B. Tech. (Third-Year)

Department of Computer Science and Information Technology

Lab Manual of Linux

Subject Code: CSIT-505

B. Tech: III Year 5th Sem

Name of the Student : Prakhar Jain

Branch & Section : B.Tech (CSIT)

Roll No. : 0827CI191044

Year : III year

Course Learning Objectives (CLO)

The Learning Objectives of **Linux** are such that the student will understand and learn the following things.

CLO1: To learn about the Basic Linux Concepts

CLO2: Students will become familiar with the Linux commands

CLO3: To learn Shell programming

CLO4: To learn about kernel

CLO5: Execute programs written in C under UNIX environment

Course Outcomes (CO)

At the end of the course, student would be able to

CO1: Understand the system calls

CO2: Compare between ANSI C AND C++ AND POSIX standards

CO3: To learn basic concepts of memory & scheduling

CO4: Mapping the relationship between UNIX Kernel support for files

CO5: Understand Kernel support for process creation and termination and memory allocation

cropolis Institute of Technology and Research, Indore

Department of Computer Science and Information Technology

LAB MANUAL

LIST OF PROGRAMS

S.NO	NAME OF EXPERIMENTS
1.	To Study basic & User status Unix/Linux Commands.
2.	Study & use of commands for performing arithmetic operations with Unix/Linux.
3.	Create a file called wlcc.txt with some lines and display how many lines, words and characters are present in that file.
4.	Append ten more simple lines to the wlcc.txt file created above and split the appended file into 3 parts. What will be the names of these split files? Display the contents of each of these files. How many lines will be there on the last file?
5.	Execute shell commands through vi editor.
6.	Write a Shell Script that accepts a filename, starting and ending line numbers as arguments and displays all lines between the given line numbers.
7.	Write a shell script that deletes all lines containing a specific word in one or more files supplied as arguments to it.
8.	Write a shell script that displays a list of files in the current directory to which the user has read, write and execute permissions.
9.	Write a shell script that accepts one or more file names as arguments and converts all of them to uppercase, provided they exist in the current directory.
10.	Write a shell script that computes the gross salary of an employee according to the following rules: i) If basic salary is <1500 then HRA=10% of the basic and DA=90% of the basic ii) If the basic salary is >=1500 then HRA=500/- and DA=98% of the basic
11.	Write an interactive file –handling shell programs. Let it offer the user the choice of copying, removing, renaming, or linking files. Once the user has made a choice, have the same program ask the user for the necessary information, such as the file name ,new name and so on.
12.	Write a shell script that accepts any number of arguments and prints them in the reverse order.

13.	Write a Shell script to count the number of lines in a file that do not contain vowels.
14.	Write a shell script which receives two file names as arguments. It should check whether the two file contents are the same or not. If they are the same then the second file should be deleted.
15.	Develop an interactive script that asks for a word and a file name and then tells how many times that word occurred in the file.
16.	Write a shell script to perform the following string operations: i. To extract a substring from a given string. ii. To find the length of a given string.
17.	Write a shell script that accepts two integers as its arguments and computes the value of the first number raised to the power of the second number.
18.	Write a shell script that takes a command –line argument and reports on whether it is directory, a file, or something else.
19.	Develop an interactive grep script that asks for a word and a file name and then tells how many lines contain that word.
20.	Write a shell script to list all of the directory files in a directory.
21.	Write and execute all process control commands.
22.	Study & Installation of SAMBA, APACHE, TOMCAT.
23.	Study & installation of Firewall & Proxy server.

EXPERIMENT NO: 1

AIM: To Study basic & User status UNIX/Linux Commands.

INTRODUCTION:

1. pwd Command

The pwd command is used to display the location of the current working directory.

2. mkdir Command

The mkdir command is used to create a new directory under any directory.

3. rmdir Command

The rmdir command is used to delete a directory.

4. ls Command

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

5. cd Command

The cd command is used to change the current directory.

6. touch Command

The touch command is used to create empty files. We can create multiple empty files by executing it once.

7. cat Command

It can be used to create a file,display content of the file, copy the content of one file to another file, and more.

9. clear Command

Linux **clear** command is used to clear the terminal screen.

10. time Command

The time command is used to display the time to execute a command.

11.date Command

The date command is used to display date, time, time zone, and more.

12.cal Command

The cal command is used to display the current month's calendar with the current date highlighted.

13. cut Command
The cut command is used to select a specific column of a file.

