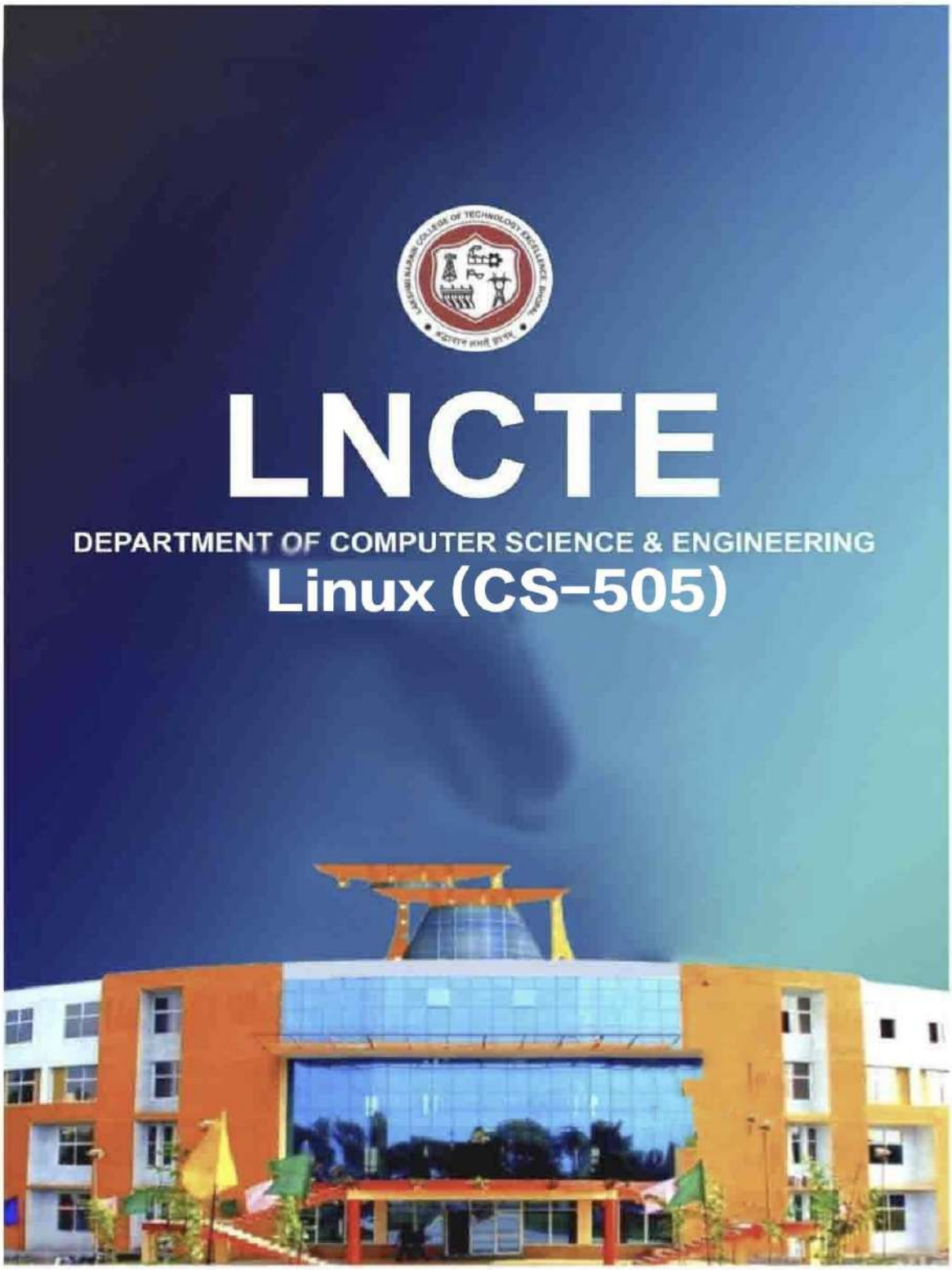

LNCTE
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Linux (CS-505)

Lakshmi Narain College of Technology Excellence

Department of Computer Science and Engineering

Lab Manual

Subject Name: Linux

Course Code: CS 505

Course: B. Tech

Session: 2023-24

Prepared By

TABLE OF CONTENT

Vision and Mission of the Institute

Vision of the institute

To become a pioneer institute in technical education and innovations to build competent technocrats and leaders for the nation.

Mission of the institute

M1. To enhance the academic environment with innovative teaching learning processes and modern tools.

M2. To Practice and nurture high standards of human values, transparency, and accountability.

M3. To collaborate with other academic and research institutes as well as industries to strengthen education and research.

M4. To uphold skill development for employability and entrepreneurship for interdisciplinary research and innovations.

Vision and Mission of the Department

Vision of the Department

To be a centre of excellence for providing quality technical education to develop future leaders with the aspects of research & computing, Software product development and entrepreneurship.

Mission of the Department

Mission No.	Mission Statements
M1	To offer academic program with state of art curriculum having flexibility for accommodating the latest developments in the areas of computer science and engineering
M2	To conduct research and development activities in contemporary and emerging areas of computer science & engineering.
M3	To inculcate moral values & entrepreneurial skills to produce professionals capable of providing socially relevant and sustainable solutions.

COURSE OUTCOMES: CS 505 – LINUX

Students should be able to

CO505.1	Implement installation process for Unix/Linux.
CO505.2	Execute basic commands of Linux OS.
CO505.3	Analyse process states, process scheduling and scheduling priorities.
CO505.4	Implement file creation, file modification and file access permissions
CO505.5	Execute basic Shell Programming assignments

Course Articulation Matrix

CO/PO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CO505.1	3	2	2	1	2	2	-	3	-	-	1	3
CO505.2	3	2	2	1	2	2	-	3	-	-	1	3
CO505.3	3	2	2	1	-	2	-	3	-	-	1	3
CO505.4	3	2	2	1	2	-	-	3	-	-	1	3
CO505.5	3	2	2	1	2	2	-	3	-	-	1	3
	3	2	2	1	2	2	-	3	-	-	1	3

Program Outcomes as defined by NBA (PO)

Engineering Graduates will be able to:

1.Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2.Problem analysis: Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

1. 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.

2. 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.

3. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

4. 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. **7. 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.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. 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.

11. 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.

12. 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.

Program Specific Outcomes (PSO)

A graduate of computer science and Engineering program will develop.

PSO 1: An ability to apply technical Knowledge of computer science and engineering. Fundamentals to become employable in industry.

PSO 2: An ability to develop programming skills using modern software tools and techniques.

PSO 3: An ability to develop real time projects for problem solving of domains such as Machine learning, Cyber security, Block chain And Big data.

PSO 4: An ability to grab research, higher studies, and entrepreneurship opportunities. Towards society with moral values and ethics.

Program Educational Objectives (PEO):

PEO-1: Evolve as globally competent computer professionals, researchers and entrepreneurs possessing collaborative and leadership skills, for developing innovative solutions in multidisciplinary domains.

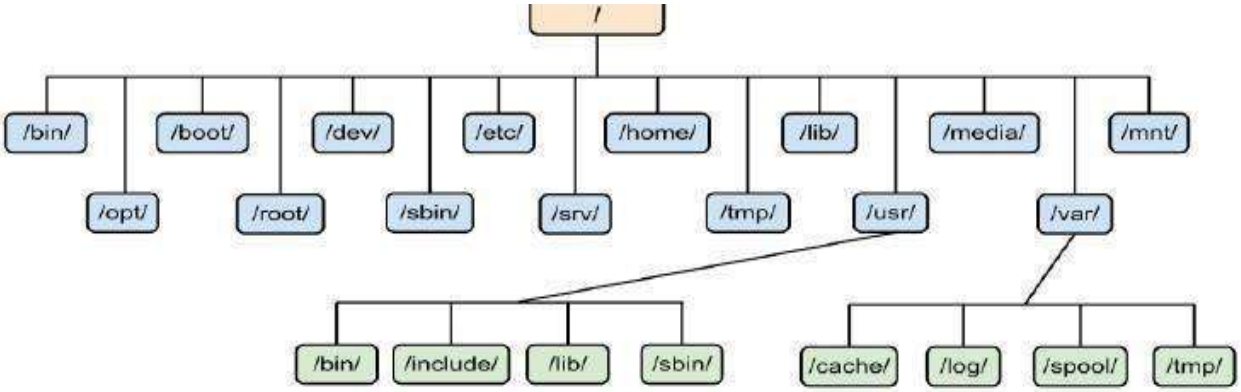
PEO-2: Excel as socially committed computer engineers having mutual respect, effective communication skills, high ethical values, and empathy for the needs of society.

PEO-3: involve in lifelong learning to faster the sustainable development in the emerging areas of technology.

List of Program to be performed: -

Sr. No.	List of Experiments	Associated CO
Lab-1	History of Unix/Linux	CO1
Lab-2	Explain all the Linux basic command	CO1
Lab-3	a) Write a shell script that receives any number of file names as arguments checks if every argument supplied is a file or directory and reports accordingly. whenever the arguments a file it reports no of lines present in it b) Write a shell script that accepts a list of file names as its arguments, counts and reports the occurrence of each word that is present in the first argument file on other argument files.	CO1
Lab-4	a) write a program to add two numbers b) Write a program for shell scripting to calculate simple interest	CO2
Lab-5	a) Study and use of commands for performing arithmetic operation with Unix/Linux b) write an awk script to count number of lines in a file that does not contain vowels c)write an awk script to find the no. of characters, words, and lines in a file	C02
Lab-6	a) Write a program to find the smallest of the three numbers that one read from keyword b) write a program to check whether number is prime or not.	CO2
Lab-7	a) Write a program to find the factorial of a number that is read from a keyboard b) Write a program to find whether a string is palindrome or not	CO2
Lab-8	Given two files each of which contains names of students. Create a program to display only those names that are found on both the files.	CO3
Lab-9	Create a program to find out the node number of any desired file.	CO3

Lab-10	Study & use of the Command for changing file permissions.	CO4
Lab-11	Execute shell commands through vi editor.	CO4
Lab-12	Write a shell script that accepts any number of arguments and prints them in the reverse order.	CO5
Lab-13	Write client server programs using c for interaction between server and client process using Unix Domain sockets.	CO5
Lab-14	Write a shell script that takes a command line argument and reports on whether it is directory, a file, or something else.	CO5



System Administration Tools

1. UNIX comes with its own tools such as SAM on HP-UX.
2. Suse Linux comes with Yast
3. Redhat Linux comes with its own gui tools called redhat-config-*.

However, editing text config file and typing commands are most popular options for sys admin work under UNIX and Linux.

