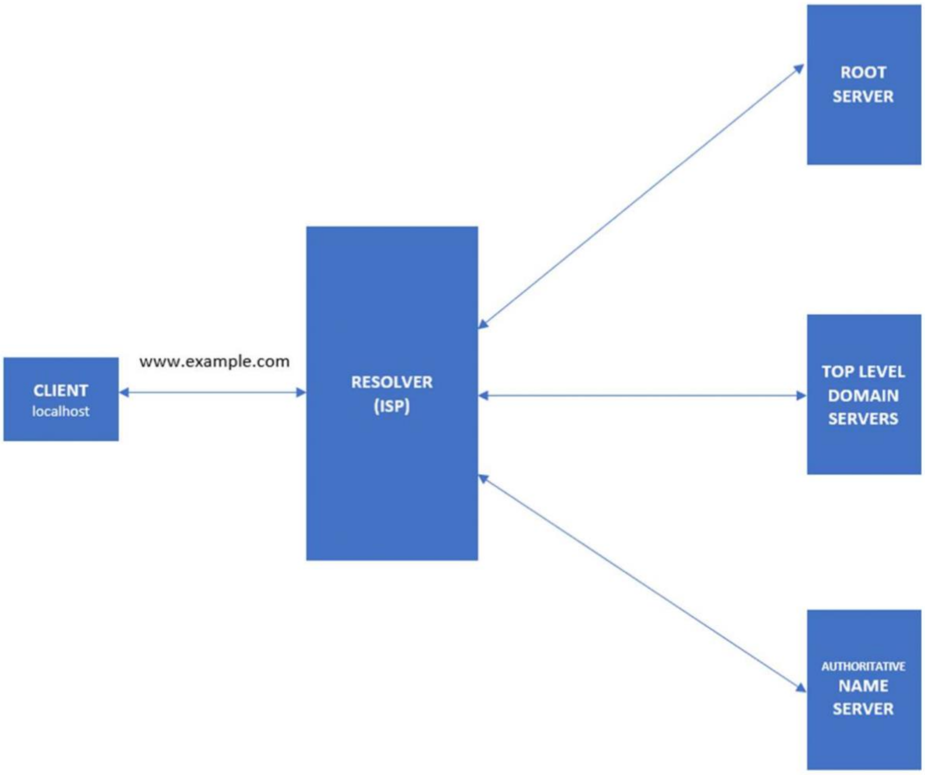
- Sending a request from the DNS client to the DNS server is called a lookup request.
- Getting a response from the DNS server to the DNS client is called a lookup response.
- The system on which the DNS service is configured is called a DNS server.
- The system that accesses the DNS server is called a DNS client.

Where does DNS get IP addresses?

You might wonder how DNS gets the IP of the corresponding hostname or domain name. How does DNS search among different IP addresses and associate your domain name correctly? Who stores those mappings between domain names and IP addresses?

The DNS workflow illustrates how communication happens within DNS and how it resolves the addresses.

- When the client searches for the domain www.example.com, the request will initially go to the internet service provider's (ISP) resolver. It will respond to the user's request to resolve a domain name.
- If the IP address is not found on the resolver, the request is forwarded to a root DNS server and later to the top-level domain (TLD) servers.
- TLD servers store information for top-level domains, such as .com or .net.

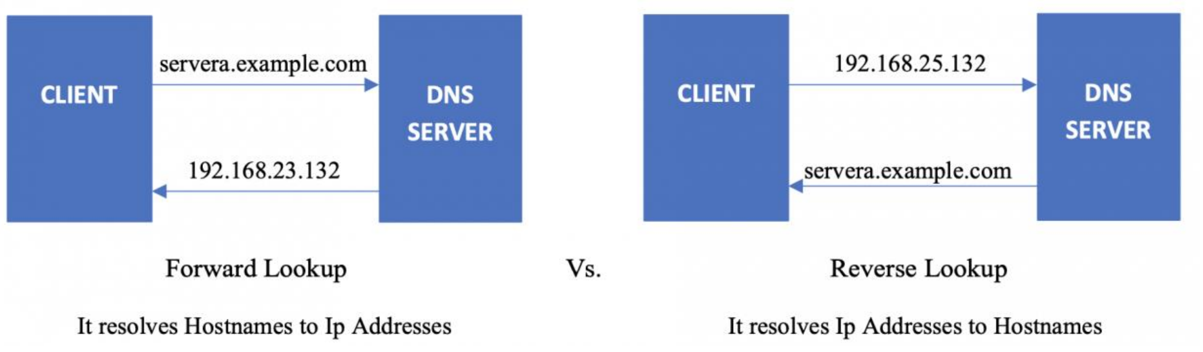


- Requests are forwarded to the nameservers, which know detailed information about domains and IP addresses.
- Nameservers respond to the ISP's resolver, and then the resolver responds to the client with the requested IP.