14. grep Command
The 'grep' stands for "**global regular expression print**." It is useful for searching the content from a file. Generally, it is used with the pipe.

15. sed command
The sed command is also known as **stream editor**. It is used to edit files using a regular expression. It does not permanently edit files; instead, the edited content remains only on display. It does not affect the actual file.

16. tee command
The tee command is quite similar to the cat command. The only difference between both filters is that it puts standard input on standard output and also write them into a file.

17. comm Command
The 'comm' command is used to compare two files or streams. By default, it displays three columns, first displays non-matching items of the first file, second indicates the non-matching item of the second file, and the third column displays the matching items of both files.

EXPERIMENT NO: 2

AIM: Study & use of commands for performing arithmetic operations with Unix/Linux.

Operator	Description	Example
+ (Addition)	**Adds values on either side of the operator**	**$((a + b))**
-(Subtraction)	**Subtracts right hand operand from left hand operand**	**$((a - b))**
***(Multiplication)**	**Multiplies values on either side of the operator**	**$((a * b))**
/ (Division)	**Divides left hand operand by right hand operand**	**$((b / a))**
% (Modulus)	**Divides left hand operand by right hand operand and returns remainder**	**$((b % a))**
= (Assignment)	**Assigns right operand in left operand**	**a = $b would assign value of b into a**

Program:

```
echo "enter a"

read a

echo "enter b"
read b

val=$(( a + b))
echo "a + b: $val"

val=$((a - b))
echo "a - b: $val"

val=$((a * b))
echo "a * b: $val"

val=$((b / a))
echo "b / a: $val"

val=$((b % a))
echo "b % a: $val"
fi
```

Output :

```
enter a
3
enter b
2
a + b: 5
a - b: 1
a * b: 6
b / a: 1
b % a: 1
```

EXPERIMENT NO: 3

AIM: Create a file called wlcc.txt with some lines and display how many lines, words and characters are present in that file

INTRODUCTION:
We have used the 'wlcc.txt' file for testing the commands. Let's find out the output of the file using cat commond

Program

[root@wlcc ~]# cat wlcc.txt

Red Hat

CentOS

Fedora

Debian

Scientific Linux

OpenSuse

Ubuntu

Xubuntu

Linux Mint

Pearl Linux

Slackware

Mandriva

- **Count Number of Lines**

  [root@wlcc ~]# wc -l wlcc.txt

  Output

  12 wlcc.txt

- **Display Number of Words**

[root@wlcc ~]# wc -w wlcc.txt

Output

16 wlcc.txt

- **Count Number of Bytes and Characters**

[root@wlcc ~]# wc -c wlcc.txt

Output

112 tecmint.txt

[root@wlcc ~]# wc -m wlcc.txt

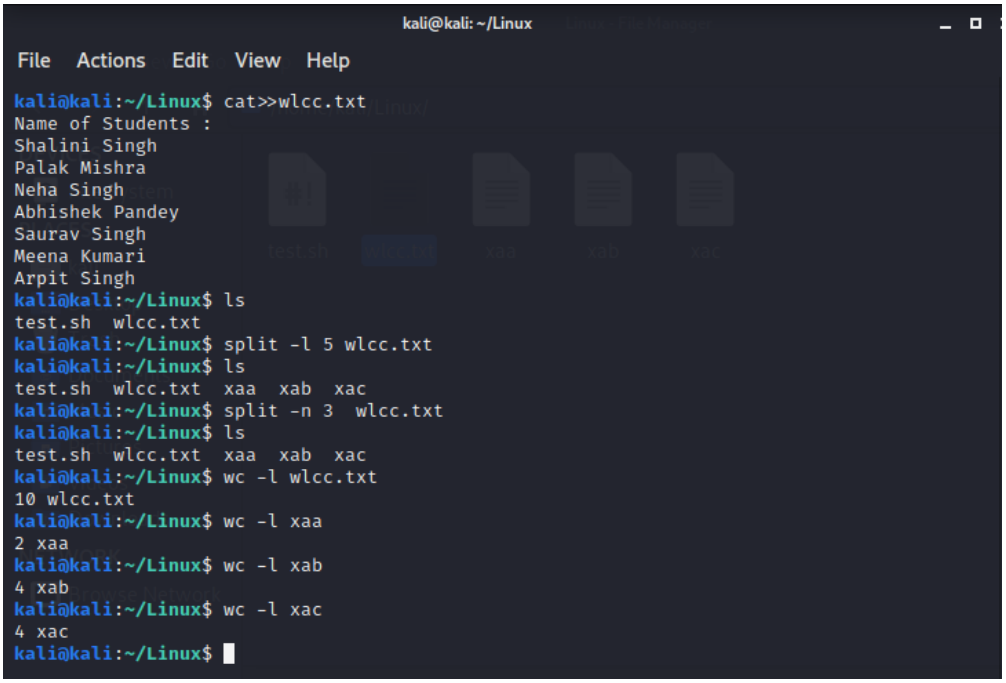
Output

112 wlcc.txt

EXPERIMENT NO: 4

AIM: Append ten more simple lines to the wlcc.txt file created above and split the appended file into 3 parts. What will be the names of these split files? Display the contents of each of these files. How many lines will be there on the last file?

INTRODUCTION:
Cat >> filename command is used to append ten more simple lines to the wlcc.txt file.
split <filename> command is used to split file.
split -l 5 <file name>
Split -n 3 <file name>
The option -l specifies the number of lines per output file. Since the input file contains 7 lines, the output files contain 3, 3 and 1 respectively. The output files generated are:
Names of these split files are:-



Here the names of the files are PREFIXaa. There will be three line in last line.

EXPERIMENT NO: 5

Aim: Execute shell commands through vi editor**.**

Solution:

i) To Launch a VI Editor

Command - vi <FILENAME>

ii) Save a File and Quit

Command – wq

iii) By typing 'i' user go into the insert mode and make some changes.

ESC - For termination Insert Mode

u – Undo last change

U – Undo all changes to the entire line

o – Open a new line (goes into insert mode)

dd – Delete line

3dd – Delete 3 lines.

D – Delete contents of line after the cursor

C – Delete contents of a line after the cursor and insert

new text. Press ESC key to end insertion.

dw – Delete word

4dw – Delete 4 words

cw – Change word

x – Delete character at the cursor

r – Replace character

Experiment no. 6

Aim: Write a shell script that accepts a file name, starting and

ending line numbers as arguments and displays all lines between the given line numbers.

Solution:

Syntax to see lines between given line numbers

Command: sed -n<start>, <end>\p<filename>

echo "enter the filename"

read fname

echo "enter the starting line number"

read s

echo "enter the ending line number"

read n

sed -n $s,$n\p $fname | cat > newline

Experiment no. 7

Aim: Write a shell script that deletes all lines containing a specific word in one or more files supplied as arguments to it.

Solution: echo “Enter a word:”

read word

for file in $*

do

grep –iv “$word” $file | tee 1> /dev/null

done

echo “Lines containing given word are deleted.”

Experiment no. 8

Aim: Write a shell script that displays a list of files in current directory to which the user has read, write and execute permissions.

Solution:

We have to check every file in current directory and display the name that has read, write and execution permission.

```
// Iterate through all the files
for file in *
do
if [-f $file]
then
// Check for permission
if [-r $file -a -w $file -a -x $file]
ls -l $file
fi
```

Experiment no. 9

Aim: Write a shell script that accepts one or more file name as arguments and converts all of them to uppercase, provided they exist in the current directory.

Solution:

```
# Get filename
echo -n "Enter File Name: "
read fileName
# Make sure file exits for reading
if [! -f $fileName]
then
echo "Filename $fileName does not exist"
exit 1
fi
# Convert uppercase to lowercase using tr command
```

```
tr '[A-Z]' '[a-z]' < $fileName
```

Experiment no. 10

Aim: Write a shell script that computes the gross salary of an employee according to the following rules:

1) If basic salary is < 1500 then HRA=10% of the basic and DA=90% of the basic

2) If the basic salary is >= 1500 then HRA=500/- and DA=98% of the basic

Solution:

```
echo "Enter the basic salary:"
read bsal
if [$bsal -lt 1500]
then
gsal=$((bsal+((bsal/100) * 10) + (bsal/100) * 90))
echo "The gross salary: $gsal"
fi
if [$bsal -ge 1500]
then
gsal=$(((bsal+500) + (bsal/100) * 98))
echo "The gross salary: $gsal"
fi
```

Experiment no. 11

Aim: Write an interactive file –handling shell program. Let it offer the user the choice of copying removing, renaming, or linking files. Once the user has made a choice, have the same program ask the user for the necessary information, such as the file name, new name and so on.

Solution:

Solution:

```
echo "Menu"
echo "1. Copy a File "
echo "2. Remove a file "
echo "3. Move a file"
echo "4. Quit"
echo "Enter your choice: \c"
read Choice
case "$Choice" in
1) echo "Enter file name to copy: \c"
read f1
echo "Enter file name: \c "
read f2
if [-f $f1]
then
cp $f1 $f2
```

```
else
echo "$f1 does not exist"
fi
;;

2) echo "Enter file to be removed:"
read r1
if [-f $r1]
then
rm -i $r1
else
echo "$r1 file does not exist "
fi
;;
3) echo "Enter file name to move: \c"
read f1
echo "Enter destination: \c "
read f2
if [-f $f1]
then
if [-d $f2]
then
mv $f1 $f2
```

```
fi
else
echo "$f1 does not exist"
fi
;;
4) echo "Exit......."
exit;;
esa
```

Experiment no. 12

Aim: Write a shell script that accepts any number of arguments and prints them in the reverse order**.**

Solution:

```
a=$#
echo "Number of arguments:" $a
x=$*
c=$a
res=''
while [1 -le $c]
do
c=`expr $c - 1`
shift $c
res=$res' '$1
set $x
```

```
done
echo "Arguments in reverse order:" $res
```

Experiment no. 13

Aim: Write a shell script to count the number of lines in a file that do not contain vowels**.**

Solution:

```
# Check to see if 1 argument is passed
if ([$# -ne 1])
then
echo "Usage: ./novowel.sh <file_name>"
exit 1
fi

awk 'BEGIN {line_count = 0}
# Initialize line_count
!/[io]/ { line_count++; print }
# Count the lines without i or o
END { printf "Number of lines which do not
contain vowels i or o = %d\n", line_count }
# Display the statistics
' $1
#awk '$0 !~ /[io]/ { print }' $1
```

EXPERIMENT NO: 14

AIM: Write a shell script which receives two file names as arguments. It should check whether the two file contents are the same or not. If they are the same then the second file should be deleted.

Solution:

```
echo \"Enter first file name \"
read file1
echo \"Enter second file name \"
read file2

cmp $file1 $file2 > error

total=\`wc -c error | cut -f 7 -d \" \"\`
echo $total

if [ $total -eq 0 ]
then
	echo \"Both file\'s contents are same\"
else
	echo \"Both file\'s contents are not same\"
fi
```

EXPERIMENT NO: 15

AIM: Develop an interactive script that ask for a word and a file name and then tells how many times that word occurred in the file.

Program:

cat f1.txt

day by day week by end

week by week month by end

month by month year by end

but friendship is never end

```
vi wcount.sh
echo " Enter the word to be searched"
read word
echo "Enter the filename to be used"
read flname
echo "the no. of times the word ['$word'] occured in the file."
grep -o $word $flname|wc -l
echo " the no of line that contains ['$word'] is"
grep -c $word $flname
```

Output:

```
sh wcount.sh
Enter the word to be searched
by
Enter the filename to be used
f1.txt
the no. of times the word ['by'] occured in the file.
6
the no of line that contains ['by'] is
3
```

EXPERIMENT NO: 16

AIM: Write a shell script to perform the following string operations:

i. To extract a substring from a given string.

ii. To find the length of a given string.

Program

```
$
vi substr.sh
echo "enter the string"
read str
strlen=${#str}
echo "The length of the given string '$str' is:$strlen "
echo "enter the string possibion in main str"
read s1
echo "ending possition"
read f1
echo $str | cut -c$s1-$f1
$
```

Output:

```
enter the string
khushi
The length of the given string 'krishnasai' is:6
enter the string possibion in main str
4
```

ending position

6

shi

$

EXPERIMENT NO: 17

AIM: Write a shell script that accepts two integers as its arguments and computes the value of first number raised to the power of the second number.

Program

```
if [ $# -ne 2 ]
then
echo "Invalid number of arguments"
exit
fi
pwr=`echo $1^$2 |bc`
echo "$1 raised to $2 is : $pwr"
```

EXPERIMENT NO: 18

AIM: Write a shell script that takes a command –line argument and reports on whether it is directory, a file, or something else.

Program

$ vi filetype.sh

echo "Enter the file name: "

read file

if [-f $file]

```
then
echo $file "---> It is a ORDINARY FILE."
elif [ -d $file ]
then
echo $file "---> It is a DIRCTORY."
else
echo $file "---> It is something else."
Fi
```

EXPERIMENT NO: 19

AIM: Develop an interactive grep script that asks for a word and a file name and then tells how many lines contain that word.

Program:

```
$ cat f1.txt
day by day week by end
week by week month by end
month by month year by end
but friendship is never end

vi wcount.sh
echo " Enter the word to be searched"
read word
echo "Enter the filename to be used"
```

```
read flname
echo "the no. of times the word ['$word'] occured in the file."
grep -o $word $flname|wc -l
echo " the no of line that contains ['$word'] is"
grep -c $word $flname
```

Output

```
sh wcount.sh
Enter the word to be searched
by
Enter the filename to be used
f1.txt
the no. of times the word ['by'] occured in the file.
6
the no of line that contains ['by'] is
3
```

EXPERIMENT NO: 20

AIM: Write a shell script to list all of the directory files in a directory.

Program:

```
# !/bin/bash
echo "enter directory name"
read dir
```

```
if[ -d $dir]

then

echo "list of files in the directory"

ls –l $dir|egrep ‘^d’

else

echo "enter proper directory name"

fi
```

Output:

```
guest-glcbIs@ubuntu:~$sh lprg6.sh

enter directory name

dir1

list of files in the directory

drwxrwxr-x 4 guest-glcbls guest-glcbls 140 2012-07-06 14:40 dir1
```

EXPERIMENT NO: 21

AIM: Write and execute all process control commands.

Solution:

Process control commands in Unix are:

bg - put suspended process into background

Syntax :

bg [job]

fg - bring process into foreground

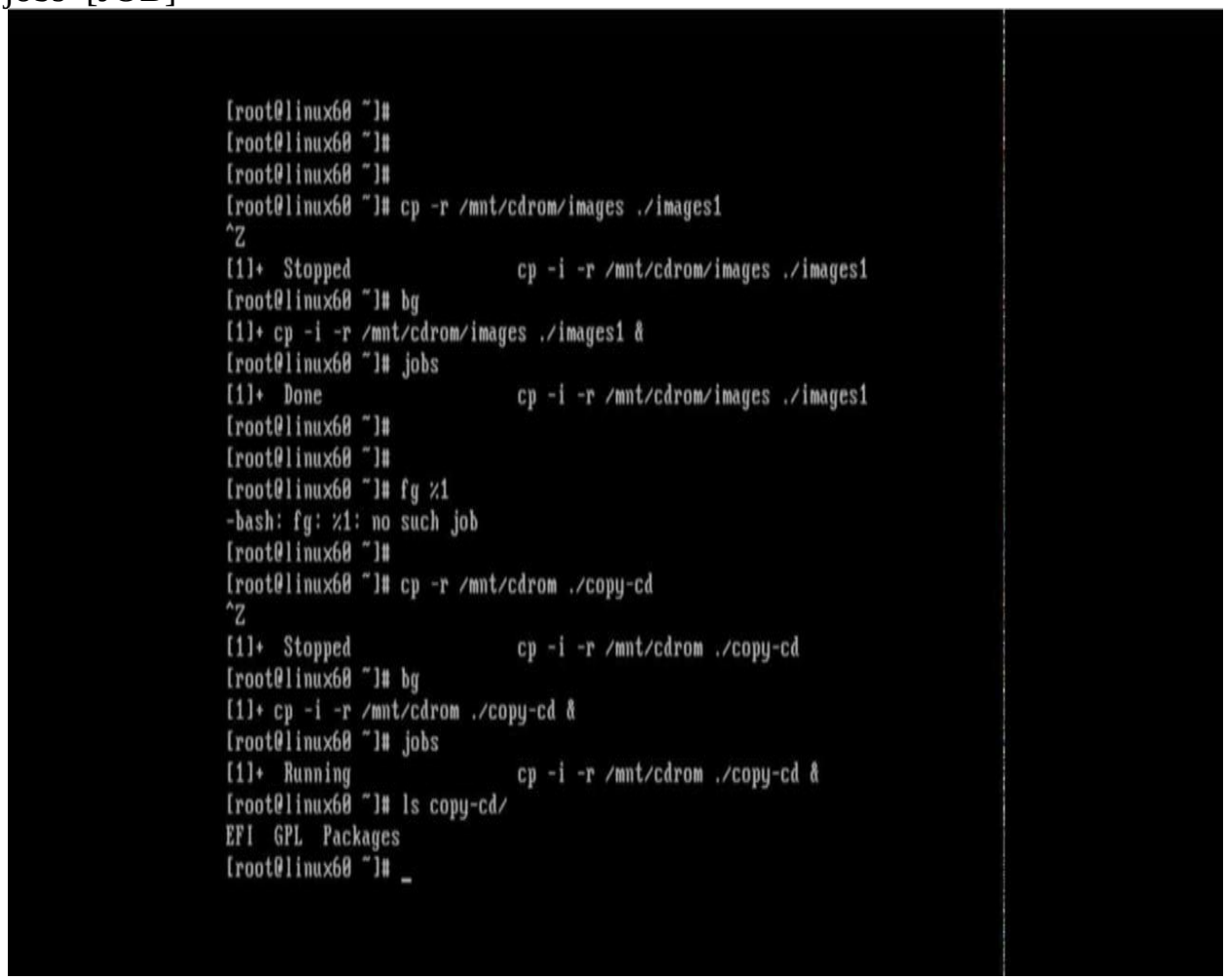
Syntax :

fg [%job]

jobs - list processes

Syntax :

jobs [JOB]



EXPERIMENT NO: 22

Aim:- Study & Installation of SAMBA, APACHE, TOMCAT.

SAMBA:- A Samba file server enables file sharing across different operating systems over a network. It lets you access your desktop files from a laptop and share files with Windows and macOS users.

For Ubuntu 16.04 LTS To install Samba, we run:

sudo apt update
sudo apt install samba
We can check if the installation was successful by running:

whereis samba

The following should be its output:

samba: /usr/sbin/samba /usr/lib/samba /etc/samba /usr/share/samba /usr/share/man/ma n7/samba.7.gz /usr/share/man/man8/samba.8.gz

APACHE:- Apache Web Server is a software package that turns a computer into an HTTP server. That is, it sends web pages – stored as HTML files – to people on the internet who request them. It is open-source software, which means it can be used and modified freely.

Open a terminal and type:

```
sudo apt-get update
```

Let the package manager finish updating.

Step 1: Install Apache

To install the Apache package on Ubuntu, use the command:

```
sudo apt-get install apache2
```

The system prompts for confirmation – do so, and allow the system to complete the installation.

```
dejan@dejan-VirtualBox:~$ sudo apt-get install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  bridge-utils ubuntu-fan
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sql
  libaprutil1-ldap
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap
0 upgraded, 8 newly installed, 0 to remove and 257 not upgraded.
Need to get 1,604 kB of archives.
After this operation, 6,493 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

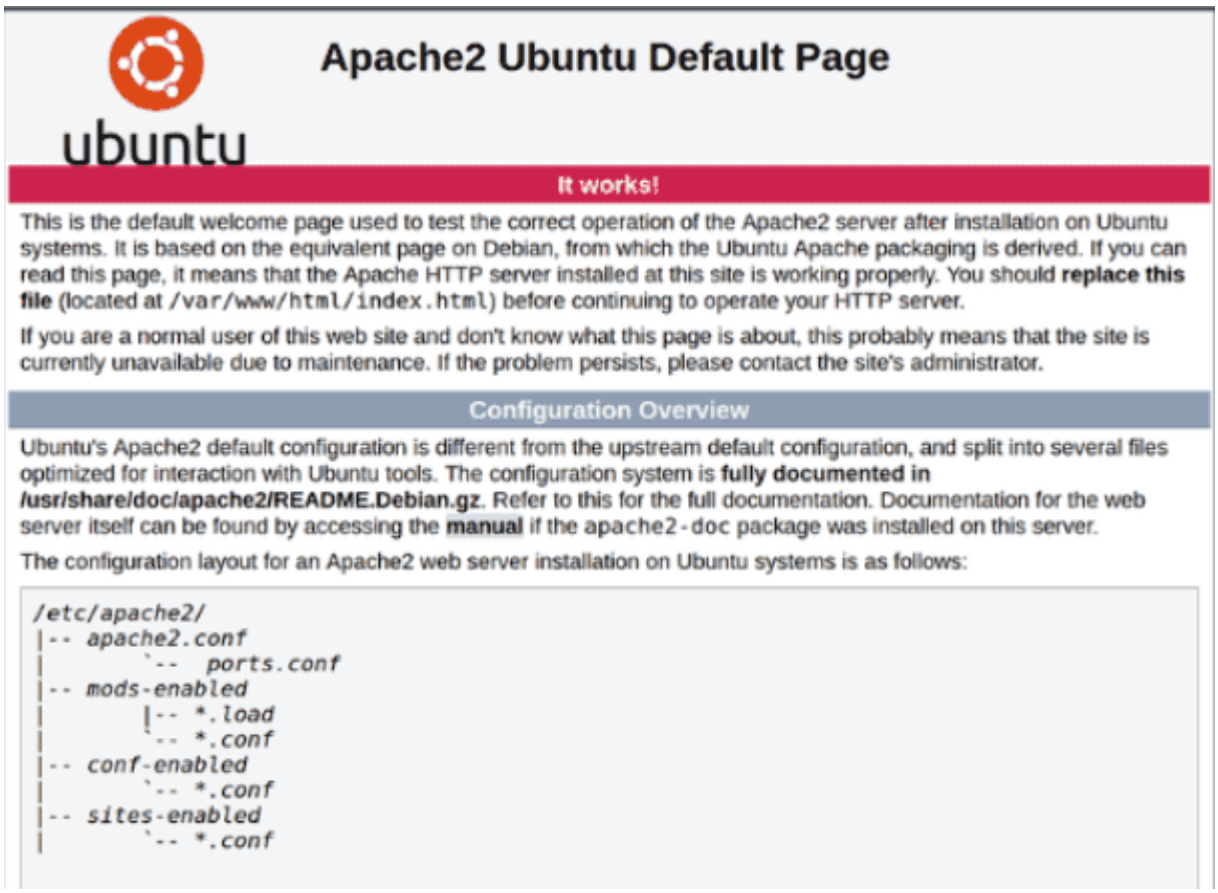
Step 2: Verify Apache Installation

To verify Apache was installed correctly, open a web browser and type in the address bar:

```
http://local.server.ip
```

The web browser should open a page labeled “Apache2 Ubuntu Default Page,” as in the image below

:



EXPERIMENT NO: 23

AIM: Study & installation of Firewall & Proxy server.

Firewall &- A firewall is a network security device, either hardware or software-based, which monitors all incoming and outgoing traffic and based on a defined set of security rules it accepts, rejects or drops that specific traffic.

Accept : allow the traffic
Reject : block the traffic but reply with an “unreachable error”
Drop : block the traffic with no reply

$ sudo systemctl start firewalld #start the service for the mean time

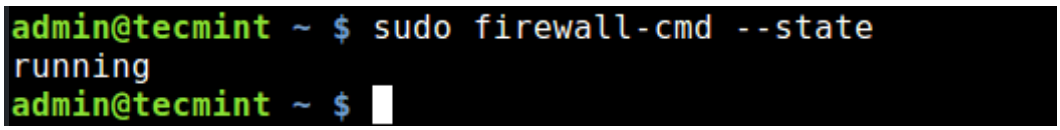
$ sudo systemctl enable firewalld #enable the service to auto-start at boot time

$ sudo systemctl status firewalld #view service status

After starting firewalld service, you can also check whether the daemon is running or not, using the firewall-cmd tool (in case it's not active, this command will output "not running").

$ sudo firewall-cmd --state

Check Firewalld Status



If you happen to save any changes permanently, you can reload firewalld. This will reload firewall rules and keep state information. The current permanent configuration will become new runtime configuration.

$ sudo firewall-cmd –reload

Proxy server:- A proxy server is a server that physically sits between your computer and the internet. ... The proxy server acts as the middleman and web page doesn't see the request coming from your computer. You can say proxy servers give the user who is requesting the web page anonymity and security.: Setting Up Temporary Proxy for a Single User

A temporary proxy connection resets after a system reboot. To establish such a connection for the current user, use the export command.

The syntax for establishing a temporary proxy connection is:

export HTTP_PROXY=[username]:[password]@[proxy-web-or-IP-address]:[port-number]

export HTTPS_PROXY=[username]:[password]@[proxy-web-or-IP-address]:[port-number]

export FTP_PROXY=[username]:[password]@ [proxy-web-or-IP-address]:[port-number]

...

export NO_PROXY=localhost,127.0.0.1,::1

Provide the proxy address (web or IP), followed by the port number. If the proxy server requires authentication, add your proxy username and password as the initial values.

This is what the set of commands should look like in terminal