UNIX Operating System Names

A few popular names:

1. HP-UX
2. IBM AIX
3. Sun Solairs
4. Mac OS X
5. IRIX

Linux Distribution Names

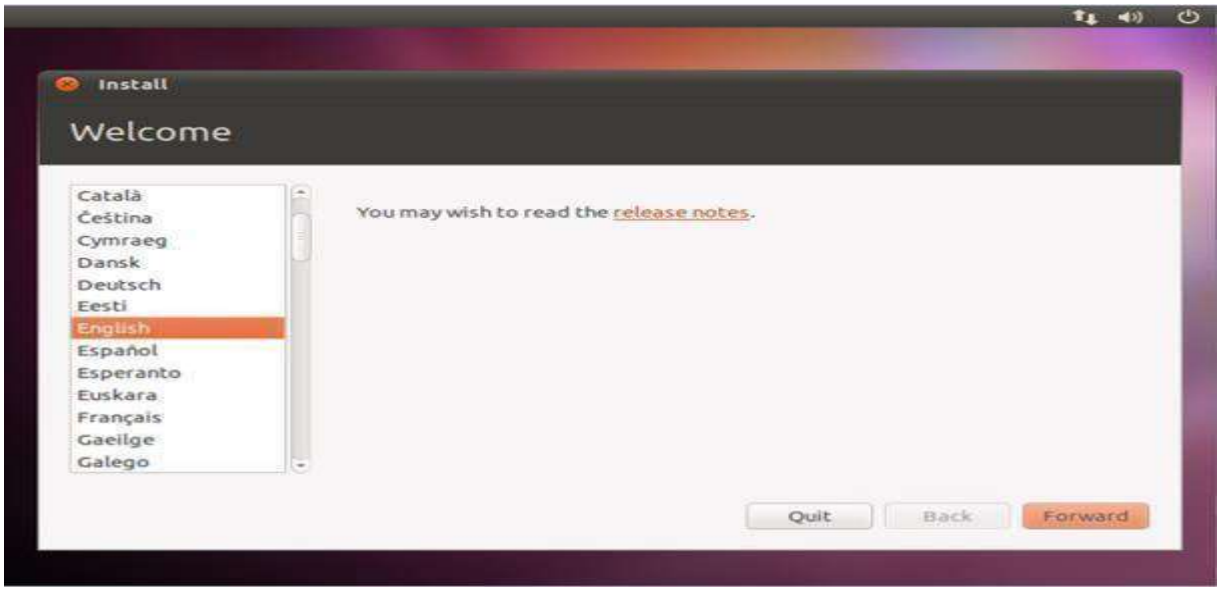
A few popular names:

1. Redhat Enterprise Linux
2. Fedora Linux
3. Debian Linux
4. Suse Enterprise Linux
5. Ubuntu Linux

Install Ubuntu Linux – Complete Step by Step

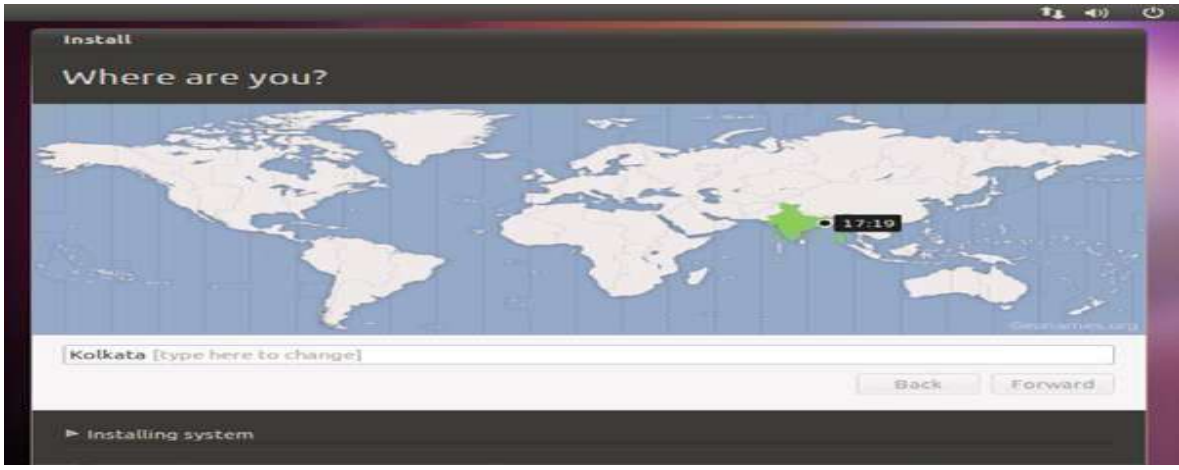
Step 1: Insert the ubuntu cd in the cd drive and boot the computer from cd. First, you will be prompted to select language. elect English or other language according to your preferences.

Step 2: Now you will see ubuntu menu, you can choose Try ubuntu without installing option to try ubuntu without installing it on your hard drive. For installing ubuntu choose the second option Install Ubuntu.



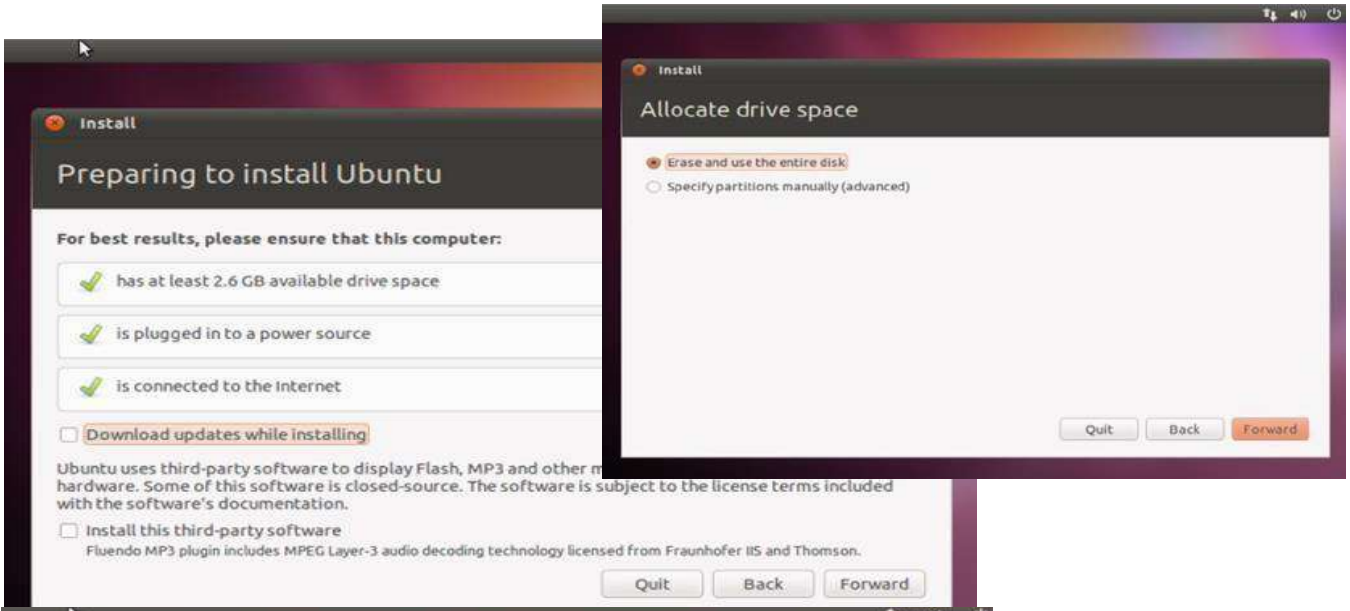
Step 3: Ubuntu will start now initializing and after few minutes you can see the installation wizard.

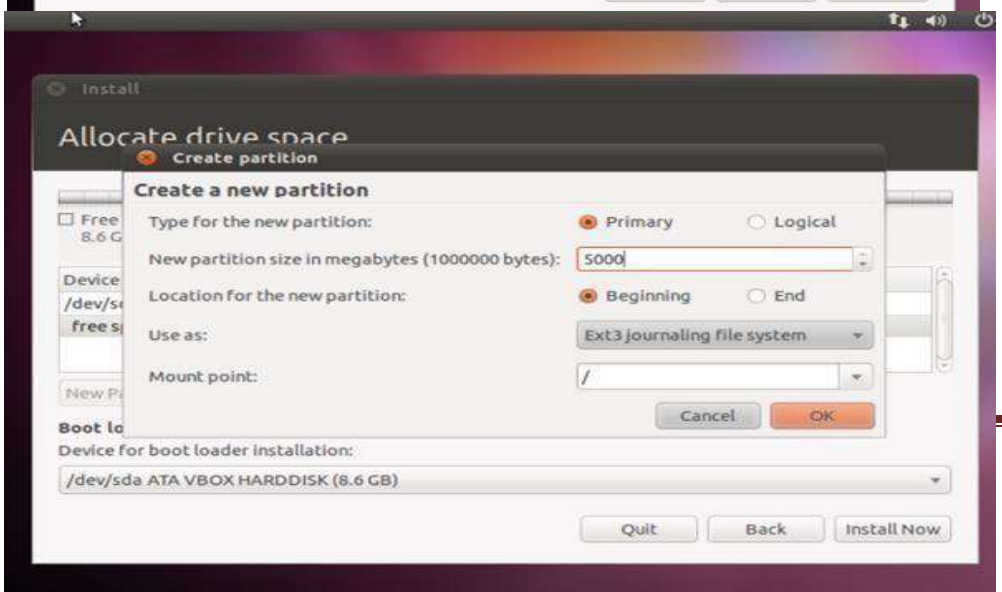
Step 4:Click



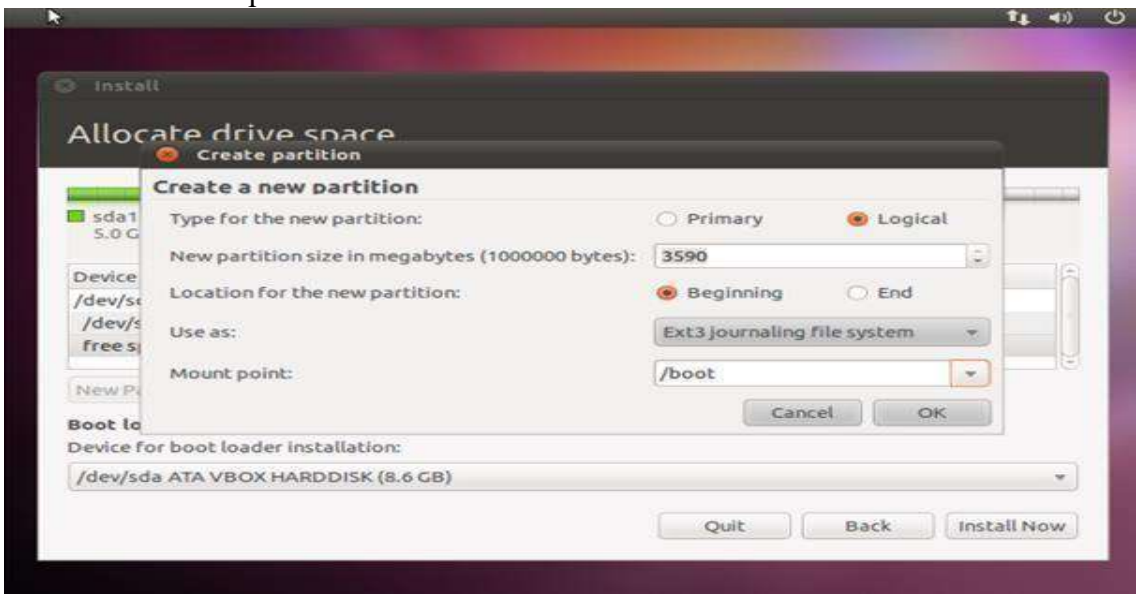
Forward and it will check the minimum requirements for running ubuntu on your PC. If everything is fine you can see green coloured tick marks. You can also select to download updates while installing and install some third-party software. After selecting the things, you want to click forward.