- When the resolver doesn't know the IP, it stores the IP and its domain in a cache to service future queries.

Forward and reverse lookups

The forward lookup zone uses the domain name to search for IP addresses, whereas the reverse lookup zone uses IP addresses to search for the domain name.



Install and configure DNS

BIND is a nameserver service responsible for performing domain-name-to-IP conversion on Linux-based DNS servers.

```
[root@servera ~] # yum install bind
```

The BIND package provides the named service. It reads the configuration from the /etc/named and /etc/named.conf files. Once this package is installed, you can start configuring DNS.

Configure the /etc/named.conf file

First, add or edit the two values in the options field. One is the DNS server address, and the other is the allow-query to any.

```
[root@servera ~] # vim /etc/named.conf

listen-on port 53 { 127.0.0.1; 192.168.25.132; };

allow-query { localhost; any; };
```

Here are the values from the above file:

- 192.168.25.132 – DNS server address
- any – matches every IP address

Define the forward and reverse zones

Define the forward and reverse zones in the /etc/named.conf or /etc/named.rfc1912.zones (you can define zones in either of those files). In this example, I am appending zone definition details to the /etc/named.rfc1912.zones file.

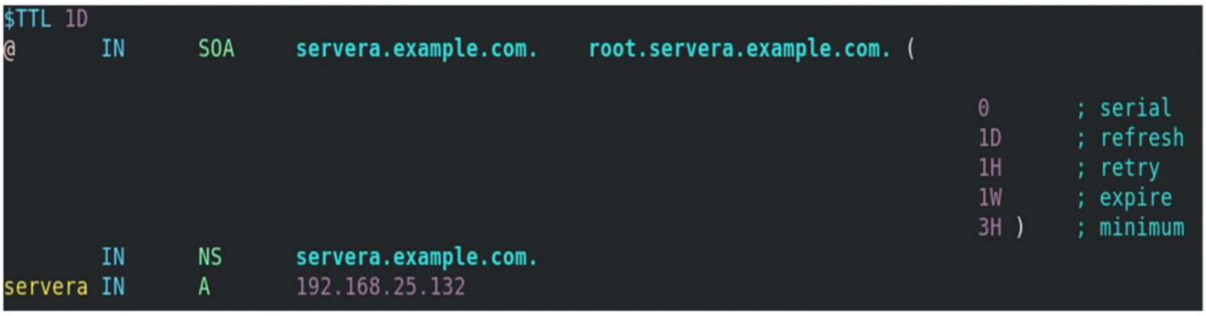
```
[root@servera ~] # vim /etc/named.rfc1912.zones

 zone "example.com" IN { type master;

 file "example.forward.zone";

 allow-update { none; };

};

 zone "25.168.192.in-addr.arpa" IN {

  type master;

  file "example.reverse.zone";

  allow-update { none; };

};
```

Create forward and reverse zone files

You also need to create forward and reverse zone files in the /var/named directory.