Step 5: Now you can choose either erase and use entire disk option or specify partitions manually option. You can choose the 1st option if you just want Linux to exist in your system. Else select second option. Now it will display the free space available for your pc.

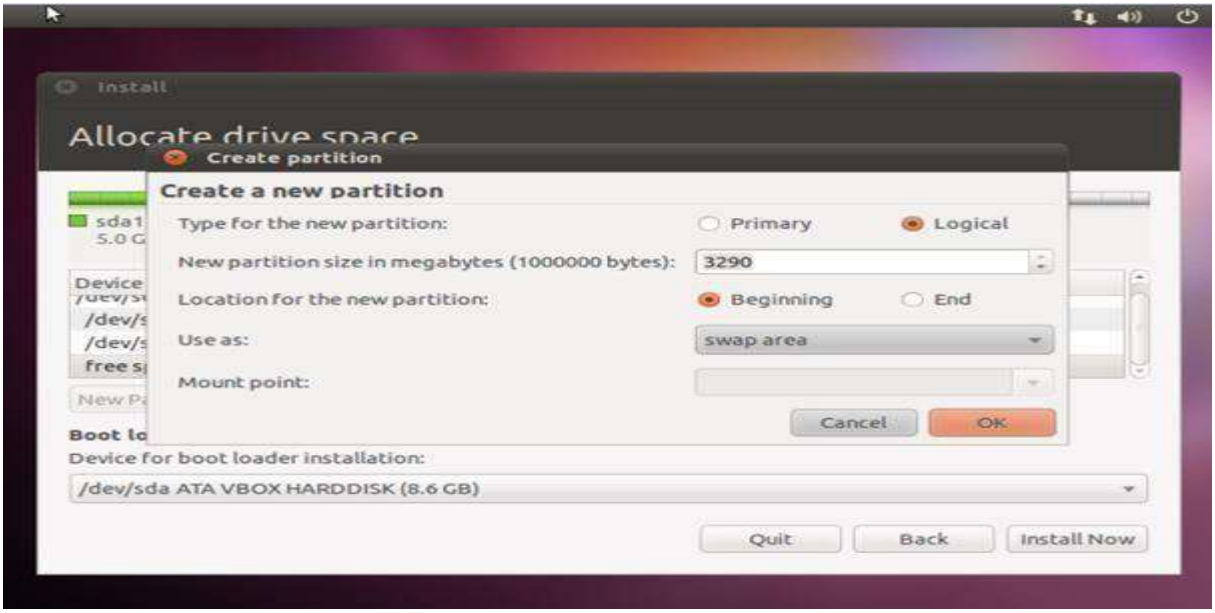


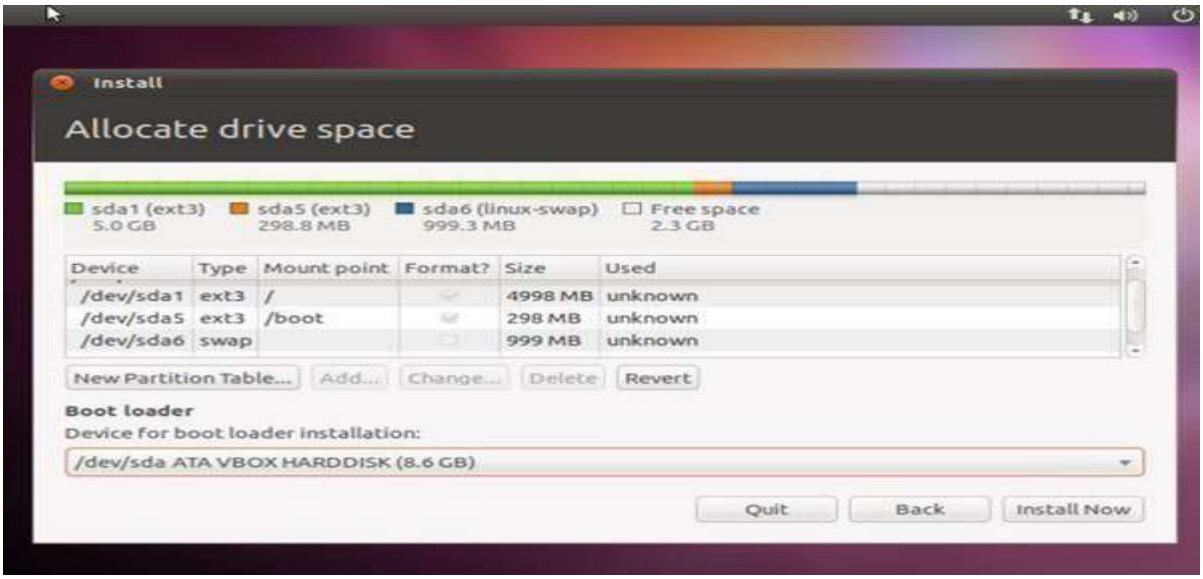


Now again select free space from the table and click add option. Now select size to be around 300mb, use as ext3 journaling file system and select mount point as /boot. Now again select free space from the table and click add option.



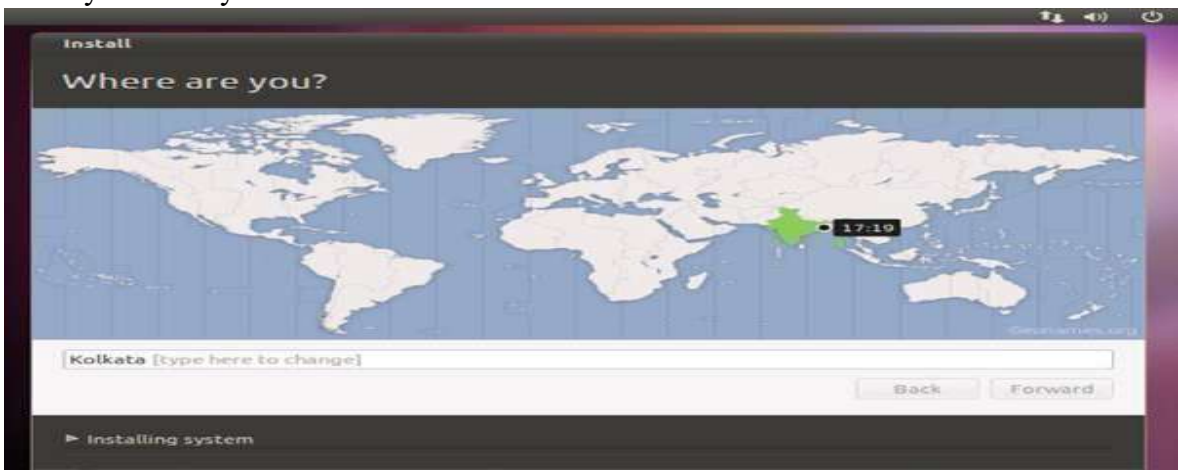
Now select size to be around twice the size of your ram that is around 1000 mb if your ram size is 512mb and select use as swap area and click ok.

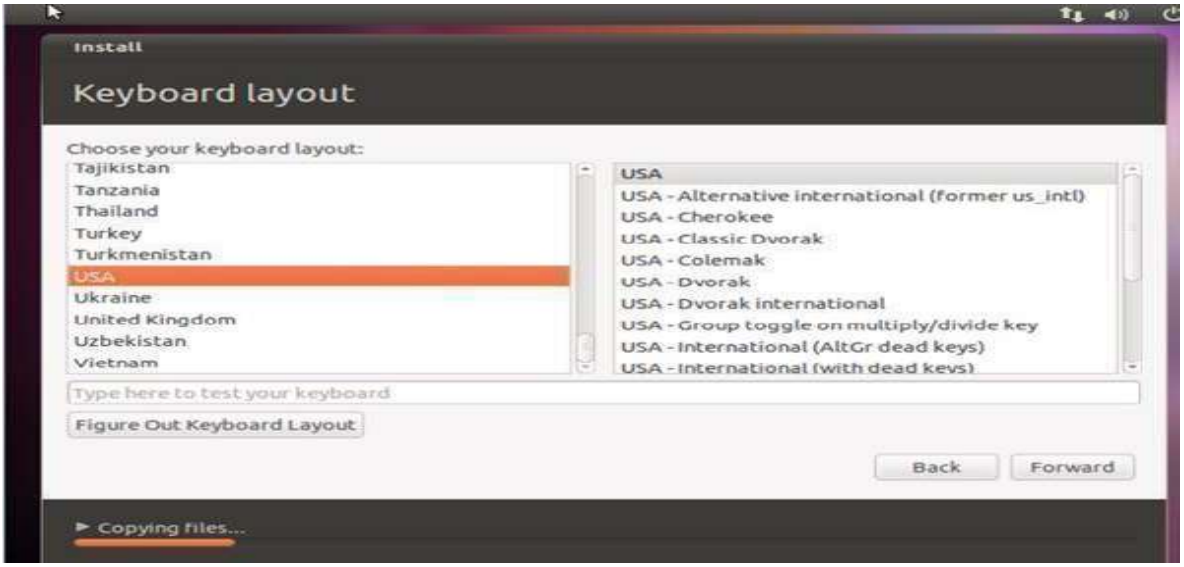




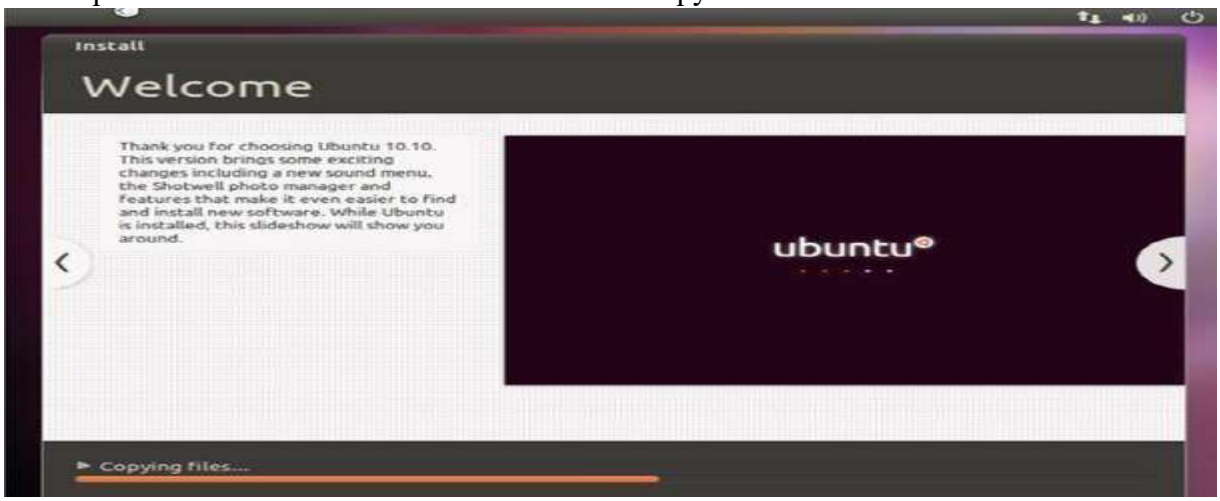
Step 6: Click Install now button and then the wizard will ask you location. Select your location and click forward.

Step 7: While you are selecting these options wizard will continue to copy files. Now select your desi red key boar d layout and click forward.

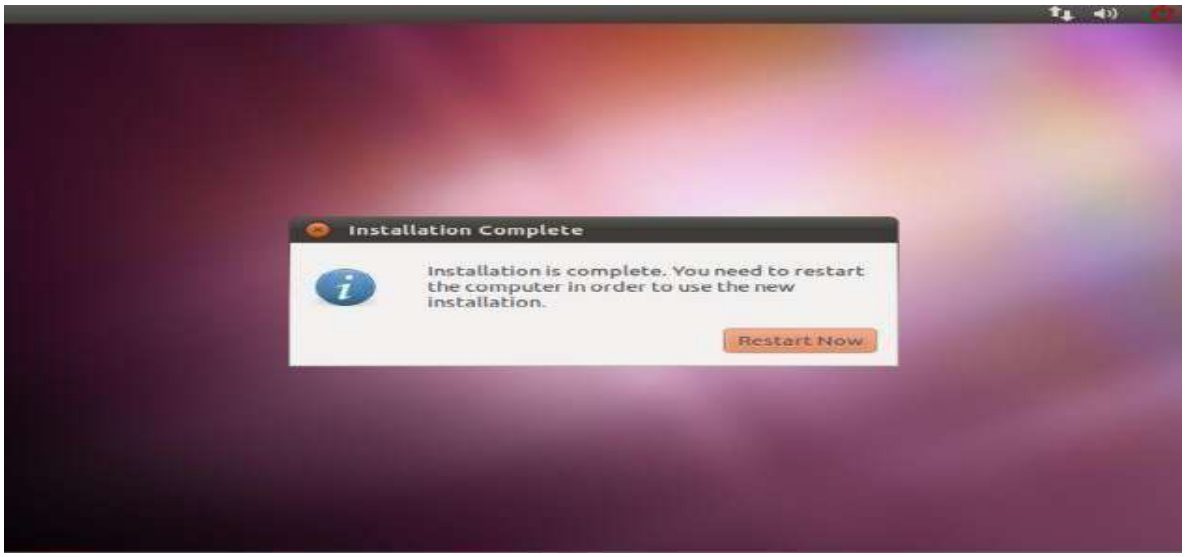




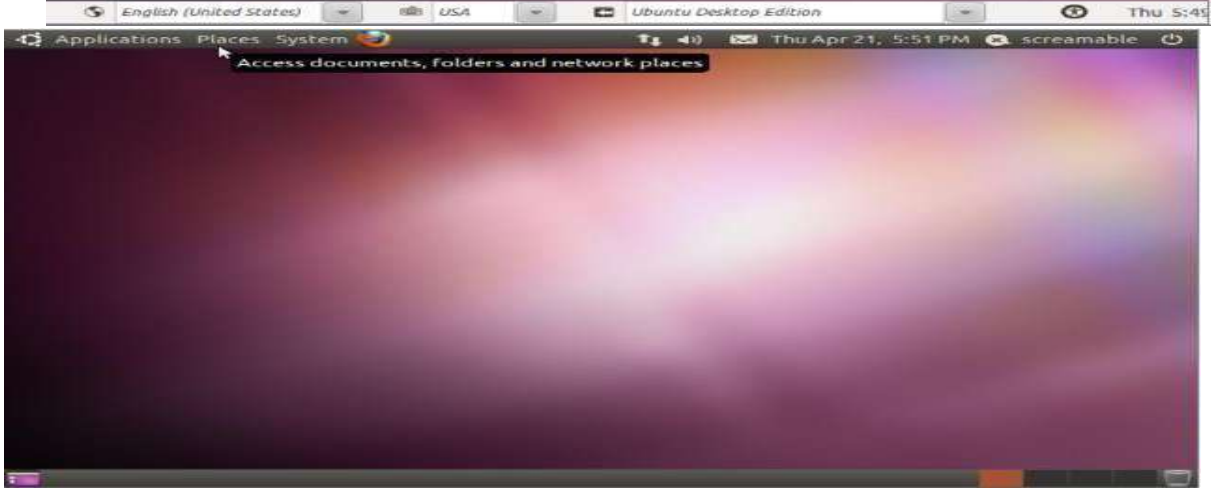
Step 8 : now fill in the details about yourself. Fill your name, computer name, choose a username and create a password and click forward and let ubuntu copy all the essential files.



Step 9 : After all files have been copied and installed ubuntu will display a message saying that installation complete and click on restart button to restart your computer. Remove the cd from the cd drive.



Step 10: After restarting your pc wait for the ubuntu to load and then it will display the login screen. Choose the user, enter password, and click login.



Experiment 1

History of Unix/Linux

History to Unix

Unix is a family of multiuser, multitasking computer OSes that derive from the actual AT&T Unix, whose integration started in 1969 at the Bell Labs research centre by Dennis Ritchie, Ken Thompson, and others. Unix was intended for use in the Bell system initially, leading to a range of both commercial and academic Unix versions from vendors, including IBM (AIX), HP/HPE (HP-UX), Sun Microsystems (Solaris/SunOS), Berkeley (BSD), Microsoft (Xenix), and the University of California.

Unix systems are designated by a modular design sometimes known as the *"Unix philosophy"*. The operating system should give a group of simple tools according to this philosophy, all of which perform a well-defined and limited function. An incode, unified-based file system and an inter-process communication technique called *"pipes"* serve as the primary communication means, and a command and shell scripting language is used to merge the tools to implement complex workflows.

Unix differentiates itself from its predecessors as the initial portable OS; almost the whole operating system is specified in the C programming language, which permits Unix to run on numerous platforms.

Components of Unix

The Unix system consists of several components that were actually packaged together. By adding the development environment, documents, libraries, and the modified, portable source code for each of these components, in inclusion to the kernel of an OS, Unix was an autonomous software system.

The filesystem locations and names of the Unix components have substantially changed across the system history. Nonetheless, the implementation of the 7 version is considered by several for having the early structure of Canonical:

- Kernel: It is composed of many sub-components, and its source code resides in the /us/sys directory.
- Development environment: The recent versions of Unix included a development environment acceptable for remaking the whole system from the source code.
- Commands: Unix makes a distinction between user-level programs or commands for system maintenance and operation, general utility commands, and more general-purpose software like typesetting packages and text formatting.
- Documentation: Unix was one of the initial OSes to add each of its documentation online inside the machine-readable format.

Introduction to Linux

Linux is a family of Unix-like open-source operating systems. Typically, Linux is packaged as the Linux distribution, which contains the supporting system libraries and software and the Kernel, several of which

are offered by the GNU Project. Several Linux distributions utilize the word "Linux", but the Free Software Foundation utilizes the "GNU/Linux" name to focus on the GNU software importance.

Famous Linux distributions are Ubuntu, Fedora Linux, and Debian, the latter of which itself composed of several different distributions and changes, including Xubuntu and Lubuntu. Commercial distributions are SUSE Linux Enterprise and Red Hat Enterprise Linux. Desktop Linux distributions are windowing systems like Wayland or X11 and desktop environments like KDE Plasma or GNOME.

Linux is one of the most outstanding examples of open-source and free software collaboration. Linux source code may be distributed, modified, and used non-commercially or commercially by anyone upon the terms of its respective licenses, like the GPL (GNU General Public License). For example, the Linux Kernel is licensed upon the GPLv2.

Components of Linux

Installed components of the Linux System contain the following:

- Bootloader: It is a program that can load the Linux Kernel into the main memory of the computer by being run by the system after the initialization of the firmware is performed and when it's turned on.
- Init program: It is the initial process begun by the Linux Kernel.
  Software libraries: These include code that can be used by active processes.
- Basic Unix commands: Basic Unix commands, along with GNU choreutids, are the typical implementation. Several alternatives are available for embedded systems, like BSD-licensed Toybox and the copyleft Busy Box.
- Widget toolkits: They are the libraries utilized to create graphical user interfaces for software applications. Several widget toolkits are present, including Clutter and GTK integrated by the GNOME Project, Qt integrated by the Qt Project and conducted by the Enlightenment Foundation Libraries (EFL), and the Qt company primarily developed by the Enlightenment team.
- Package management system: The package management system includes RPM and dpkg. The packages can alternatively be compiled from source and binary tarballs.
- User interface program: The user interface programs are also available in Linux, such as windowing environments or command shells.

Experiment 2

A-Z Index of the command line for Linux

Awk command

he awk command is used for text processing in Linux. Although, the sad command is also used for text processing, but it has some limitations, so the awk command becomes a handy option for text processing. It provides powerful control to the data. syntax

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

The options can be:

-f program files**:** It reads the source code of the script written on the awk command -

F fs: It is used as the input field separator.

ls command

Option	Description
ls R	lists all the files in the sub-directories as well
ls S	sorts and lists all the contents in the specified directory by size
ls -al	list the files and directories with detailed information
ls -a	shows the hidden files in the specified directory

The ls command allows the user to view the contents of a directory. It lists the files and directories. The ls command displays the contents of the current working directory by default (if the user does not specify any other directory). To check the content of other directories, you can type the ls command followed by the directory path.

ls command syntax:
ls [Options] [File]
Some of the common option tags that you can use with the ls command:

clear

The clear command clears the terminal screen. This command ignores any command-line parameters and does not take any argument clear command syntax.

$ clear

pwd command

The full form of pwd command is Print Working Directory. It allows users to find the path of the current working directory or folder they are in. The pwd command has two options:

- -L: prints a symbolic path
- -P: prints the actual full path pwd command syntax

$ pwd **cd**

command

Use the cd command to navigate through the Linux files and directories. You will have to write the full path or the name of the directory to use this command.

For example, if you are working in /home/username/Documents and want to go in the Pictures subdirectory of the same directory, you can write cd Pictures.

Command	Description
Cd	to go to the home folder
cd..	to move one directory up
cd-	move to your previous directory

If you want to go to a new directory, you can write cd followed by the absolute path of the directory – cd /home/username/Music.

cp command

You can use the cp command to copy files from the current directory to a different directory.

cp source file destination file. In case you need a copy of the file second.txt in the same directory you have to use the cp command

cp command syntax
$ cp source file destination file
Example – to copy the contents of the red file into the blue file.
$ cp red.txt blue.txt

mv command

Use the mv command to move a file from a given directory to a different directory. It helps programmers to organize data easily. You can also use this command to rename files. The file or directory that is moved is deleted from the working directory.

mv command syntax – to move a file $
mv <Filename> <Directory Name>
mv command syntax – to rename a file
mv old_filename new_filename.
Also Read: Top Unix Interview Questions and Answers

rm command

The rm command deletes directories as well as the contents within them. This command needs to be used carefully as it deletes everything.

rm command syntax *$ rm <filename>* rmdir command rmdir allows users to delete a directory, provided that the directory is empty. You will need to ensure that there is no file or subdirectory under the directory that you want to delete.

$ rmdir <directoryname>

touch command

With the touch command, you can create a new blank file with the given name.

syntax
$ rmdir <directoryname>
You can also create multiple files simultaneously.

$ touch <filename1> <filename2>

locate command.

The locate command enables users to locate a file. If you don't remember the exact file name, you can use the -i argument to make it case-insensitive.

$ locate <filename>

sudo command.

sudo stands for 'SuperUser Do'. It enables users to run some commands as a super user or System Administrator that normal users cannot do. You can run such commands that need elevated rights on a Linux system.

syntax
sudo command_you_want_to_execute

df command

With the df command, users can get the information related to file systems about total space and available space. You can get a report on display the size, available space, and system's disk space usage, and more. This command shows the result in percentage and KBs.

df command syntax
df [File] df
command options

Option	Description
-x	excludes specific filesystems
-T, –print-type	prints the type of file system
-a, –all	includes duplicate and files that are inaccessible
-m	shows the result in megabytes

du command

The du (Disk Usage) command shows how much space a file or a directory takes.

syntax
$ du du command has the below
options:

Option	Description
-h	shows human-readable form
-s	gives a summary of the output total size

Example $ du -h

chown command

All files are owned by a specific user in Linux. With the chown command, you can change the ownership of a file or folder to the specified username.

chown owner_name file_name
Example – to make linuxuser1 the owner of the file.ext
chown owner_name file_name

echo command

The echo command displays a text or a string to the standard output or a file.

echo command syntax
$echo "<String>"
Example
$ echo "This is an example of echo"

head command

This command allows users to view the first lines of any text file. It shows the first ten lines by default.

syntax
$ head [Filename]
You can also manually input the number of lines you want to view.

Syntax
$ head -n <number> <Filename>

tail command

This command shows the last ten lines of a text file.

syntax
$ head [Filename]
The number of output lines is ten by default, however, this can be changed to any number with the -n (number) option.

Syntax
tail -n <number> <Filename>

uname command

uname stands for Unix Name. This command prints detailed information about your Linux system. The information includes machine name, kernel, operating system, and more. Different options tell about different pieces of information.

Option	Description
-a	this option displays everything
-v	shows the kernel information
-s	displays the kernel version of the system
-r	displays the kernel release

uname command syntax
uname [Option]
Example uname -s

history command

The history command shows the previously used commands. It displays the information about the commands executed by a user.

history command syntax
$ history

man command

This command is used to check the reference manual pages for commands or programs.

man command syntax $
man [Command Name]
Example $ man
head **useradd**
command

It is used to add or remove a user on a Linux server.

syntax:

useradd [options] username
Example: useradd
newperson1

Cat Command

The cat command is used to display the contents of a file.
Syntax of cat command: cat <filename.extension>
Example:
cat newfile.txt
A few other use of cat command in Linux:

Option	Function
cat > [fileName]	Create a new file.
cat [old_file] > [new_file]	Copy content from old file to new file.
cat [file1, file2,….] > [new file name]	Concatenate contents of multiple files into one file.
cat –n [File_Name] / cat -b [File_Name]	Display line numbers.
cat -e [fileName]	Display $ character at the end of each line.

Ping command

Utilize the ping command to check your connection to a server. It checks the reachability of a host on an Internet Protocol (IP) network".

Syntax of ping command:

ping <Domain Name>

Example:

ping google.com

Experiment 3

a) Write a shell script that receives any number of file names as arguments checks if every argument supplied is a file or directory and reports accordingly. whenever the arguments a file it reports no of lines present in it?

```
if [ $# -eq 0 ]
then
echo "no arguments"
else
tr " " "
" < $1 > temp
shift
for i in $* do tr " " " " < $i >
temp1 y=`wc -l < temp` j=1
while [ $j -le $y ] do
x=`head -n $j temp | tail -1`
c=`grep -c "$x" temp1`
echo $x $c j=`expr $j 1`
done done
fi
```

Output:

```
$sh 9a.sh hegde.sh ravi.sh
Raghu 2
Hary 1
```

Vinay 9

Excersice

1)Write a shell script to count no of regular files in the current workingdirectory ?
2)Write a shell script to display list of currently logged users?

Viva questions:

1. What is an internal command in Linux? Internal commands are also called shell built-in commands. Example: cd,fg. Since these are shell built-in, no process is created while executing these commands, and hence are considered to be much faster.
2. x and y are two variables containing numbers? How to add these 2 numbers? $ expr $x + $y
3. How to add a header record to a file in Linux? $ sed -i '1i HEADER' file
4. How to find the list of files modified in the last 30 mins in Linux? $ find . -mmin -30

b) Write a shell script that accepts a list of file names as its arguments, counts and reports the occurrence of each word that is present in the first argument file on other argument files.

ALGORITHM:

step1: Check the no of arguments for shell script if 0 arguments then print no arguments
step2:else translate each word in the first file is to be on separate line which will be stored in temp file
step3: for i in $* for every filename in given files
step 4: translate each word in the file is to be on separate line which will be stored in temp1 file
step5: count no of lines in temp file assign it to j step6: initialize j=1
step 7: while i< j extract the line that are common in both the file by using head and tail commands then apply the filter grep to count and print the lines which are common to files increment j step 8: stop Script name:5.sh

```
#!/bin/bash echo "no of
 arguments $#"
if [ $# -le 2 ] then echo "Error : Invalid number
of arguments." exit fi
str=`cat $1 | tr '\n' ' '`
for a in $str do echo
"in file $a"
        echo "Word = $a, Count = `grep -c "$a" $2`"
done
```

Execution and output: check data
in abc1.txt file [root@localhost
sh]# cat abc1.txt abc def ghi abc
abc cccc check data in abc1.txt
file [root@localhost sh]# cat
abc2.txt abc def ghi abc abc cccc
executing script
[root@localhost sh]# sh 5.sh abc1.txt abc2.txt
Word = abc, Count = 3
Word = def, Count = 1

Word = ghi, Count = 1
Word = abc, Count = 3
Word = abc, Count = 3
Word = cccc, Count = 1

Exercise

1)Write a shell script to print prime numbers ?
2)Write a shell script to print Fibonacci numbers?

Viva question

1. What is Shell Scripting ? Shell scripting, in Linux or Unix, is programming with the shell using which you can automate your tasks. A shell is the command interpreter which is the interface between the User and the kernel. A shell script allows you to submit a set of commands to the kernel in a batch. In addition, the shell itself is very powerful with many properties on its own, be it for string manipulation or some basic programming stuff.

2. The command "cat file" gives error message "--bash: cat: Command not found". Why? It is because the PATH variable is corrupt or not set appropriately. And hence the error because the cat command is not available in the directories present PATH variable.

3. How to find the length of a string inLinux? $ x="welcome" $ echo ${#x} 7

4. What are the different timestamps associated with a file? Modification time:- Refers to the time when the file is last modified. Access time :- The time when the file is last accessed. Changed time :The time when the attributes of the file are last changed?

5.diffrence between unix and linux?

Experiment4

a)write a program to add two numbers

```
!/bin/bash
#function to add two numbers
add() { x=$1 y=$2 echo -e "Number entered
by u are: $x and $y" echo "sum of $1 and $2
is `expr $x + $y` "
}
# main script echo "enter
first number" read first
echo "enter second number"
read sec #calling function
add $first $sec
```

```
echo "end of the script"
output:
Enter first number
89
Enter second number
45
Number entered by u are: 89 and 45
Sum of 89 and 45 is 134 end of the
script
```

b) Write a program for shell scripting to calculate simple interest?

```
echo " Enter the principle value: "
read p
echo " Enter the rate of interest:"
read r
echo " Enter the time period:"
read t s=`expr $p \* $t \* $r /
100` echo " The simple
interest is " echo $s output:
[redhat35@localhost Rhel5]$ sh si.sh
Enter the principle value: 2000
Enter the rate of interest:
4
Enter the time period:
10
The simple interest is
800
```

Experiment 5

a) Study and use of commands for performing airthmetic operation with unix/linux

```
a=10
b=20

val=`expr $a + $b`
echo "a + b : $val"

val=`expr $a - $b`
echo "a - b : $val"

val=`expr $a \* $b`
echo "a * b : $val"
```

```
val=`expr $b / $a` echo
"b / a : $val"

val=`expr $b % $a`
echo "b % a : $val"

if [ $a == $b ]
then echo "a is equal
  to b"
fi

if [ $a != $b ] then echo "a
is not equal to b" fi
```

output:

```
a + b : 30
a - b : -10
a * b : 200
b / a : 2 b
% a : 0
a is not equal to b
```

b) write an awk script to count number of lines in a file that does not contain vowels?

ALGORITHM
Step 1: create a file with 5-10 lines of data
Step 2: write an awk script by using grep command to filter the lines
that do not contain vowels awk ‘ $0 ~/aeiou/ {print $0}’ file1 step3:
count=count+1 step4:print count
step5:stop
Awk script name:nm.awk

```
BEGIN{}
{
 If($0 !~/[aeiou AEIOU]/)
wordcount+=NF
}
END
{
print "Number of Lines are", wordcount
}
```

input file for awk script:data.dat
bcdfghj abcdfghj bcdfghj

ebcdfghj
bcdfghj
ibcdfghj
bcdfghj
obcdfghj
bcdfghj
ubcdfghj
Executing the script:
[root@localhost awk] #
awk -f nm.awk data.dat
bcdfghj
bcdfghj
bcdfghj
bcdfghj
bcdfghj
Number f lines are 5

c) write an awk script to find the no of characters ,words and lines in a file?

```
#! /bin/bash

echo "Enter a String" #
Taking input from user
read text

# Counting words
word=$(echo -n "$text" | wc -w) #
Counting characters char=$(echo
-n "$text" | wc -c)

# Counting Number of white spaces (Here,specificly " ")
# sed "s/ change this to whitespace//g" space=$(expr length
"$text" - length `echo "$text" | sed "s/ //g"`)

# Counting special characters special=$(expr
length "${text//[^\~!@#$&*()]/}")

# Output echo "Number of Words = $word"
echo "Number of Characters = $char" echo
"Number of White Spaces = $space" echo
"Number of Special symbols = $special"
```

Experiment 6

a)Write a program to find the smallest of the three numbers that one read from keyword?

```
echo "Enter three numbers:"
```

```
read num1
read num2
read num3
smallest=$nu
m1

if [ $num2 -lt $smallest ]; then
   smallest=$num2
fi

if [ $num3 -lt $smallest ]; then
   smallest=$num3
fi

echo "The smallest number is: $smallest"
```

b)write a program to check whether number is prime or not

```
#!/bin/bash

echo "Enter a number:"
read number i=2

if [ $number -lt 2 ]
then echo "$number is not a prime
number." exit fi

while [ $i -lt $number ]
do
   if [ `expr $number % $i` -eq 0 ] then echo
   "$number is not a prime number." exit fi
   i=`expr $i + 1`
done

echo "$number is a prime number."
```

Experiment 7

a) write a program to find the factorial of a number that is read from a keybord?

```
echo "Enter a
number" read num
fact=1 while [ $num -
gt 1 ]
do
```

```
    fact=$((fact * num)) #fact = fact * num
    num=$((num - 1))     #num = num - 1
  done echo
  $fact
```

b)Write a program to find whether a string a per is palindrome or not?

```
is_palindrome () {
   local word=$1 local
   len=$((${#word} - 1))
   local i
   for ((i=0; i <= (len/2); i++)); do
      [[ ${word:i:1} == ${word:len-i:1} ]] || return 1
done return 0 }

for word in hello kayak; do
   if is_palindrome $word; then
      echo $word is a palindrome
   else
      echo $word is NOT a palindrome
   fi
done
```

Inspired by gniourf_gniourf:

```
is_palindrome() {
  (( ${#1} <= 1 )) && return 0 [[
  ${1:0:1} != ${1: -1} ]] && return 1
  is_palindrome ${1:1: 1}
}
```

Excersices

1Write a shell script to find sum of first n natural numbers

2 Write a shell script to find largest of given three numbers

Experiment 8

1)Given two files each of which contains names of students?

```
$ bash --version
GNU bash, version 3.2.51(1)-release
Copyright (C) 2007 Free Software Foundation, Inc.
```

```
$ cat > abc
123
567
132
$ cat > def
132
777
321
```

So the files abc and def have one line in common, the one with "132". Using comm on unsorted files:

```
$ comm abc def
123
    132
567
132
    777
    321
$ comm -12 abc def # No output! The common line is not found
$
```

The last line produced no output, the common line was not discovered.

Now use **comm** on sorted files, sorting the files with process substitution:

```
$ comm <( sort abc ) <( sort def )
123
        132
    321
567
    777
$ comm -12 <( sort abc ) <( sort def )
132
```

2)Create a program to display only those names that are found on both the files?

// displaying contents of file1 //

```
$cat file1.txt
Apaar
Ayush Rajput
Deepak
```

Hemant

// displaying contents of file2 //
$cat file2.txt
Apaar
Hemant
Lucky
Pranjal Thakral
// using comm command for comparing

two files //

$comm file1.txt file2.txt Apaar
Ayush Rajput
Deepak Hemant
Lucky
Pranjal Thakral

Excersice

1Write an awk script to find square root of a given number ?
2 Write an awk script to find maximum of two numbers , read input from keyboard?

Experiment 9

Create a program to find out the inode number of any desired file?

An Inode number is a uniquely existing number for all the files in Linux and all Unix type systems.

When a file is created on a system, a file name and Inode number is assigned to it.
Generally, to access a file, a user uses the file name but internally file name is first mapped with respective Inode number stored in a table.

Note: Inode doesn't contain the file name. Reason for this is to maintain hard-links for the files. When all the other information is separated from the file name then only we can have various file names pointing to the same Inode.

node Contents

An Inode is a data structure containing metadata about the files.

Following contents are stored in the Inode from a file:

- User ID of file ○ Group ID of file ○ Device ID ○ File size ○ Date of creation ○ Permission ○ Owner of the file
  - File protection flag
- Link counter to determine number of hard links

The best way to get the inode number of a file in Linux is using the command ls -li filename. This command will give you a lot of information about the file. The inode number will be displayed on the first column of the output.

For example, if we have a file named test.txt, we would type: ls -li test.txt

which would give us the output:

1200 -rw-r–r– 1 root root 2049 Apr 18 2034 test.txt

The inode number for this file is “1200”.

The ls command is used to display the contents of a directory. By default, it will show the name for each file and directory under the directory. You can use various options with the ls command to change the output.
For example, if we want to see the inode numbers for all of the files in our current directory, we would type: ls -li

This would give us the output:

8398668 drwxrwxr-x. 14 postgres postgres 152 Sep 12 09:21 test
16800775 drwx——. 2 root root 6 Sep 16 07:12 testdir
25173208 drwx——. 2 root root 6 Sep 14 23:58 testdir2
As you can see, the inode number is displayed on the left side of the output.

Check file inode number with stat command in Linux
We can use stat command to get the inode number of a file in Linux. This command will give you a lot of information about the file such as file size, file permission, inode number etc.
Inode: 1200 Links: 13
The number after the word “Inode:” is the inode number. In the example above, it is “1200”.

The stat command has a number of options, including the -c option. This option allows you to output the inode number as a character.The stat command with -c option allows you to use a particular or custom format instead of the default.

we can use stat -c “%i” filename to get the inode number directly.

Experiment 10

Study & use of the Command for changing file permissions?

In Linux, there are three types of owners: user, group, and others .

Linux User

A user is the default owner and creator of the file. So this user is called owner as well.

Linux Group

A user-group is a collection of users. Users that belonging to a group will have the same Linux group permissions to access a file/ folder.

You can use groups to assign permissions in a bulk instead of assigning them individually. A user can belong to more than one group as well.

Other

Any users that are not part of the user or group classes belong to this class.

Linux File Permissions

File permissions fall in three categories: read, write, and execute.

Read permission

For regular files, read permissions allow users to open and read the file only. Users can't modify the file.

Similarly for directories, read permissions allow the listing of directory content without any modification in the directory.

Write permission

When files have write permissions, the user can modify (edit, delete) the file and save it.

For folders, write permissions enable a user to modify its contents (create, delete, and rename the files inside it), and modify the contents of files that the user has write permissions to.

Execute permission

For files, execute permissions allows the user to run an executable script. For directories, the user can access them, and access details about files in the directory.

Below is the symbolic representation of permissions to user, group, and others.

Symbolic representation of permissions

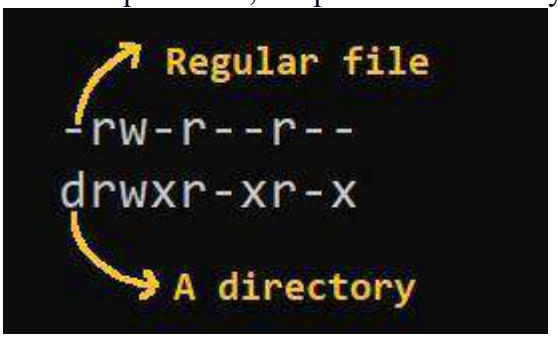
Note that we can find permissions of files and folders using long listing (ls -l) on a Linux terminal.

```
zaira@Zaira:~/freeCodeCamp$ ls -l
total 3856
-rw-r--r--    1 zaira zaira      89 Apr  5 20:46 CODE_OF_CONDUCT.md
-rw-r--r--    1 zaira zaira     210 Apr  5 20:46 CONTRIBUTING.md
-rw-r--r--    1 zaira zaira    1513 Apr  5 20:46 LICENSE.md
-rw-r--r--    1 zaira zaira   19933 Apr  5 20:46 README.md
drwxr-xr-x    4 zaira zaira    4096 Apr  6 22:45 api-server
-rw-r--r--    1 zaira zaira      67 Apr  5 20:46 babel.config.js
drwxr-xr-x   10 zaira zaira    4096 Apr  6 22:55 client
drwxr-xr-x    5 zaira zaira    4096 Apr  6 22:54 config
```

MODE OWNER GROUP SIZE MODIFICATION DATE FILE/FOLDER NAME

Output of long listing

In the output above, d represents a directory and- represents a regular file.



How to Change Permissions in Linux Using the chmod **Command**

Now that we know the basics of ownerships and permissions, let's see how we can modify permissions using the chmod command. Syntax of chmod:

chmod permissions filename

Where,

- permissions can be read, write, execute or a combination of them.
- filename is the name of the file for which the permissions need to change. This parameter can also be a list if files to change permissions in bulk.

We can change permissions using two modes:

1. Symbolic mode: this method uses symbols like u, g, o to represent users, groups, and others. Permissions are represented as r, w, x for read write and execute, respectively. You can modify permissions using +, - and =.

2. Absolute mode: this method represents permissions as 3-digit octal numbers ranging from 0-7. Now, let's see them in detail.

How to Change Permissions using Symbolic Mode

The table below summarize the user representation:

USER REPRESENTATION	DESCRIPTION
u	user/owner
g	Group
o	Other

We can use mathematical operators to add, remove, and assign permissions. The table below shows the summary:

OPERATOR	DESCRIPTION
+	Adds a permission to a file or directory
–	Removes the permission
=	Sets the permission if not present before. Also overrides the permissions if set earlier.

Example:

Suppose, I have a script and I want to make it executable for owner of the file zaira.

Current file permissions are as follows:

```
zaira@Zaira:~$
zaira@Zaira:~$ ls -lrt | grep mymotd.sh
-rw-r--r--  1 zaira zaira   77 Mar 11 13:41 mymotd.sh
```

Let's split the permissions like this:

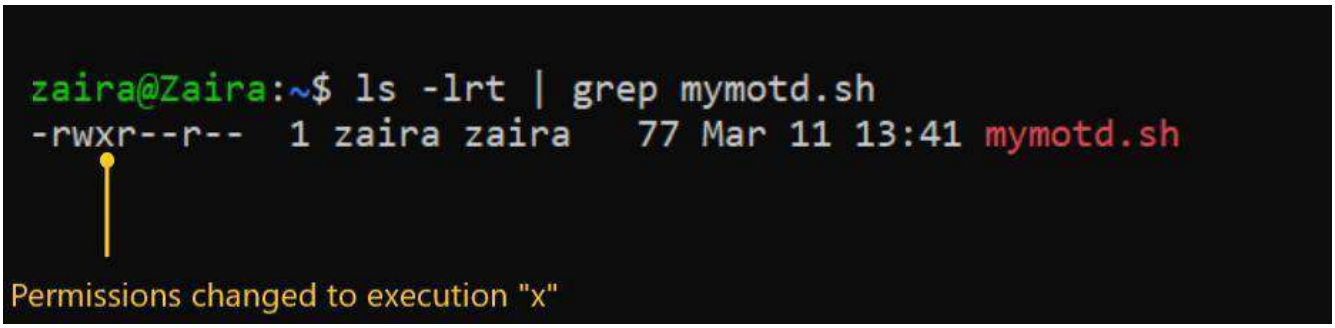
-	rw-	r--	r--
file type	user	group	other

To add execution rights (x) to owner (u) using symbolic mode, we can use the command below:

chmod u+x mymotd.sh

Output:

Now, we can see that the execution permissions have been added for owner zaira.



Additional examples for changing permissions via symbolic method:

- Removing read and write permission for group and others: chmod go-rw.
- Removing read permissions for others: chmod o-r.
- Assigning write permission to group and overriding existing permission: chmod g=w.

How to Change Permissions using Absolute Mode

Absolute mode uses numbers to represent permissions and mathematical operators to modify them.

The below table shows how we can assign relevant permissions:

PERMISSION	PROVIDE PERMISSION
Read add 4 Write	add
2 execute add 1	

Permissions can be revoked using subtraction. The below table shows how you can remove relevant permissions.

PERMISSION	REVOKE PERMISSION
Read subtract 4 Write subtract	
2 execute subtract 1	

Example:

- Set read (add 4) for user, read (add 4) and execute (add 1) for group, and only execute (add 1) for others.

chmod 451 file-name

This is how we performed the calculation:

	read	write	execute	sum
user	4	0	0	4
group	4	0	1	5
other	0	0	1	1

451

Final permission

Note that this is the same as r--r-x--x.

- Remove execution rights from other and group.

To remove execution from other and group, subtract 1 from the execute part of last 2 octets.

	read	write	execute	sum
user	4	0	0	4
group	4	0	1-1 (subtract)	4
other	0	0	1-1 (subtract)	0

440

Updated permission

- Assign read, write and execute to user, read and execute to group and only read to others. This would be the same as rwxr-xr--.

	read	write	execute	sum
user	4	2	1	7
group	4	0	1	5
other	4	0	0	4

754

Final permission

How to Change Ownership using the chown **Command**

Next, we will learn how to change the ownership of a file. You can change the ownership of a file or folder using the chown command. In some cases, changing ownership requires sudo permissions.

Syntax of chown:

chown user filename

How to change user ownership with **chown**

Let's transfer the ownership from user zaira to user news.

chown news mymotd.sh

Command to change ownership: sudo chown news mymotd.sh

Output:

Experiment 11

a)Execute shell commands through vi editor.?

The vi editor is elaborated as visual editor. It is installed in every Unix system. In other words, it is available in all Linux distros. It is user-friendly and works same on different distros and platforms. It is a very powerful application. An improved version of vi editor is vim.

The vi editor has two modes:

- Command Mode: In command mode, actions are taken on the file. The vi editor starts in command mode. Here, the typed words will act as commands in vi editor. To pass a command, you need to be in command mode.
- Insert Mode: In insert mode, entered text will be inserted into the file. The Esc key will take you to the command mode from insert mode.

By default, the vi editor starts in command mode. To enter text, you have to be in insert mode, just type 'i' and you'll be in insert mode. Although, after typing i nothing will appear on the screen but you'll be in insert mode. Now you can type anything.

To exit from insert mode press Esc key, you'll be directed to command mode.

If you are not sure which mode you are in, press Esc key twice and you'll be in command mode.

Using vi

The vi editor tool is an interactive tool as it displays changes made in the file on the screen while you edit the file.

In vi editor you can insert, edit or remove a word as cursor moves throughout the file.

Commands are specified for each function like to delete it's x or dd.

The vi editor is case-sensitive. For example, p allows you to paste after the current line while P allows you to paste before the current line.
vi syntax:

vi **<fileName>**

In the terminal when you'll type vi command with a file name, the terminal will get clear and content of the file will be displayed. If there is no such file, then a new file will be created and once completed file will be saved with the mentioned file name.

Linux vi example

Let's understand vi through an example:

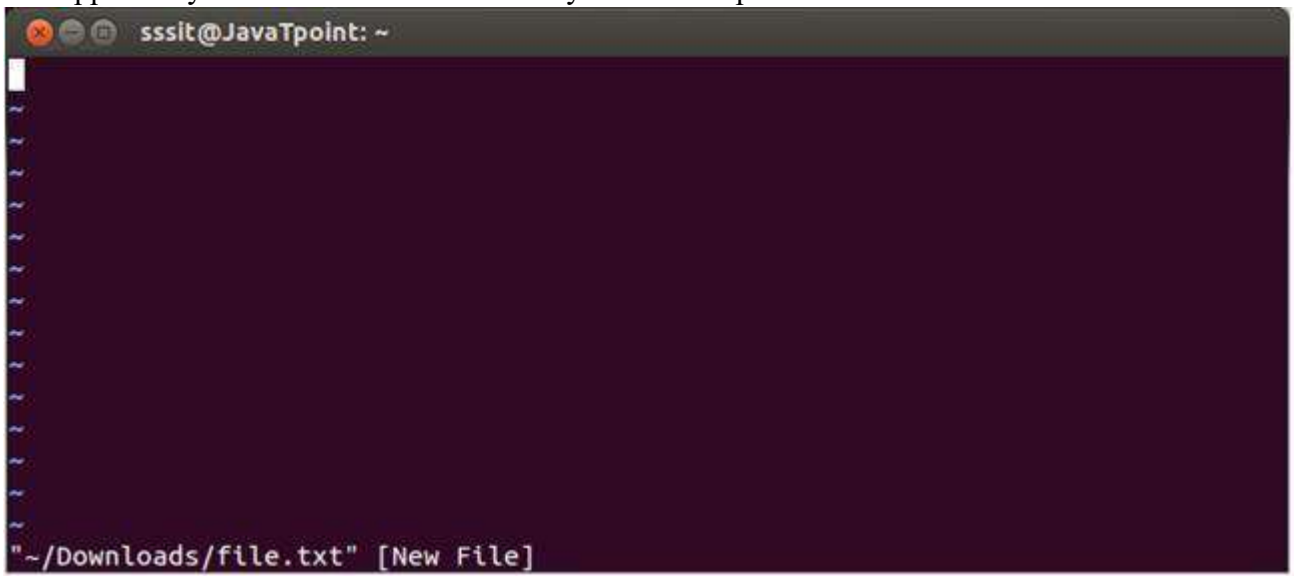
To start vi open your terminal and type vi command followed by file name. If your file is in some other directory, you can specify the file path. And if in case, your file doesn't exist, it will create a new file with the specified name at the given location. Example:

1. vi /home/sssit/Downloads/file.txt

Look at the above snapshot, we are creating a new file file.txt (as this file doesn't exist) and have entered the full path for the directory Downloads.

Command mode

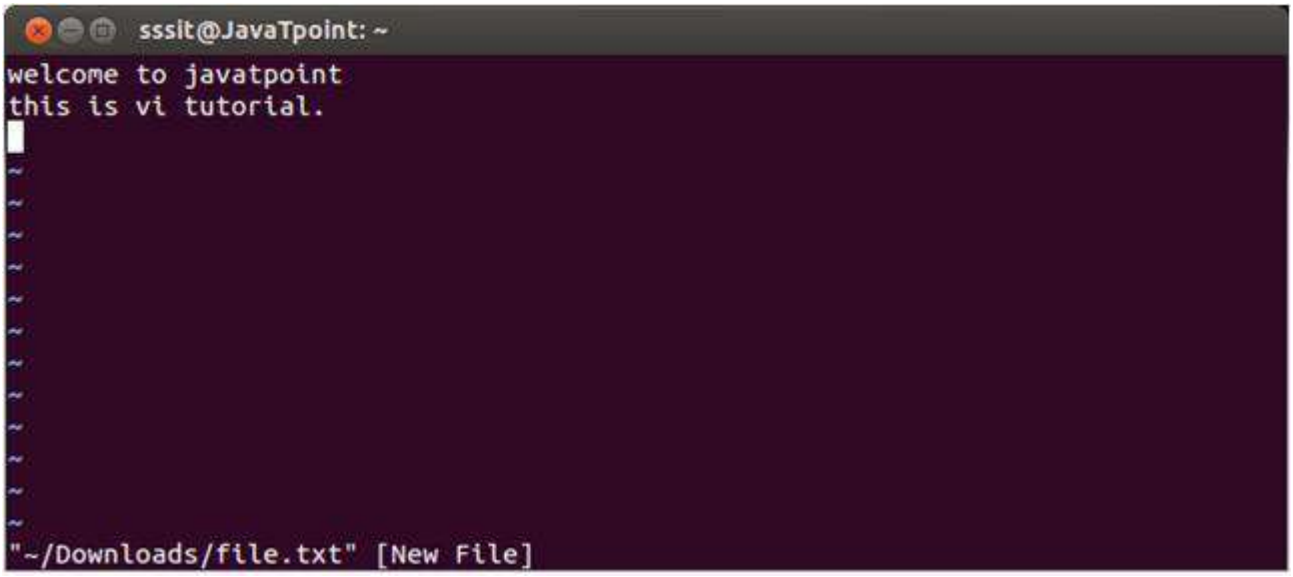
This is what you'll see when you'll press enter after the above command. If you'll start typing, nothing will appear as you are in command mode. By default vi opens in command mode.



Look at the above snapshot, it is blank as it is a new file. To start typing, you have to move to the insert mode. At the end of the terminal window, directory name and file name are displayed.

Insert mode

To move to the insert mode press i. Although, there are other commands also to move to insert mode which we'll study in next page.



Look at the above snapshot, after pressing i we have entered into insert mode. Now we can write anything. To move to the next line press enter.

Once you have done with your typing, press esc key to return to the command mode.

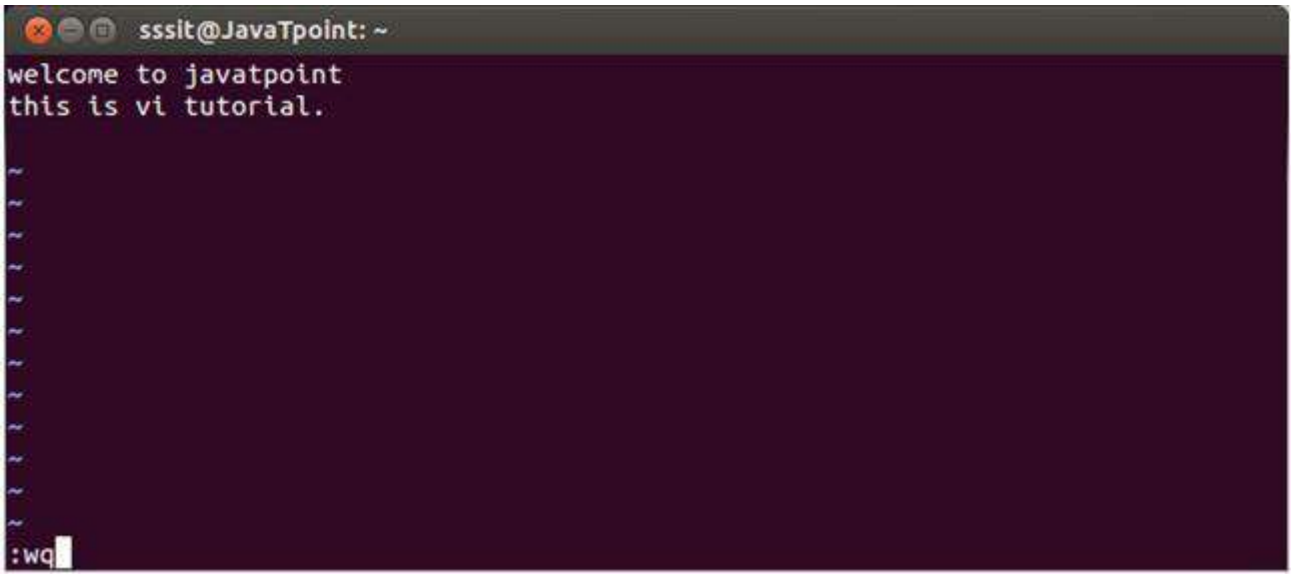
To save and quit

You can save and quit vi editor from command mode. Before writing save or quit command you have to press colon (:). Colon allows you to give instructions to vi. exit vi table:

Commands	Action
:wq	Save and quit
:w	Save
:q	Quit
:w fname	Save as fname
ZZ	Save and quit
:q!	Quit discarding changes made
:w!	Save (and write to non-writable file)

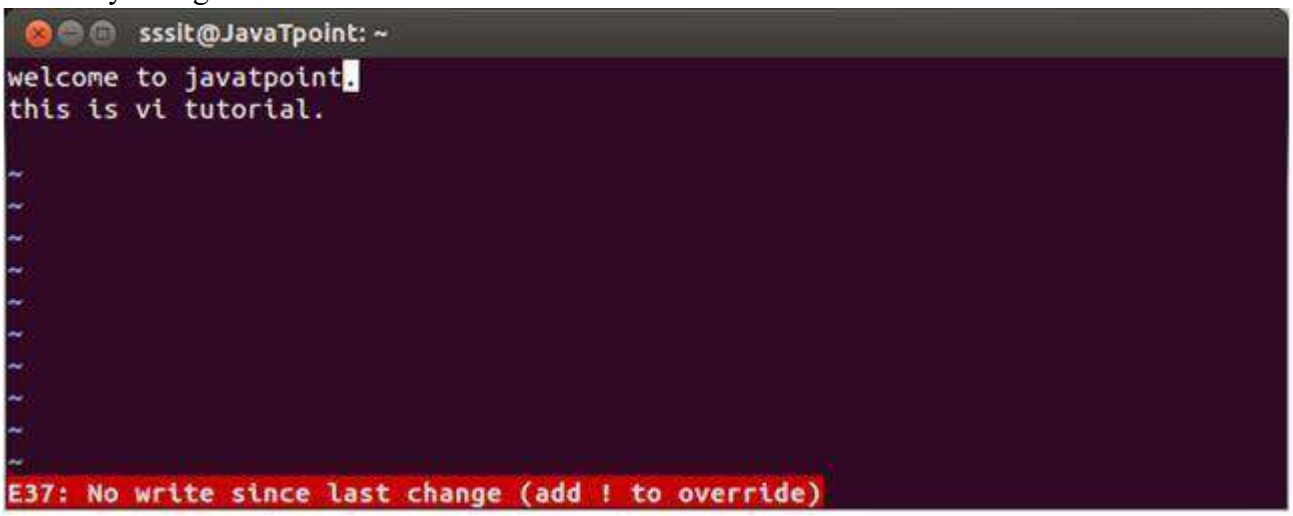
To exit from vi, first ensure that you are in command mode. Now, type :wq and press enter. It will save and quit vi.

Type :wq to save and exit the file.



Look at the above snapshot, command :wq will save and quit the vi editor. When you'll type it in command mode, it will automatically come at bottom left corner.

If you want to quit without saving the file, use :q. This command will only work when you have not made any changes in the file.



Look at the above snapshot, this file is modified and hence on typing :q it displays this message at bottom left corner.

The above file can be saved with the command :!q. It discards the changes made in the file and save it.

Look at the above snapshot, we have typed :!q, it will save our file by discarding the changes made.

Vi Commands

Linux vi editor is different from other editors. You have to use different keys to use different functions. Although, it's quite easy and interesting to use vi editor.

The vi editor commands are case sensitive.

Have a look at the vi commands in the following table.

To switch from command to insert mode:

Command	Action
I	Start typing before the current character
I	Start typing at the start of current line
A	Start typing after the current character
A	Start typing at the end of current line
O	Start typing on a new line after the current line
O	Start typing on a new line before the current line

To move around a file:

Commands	Action
J	To move down
K	To move up
H	To move left
L	To move right

To jump lines:

Commands	Action
G	Will direct you at the last line of the file
``	Will direct you to your last position in the file
Commands	**Action**
X	Delete the current character

X	Delete the character before the cursor
R	Replace the current character
xp	Switch two characters
dd	Delete the current line
D	Delete the current line from current character to the end of the line
dG	delete from the current line to the end of the file

To repeat and undo:

Commands	**Action**
U	Undo the last command
.	Repeat the last command

Command to cut, copy and paste:

Commands	**Action**
dd	Delete a line
yy	(yank yank) copy a line
P	Paste after the current line
P	Paste before the current line

Command to cut, copy and paste in blocks:

Commands	**Action**
<n>dd	Delete the specified n number of lines
<n>yy	Copy the specified n number of lines

Experiment 12

Write a shell script that accepts any number of arguments and prints them in the reverse order?

Algorithm

step 1:- Take user input in a string step 2:- Findthe length of

given string using length function step 3:- Set i = length-1

and run loop till i <= 0 step 4:- echo the $i step 5:- repeat

step 3 and 4 till i==0 step 6:- end

```
// reverse a string using shell script
// reverse a string is in linux and unix

#!/ bin / bash
// reading a string // using
via user input read - p "Enter
string:" string // getting the
length of given string
   len
```

```
    = $
  {
    #string
  }
  // looping for reversing a string
  // initialize i=len-1 for reversing a string and run till i=0
  // printing in the reverse order of the given string for ((i
  = $len - 1; i >= 0; i--)) do
    // "${string:$i:1}"extract single character from string.
reverse = "$reverse${string:$i:1}" done echo "$reverse"
```

Output:

```
Skeegrofskeeg
```

Experiment 13

Write client server programs using c for interaction between server and client process using Unix Domain sockets?

Algorithm:- Sample
UNIX server
Step 1:define NAME "socket"
Step 2: sock = socket(AF_UNIX, SOCK_STREAM, 0);
Step 3:if (sock < 0) perror("opening stream socket"); exit(1);
step4: server.sun_family = AF_UNIX; strcpy(server.sun_path,
NAME);
if (bind(sock, (struct sockaddr *) &server, sizeof(struct sockaddr_un)))
{ perror("binding stream socket"); exit(1);
} step 5: print ("Socket has name %s\n",
server.sun_path); listen(sock, 5);
step 6: for (;;)
{ msgsock = accept(sock, 0,
0);
if (msgsock == -1) perror("accept");
else
do { bzero(buf, sizeof(buf));
if ((rval = read(msgsock, buf, 1024)) < 0)

perror("reading stream message");
else if (rval == 0)
else print ("-->%s\n", buf);
} while (rval > 0);
close(msgsock);
}

close(sock); unlink(NAME);
}

Step 7:stop

Experiment 14

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

Algorithm:

1.First check the provides argument is the directory or not using the if statement using the -d option for the first argument using the $1 parameter. If it is true then print the message that the provided argument is the directory.

2.If the argument is not the directory then check it for the file. Use the -f option and if statement for the first argument using the $1 parameter. If it is true then the print the message that provided the argument is the file.

3.If both conditions are false then it is clear that the provided argument is neither file and nor directory. Then print the message the given argument is neither file and nor directory.

Script:

```
#!/bin/sh
#Using -d option we are checking whether the first argument is a directory or not.
#$1 refers to the first argument if [ -d $1 ] then echo
"The provided argument is the directory."
#Using -f option we are checking whether the first argument is a file or not.
elif [ -f $1 ] then echo "The provided
argument is the file."
#if the provided argument is not file and directory then it does not exist on the system.
else echo "The given argument does not exist on the file system."
fi
./script_name.sh filename # For files
```

./script_name.sh foldername # For folders

Viva questions

1. define shared memory

2.what are file locking functions.

3.what are shared memory systemcalls.

4.define internet domain sockets

5.Difference between internet and unix domain sockets

.

Exercises

1 Write a program to demonstrate communication of two different process via shared memory

2 Write a program to demonstrate that the shared memory created will beavailable even after the process which created is exited.

VISION OF THE DEPARTMENT

To be a centre of excellence for providing quality technical education to develop future leaders with the aspects of research &computing, Software product development and entrepreneurship.

MISSION OF THE DEPARTMENT

M1: To offer academic programme with state of art curriculum having flexibility for accommodating the latest developments in the areas of Computer science engineering

M2: To conduct research and development activities in contemporary and emerging areas of computer science & engineering.

M3: To inculcate moral values & entrepreneurial skills to produce professionals capable of providing socially relevant and sustainable solutions.