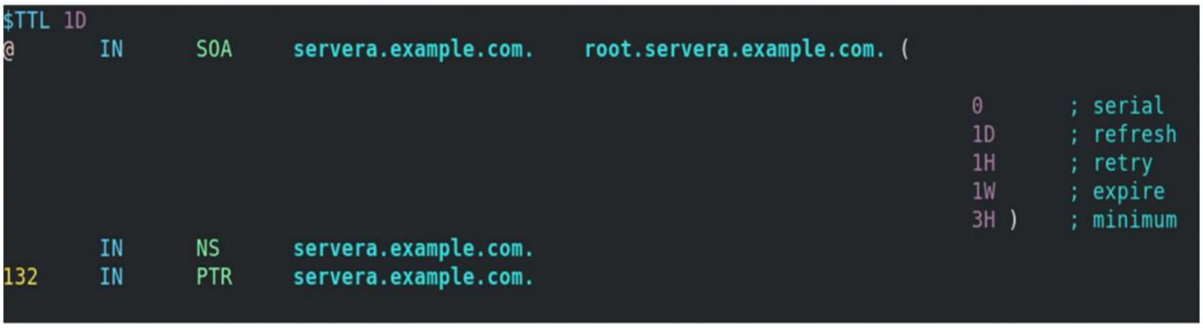
Note: By default, the named.conf file includes the /var/named directory for checking zone files. Sample zone files named.localhost and named.loopback are created during the installation of the BIND package.

```
[root@servera ~] # vim /var/named/example.forward.zone
```



```
$TTL 1D
@       IN      SOA     servera.example.com.    root.servera.example.com. (
                                                                        0       ; serial
                                                                        1D      ; refresh
                                                                        1H      ; retry
                                                                        1W      ; expire
                                                                        3H )    ; minimum
        IN      NS      servera.example.com.
servera IN      A       192.168.25.132
```

```
[root@servera ~] # vim /var/named/example.reverse.zone
```



```
$TTL 1D
@       IN      SOA     servera.example.com.    root.servera.example.com. (
                                                        0       ; serial
                                                        1D      ; refresh
                                                        1H      ; retry
                                                        1W      ; expire
                                                        3H )    ; minimum
        IN      NS      servera.example.com.
132     IN      PTR     servera.example.com.
```

Add the nameserver IP to /etc/resolv.conf

First, you must disable DNS processing by NetworkManager because it dynamically updates the /etc/resolv.conf file with DNS settings from its active connection profiles. To disable this and allow manual editing of /etc/resolv.conf, you must create a file (For example, 90-dns-none.conf), as root in the /etc/NetworkManager/conf.d/ directory that contains the following:

```
[main]

dns=none
```

Save the file and reload (restart) NetworkManager.

```
# systemctl reload NetworkManager
```

After you reload NetworkManager, it won't update /etc/resolv.conf. Now, you can manually add the nameserver's IP address to the /etc/resolv.conf file.

```
[root@servera ~] # vim /etc/resolv.conf

# Generated by NetworkManager

search localdomain example.com

nameserver 192.168.25.132
```

[Be prepared in case something goes wrong. Read An introduction to DNS troubleshooting.]

Start/restart and enable the named service

If the named service is not running or is disabled, then start and enable it. If it is already active (running) and you made all these configurations, you need to restart the service to make changes.

```
[root@servera ~] # systemctl status named.service

[root@servera ~] # systemctl start named.service

[root@servera ~] # systemctl enable named.service
```

```
[root@servera ~] # systemctl restart named.service
```

Verify the DNS name resolution

You have installed the BIND package, configured named files, created lookup zones, and restarted the service to make configurations take effect. Now use the nslookup and dig commands to check whether DNS is working properly and verify whether you are getting the intended results.

- nslookup is a program to query internet domain name servers.
- dig is a tool for interrogating DNS servers. It performs DNS lookups and displays the answers that are returned from the nameserver.

Query with nslookup

```
[root@servera ~] # nslookup servera.example.com

  Server: 192.168.25.132

  Address: 192.168.25.132#53

  Name: servera.example.com

  Address: 192.168.25.132

[root@servera ~] # nslookup 192.168.25.132

   132.25.168.192.in-addr.arpa name = servera.example.com.
```

Query with dig

Here is a forward lookup, where DNS responds with 192.168.11.132 as an IP for servera.example.com:

```
[root@servera ~] # dig servera.example.com

   ...output truncated...

;; ANSWER SECTION:

servera.example.com. 86400 IN A 192.168.25.132


;; AUTHORITY SECTION:
```

```
example.com. 86400 IN NS servera.example.com.


...output truncated...
```

This example displays a reverse lookup, where the DNS server responds with servera.example.com as the domain name for 192.168.25.132:

```
[root@servera ~] # dig -x 192.168.25.132
  ...output truncated...


;; ANSWER SECTION:
132.25.168.192.in-addr.arpa. 86400 IN PTR servera.example.com.


;; AUTHORITY SECTION:
25.168.192.in-addr.arpa. 86400 IN NS servera.example.com.


;; ADDITIONAL SECTION:
servera.example.com. 86400 IN A 192.168.25.132


...output truncated...
```

Conclusion: