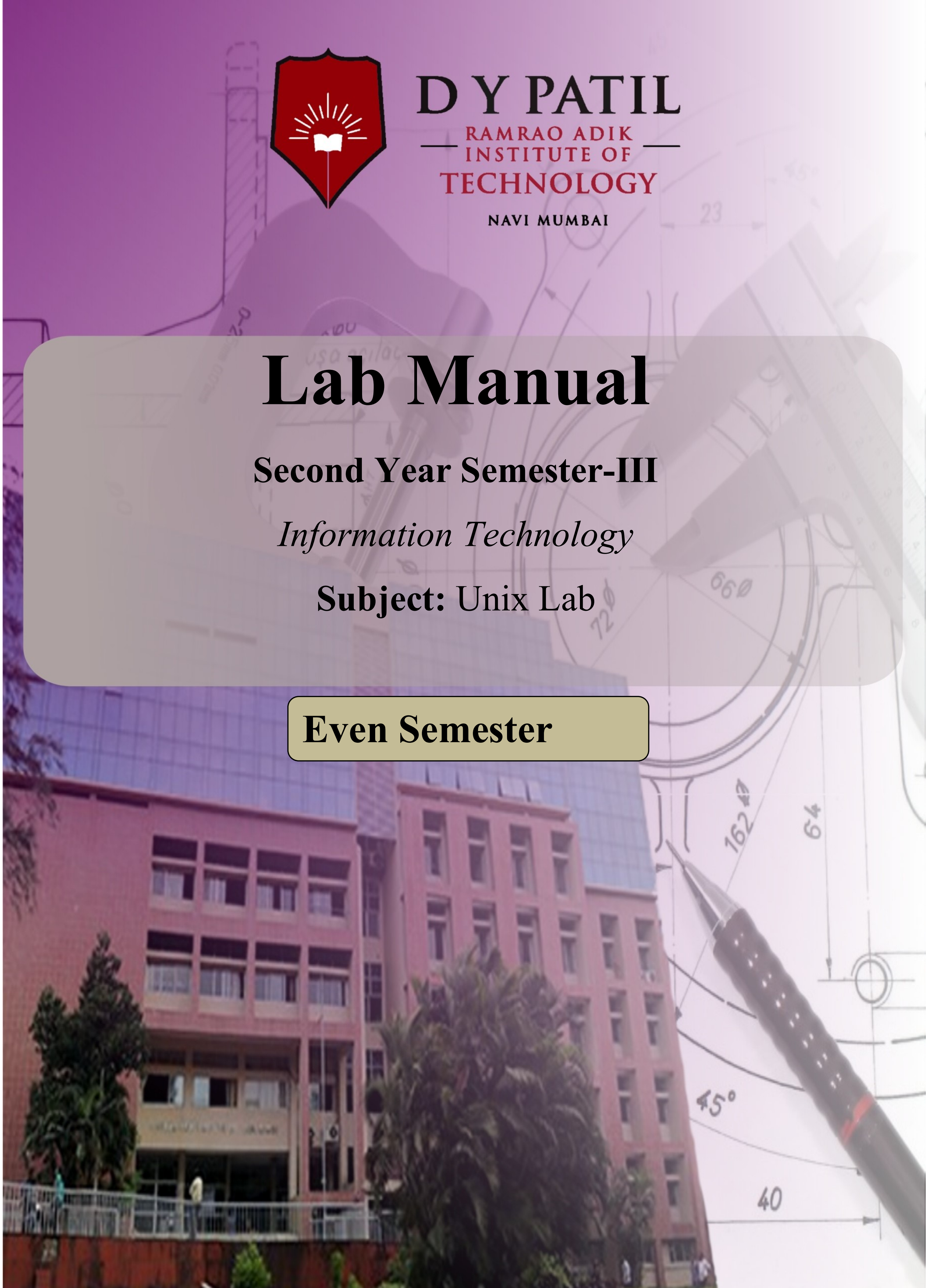


D Y PATIL
RAMRAO ADIK INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

Lab Manual

Second Year Semester-III

Information Technology

Subject: Unix Lab

Even Semester

Institutional Vision, Mission and Quality Policy

Our Vision

To foster and permeate higher and quality education with value added engineering, technology programs, providing all facilities in terms of technology and platforms for all round development with societal awareness and nurture the youth with international competencies and exemplary level of employability even under highly competitive environment so that they are innovative adaptable and capable of handling problems faced by our country and world at large.

RAIT's firm belief in new form of engineering education that lays equal stress on academics and leadership building extracurricular skills has been a major contribution to the success of RAIT as one of the most reputed institution of higher learning. The challenges faced by our country and world in the 21 Century needs a whole new range of thought and action leaders, which a conventional educational system in engineering disciplines are ill equipped to produce. Our reputation in providing good engineering education with additional life skills ensures that high grade and highly motivated students join us. Our laboratories and practical sessions reflect the latest that is being followed in the Industry. The project works and summer projects make our students adept at handling the real life problems and be Industry ready. Our students are well placed in the Industry and their performance makes reputed companies visit us with renewed demands and vigour.

Our Mission

The Institution is committed to mobilize the resources and equip itself with men and materials of excellence thereby ensuring that the Institution becomes pivotal center of service to Industry, academia, and society with the latest technology. RAIT engages different platforms such as technology enhancing Student Technical Societies, Cultural platforms, Sports excellence centers, Entrepreneurial Development Center and Societal Interaction Cell. To develop the college to become an autonomous Institution & deemed university at the earliest with facilities for advanced research and development programs on par with international standards. To invite international and reputed national Institutions and Universities to collaborate with our institution on the issues of common interest of teaching and learning sophistication.

RAIT's Mission is to produce engineering and technology professionals who are innovative and inspiring thought leaders, adept at solving problems faced by our nation and world by providing quality education.

The Institute is working closely with all stake holders like industry, academia to foster knowledge generation, acquisition, dissemination using best available resources to address the great challenges being faced by our country and World. RAIT is fully dedicated to provide its students skills that make them leaders and solution providers and are Industry ready when they graduate from the Institution.

We at RAIT assure our main stakeholders of students 100% quality for the programmes we deliver. This quality assurance stems from the teaching and learning processes we have at work at our campus and the teachers who are handpicked from reputed institutions IIT/NIT/MU, etc. and they inspire the students to be innovative in thinking and practical in approach. We have installed internal procedures to better skills set of instructors by sending them to training courses, workshops, seminars and conferences. We have also a full fledged course curriculum and deliveries planned in advance for a structured semester long programme. We have well developed feedback system employers, alumni, students and parents from to fine tune Learning and Teaching processes. These tools help us to ensure same quality of teaching independent of any individual instructor. Each classroom is equipped with Internet and other digital learning resources.

The effective learning process in the campus comprises a clean and stimulating classroom environment and availability of lecture notes and digital resources prepared by instructor from the comfort of home. In addition student is provided with good number of assignments that would trigger his thinking process. The testing process involves an objective test paper that would gauge the understanding of concepts by the students. The quality assurance process also ensures that the learning process is effective. The summer internships and project work based training ensure learning process to include practical and industry relevant aspects. Various technical events, seminars and conferences make the student learning complete.

Our Quality Policy

ज्ञानधीनं जगत् सर्वम।

Knowledge is supreme.

Our Quality Policy

It is our earnest endeavour to produce high quality engineering professionals who are innovative and inspiring, thought and action leaders, competent to solve problems faced by society, nation and world at large by striving towards very high standards in learning, teaching and training methodologies.

Our Motto: If it is not of quality, it is NOT RAIT!

Department Vision & Mission

Vision

- To pervade higher and quality education with value added engineering, technology programs to deliver the IT graduates with knowledge, skills, tools and competencies necessary to understand and apply the technical knowledge and to become competent to practice engineering professionally and ethically in tomorrow's global environment.
- To contribute to the overall development by imparting moral, social and ethical values.

Mission

- The mission of the IT department is to prepare students for overall development including employability, entrepreneurship and the ability to apply the technology to real life problems by educating them in the fundamental concepts, technical skills/programming skills, depth of knowledge and development of understanding in the field of Information Technology.
- To develop entrepreneurs, leaders and researchers with exemplary level of employability even under highly competitive environments with high ethical, social and moral values.

Index

List of Experiments

Sr. No.	Experiments Name
1	(a)Study experiment on Installation of Ubuntu (b) Execution of various Unix general purpose utility commands
2	Study and Execution of Unix networking commands
3	(a)Study and implementation of vi editor (b) Study of File management system in Unix
4	Implementation of i)User Management ii)Process Management iii)Memory Management in Unix
5	(a)Write a shell script program to display "Hello World" (b) Write a shell script program to check whether given number is even or odd
6	Write a shell script program to copy contents of one file to another
7	(a)Write a program using sed command to print duplicated lines of input (b)Use a pipeline and command substitution to set length of a line in file to a variable
8	Write a grep/egrep program to find number of words character, words, line in file
9	Write a perl script to check number is prime or not
10	Write an awk script to develop a Fibonacci series
11	Mini Project

Experiment Plan & Lab Outcomes

Lab Outcomes:

LO1	Identify the basic Unix general purpose commands
LO2	Apply and change the ownership and file permission using advance Unix commands
LO3	Use the awk, grep and perl scripts
LO4	Implement shell script and sed
LO5	Apply basics of administrative task
LO6	Apply networking Unix commands

Module No.	Week No.	Experiments Name	Lab Outcome
1	W1	(a)Study experiment on Installation of Ubuntu (b) Execution of various Unix general purpose utility commands	LO1
2	W2	Study and Execution of Unix networking commands	LO6
3	W3	(a)Study and implementation of vi editor (b) Study of File management system in Unix	LO2
4	W4&W5	Implementation of i)User Management ii)Process Management iii)Memory Management in Unix	LO5
5	W6	(a)Write a shell script program to display "Hello World" (b) Write a shell script program to check whether given number is even or odd	LO4
6	W7	Write a shell script program to copy contents of one file to another	LO4
7	W8	(a)Write a program using sed command to print duplicated lines of input (b)Use a pipeline and command substitution to set length of a line in file to a variable	LO4
8	W9	Write a grep/egrep program to find number of words character, words, line in file	LO3

9	W10	Write a perl script to check number is prime or not	LO3
10	W111	Write an awk script to develop a Fibonacci series	LO3
11	W12&W13	Mini Project	LO1,LO2,LO3, LO4,LO5,LO6

Study and Evaluation Scheme

Course Code	Course Name	Theory	Practical	Tutorial	Theory	TW/ Practical	Tutorial	Total
ITL 402	Unix Lab	-	02	--	-		--	01

Course Code	Course Name	Examination Scheme						
		Theory Marks				Term Work	Practical & Oral	Total
		Internal Assessment			End Sem marks			
		Test 1	Test 2	Avg Of 2 test				
ITL 402	Unix Lab	-	-	-	-	25	25	50

Term Work:

1. Term work assessment must be based on the overall performance of the student with every experiment graded from time to time.
2. The final certification and acceptance of term work ensures satisfactory performance of laboratory work and minimum passing marks in term work.

Practical & Oral:

1. Practical & Oral exam will be based on the entire syllabus of Unix

Unix Lab

Experiment No. : 1(a)

Study experiment on Installation of Ubuntu

Experiment No. 1(a)

Aim: Study experiment to understand the installation of ubuntu

What will you learn by performing this experiment?

Installation of ubuntu

Theory:

When it comes to installing popular Linux flavour Ubuntu, there are so many useful snippets of information on blogs and guides all over the internet.

How To Install Ubuntu

- Download Ubuntu
- Check if Your Computer will Boot from USB
- Make BIOS Changes
- Try Ubuntu Before you Install It
- Create Bootable USB
- Install Ubuntu

Create a username, password, and computer name. You will login with this user id after the installation is complete.

Installation of Ubuntu in windows using Virtual box

VirtualBox allows you to run an entire operating system inside another operating system. Please be aware that you should have a minimum of 512 MB of RAM. 1 GB of RAM or more is recommended.

Comparison to Dual-Boot

Many websites (including the one you're reading) have tutorials on setting up dual-boots between Windows and Ubuntu. A dual-boot allows you, at boot time, to decide which operating system you want to use. Installing Ubuntu on a virtual machine inside of Windows has a lot advantages over a dual-boot (but also a few disadvantages).

Advantages of virtual installation

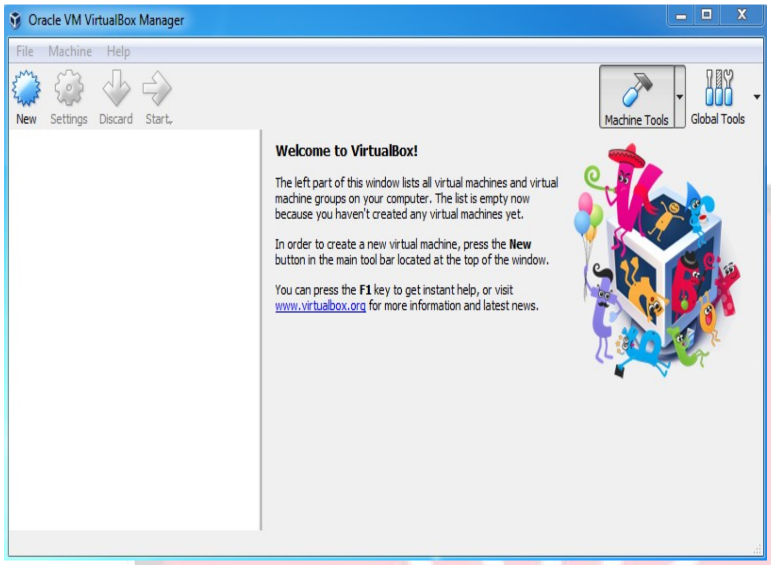
- The size of the installation doesn't have to be predetermined. It can be a dynamically resized virtual hard drive.
- You do not need to reboot in order to switch between Ubuntu and Windows.
- The virtual machine will use your Windows internet connection, so you don't have to worry about Ubuntu not detecting your wireless card, if you have one.
- The virtual machine will set up its own video configuration, so you don't have to worry about installing proprietary graphics drivers to get a reasonable screen resolution.
- You *always* have Windows to fall back on in case there are any problems. All you have to do is press the right Control key instead of rebooting your entire computer.
- For troubleshooting purposes, you can easily take screenshots of any part of Ubuntu (including the boot menu or the login screen).
- It's low commitment. If you later decide you don't like Ubuntu, all you have to do is delete the virtual hard drive and uninstall VirtualBox.

Disadvantages of virtual installation

- In order to get any kind of decent performance, you need at least 512 MB of RAM, because you are running an entire operating system (Ubuntu) inside another entire operating system (Windows). The more memory, the better. I would recommend at least 1 GB of RAM.
- Even though the low commitment factor can seem like an advantage at first, if you later decide you want to switch to Ubuntu and ditch Windows completely, you cannot simply delete your Windows partition. You would have to find some way to migrate out your settings from the virtual machine and then install Ubuntu over Windows outside the virtual machine.
- Every time you want to use Ubuntu, you have to wait for two boot times (the time it takes to boot Windows, and then the time it takes to boot Ubuntu within Windows).

Installation Procedure

The first thing you have to do is obtain VirtualBox. Visit the VirtualBox website's download page. Install it the same way you would any normal Windows program.

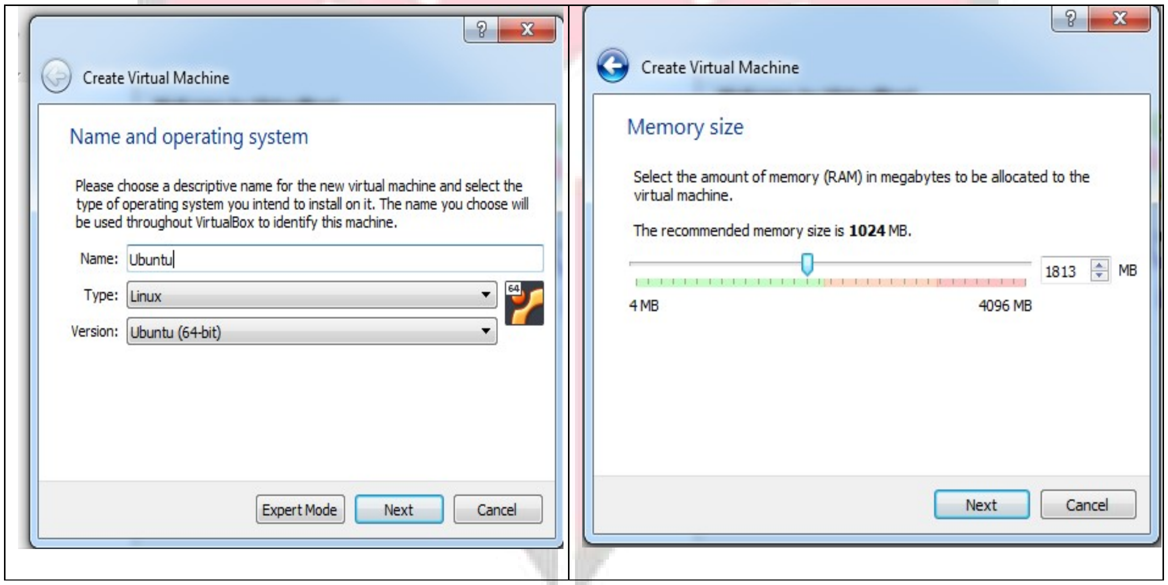

Oracle VM VirtualBox Manager
File Machine Help
New Settings Discard Start
Machine Tools Global Tools

Welcome to VirtualBox!

The left part of this window lists all virtual machines and virtual machine groups on your computer. The list is empty now because you haven't created any virtual machines yet.

In order to create a new virtual machine, press the **New** button in the main tool bar located at the top of the window.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for more information and latest news.



Create Virtual Machine

Name and operating system

Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name: Ubuntu

Type: Linux

Version: Ubuntu (64-bit)

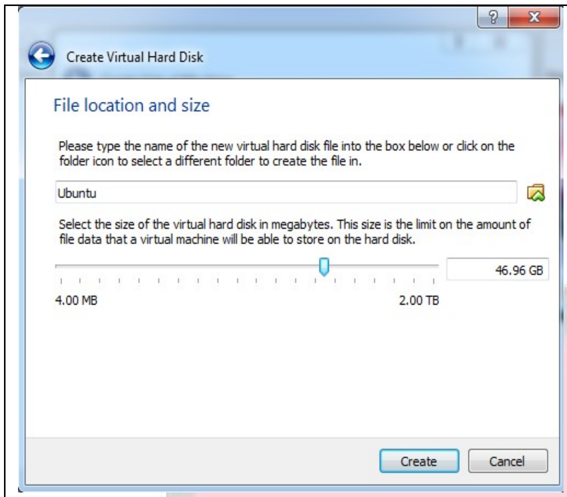
Create Virtual Machine

Memory size

Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **1024** MB.

1813 MB



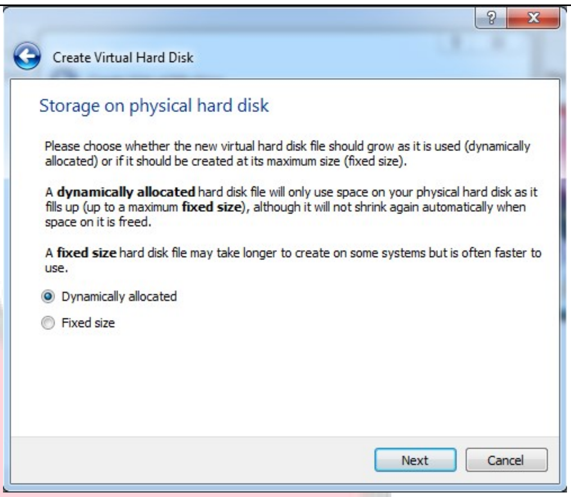
File location and size

Please type the name of the new virtual hard disk file into the box below or click on the folder icon to select a different folder to create the file in.

Ubuntu

Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.

46.96 GB



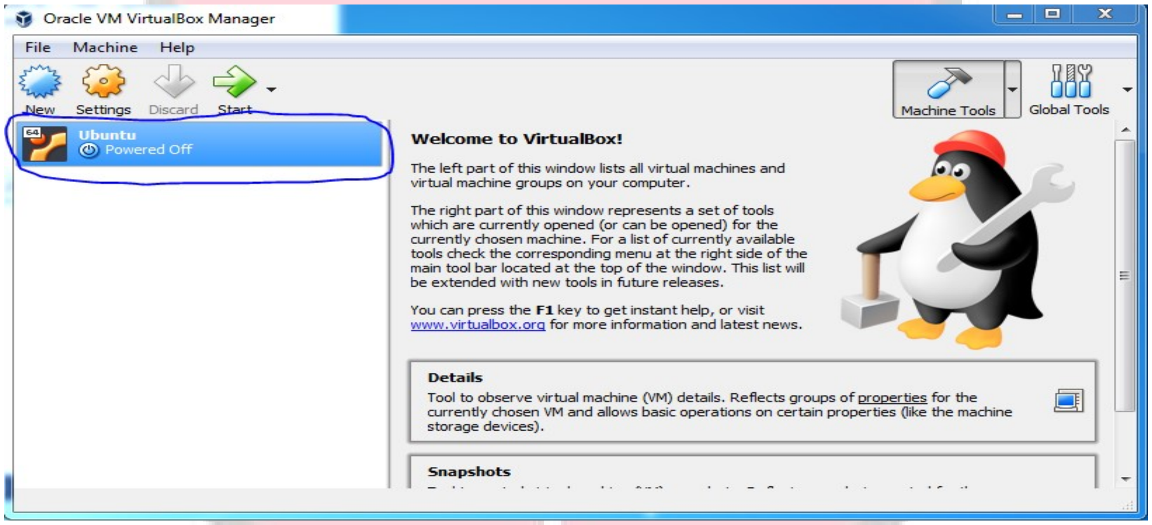
Storage on physical hard disk

Please choose whether the new virtual hard disk file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).

A **dynamically allocated** hard disk file will only use space on your physical hard disk as it fills up (up to a maximum **fixed size**), although it will not shrink again automatically when space on it is freed.

A **fixed size** hard disk file may take longer to create on some systems but is often faster to use.

[✓ Dynamically allocated] Fixed size



Welcome to VirtualBox!

The left part of this window lists all virtual machines and virtual machine groups on your computer.

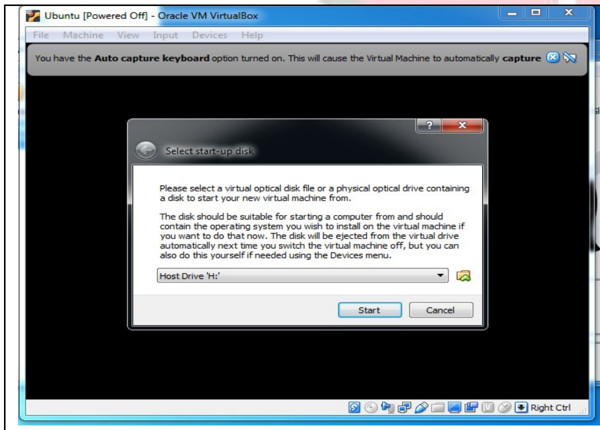
The right part of this window represents a set of tools which are currently opened (or can be opened) for the currently chosen machine. For a list of currently available tools check the corresponding menu at the right side of the main tool bar located at the top of the window. This list will be extended with new tools in future releases.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for more information and latest news.

Details

Tool to observe virtual machine (VM) details. Reflects groups of properties for the currently chosen VM and allows basic operations on certain properties (like the machine storage devices).

Snapshots

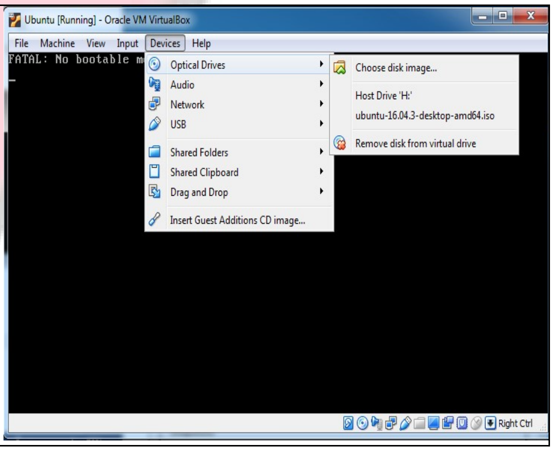


You have the **Auto capture keyboard** option turned on. This will cause the Virtual Machine to automatically **capture**

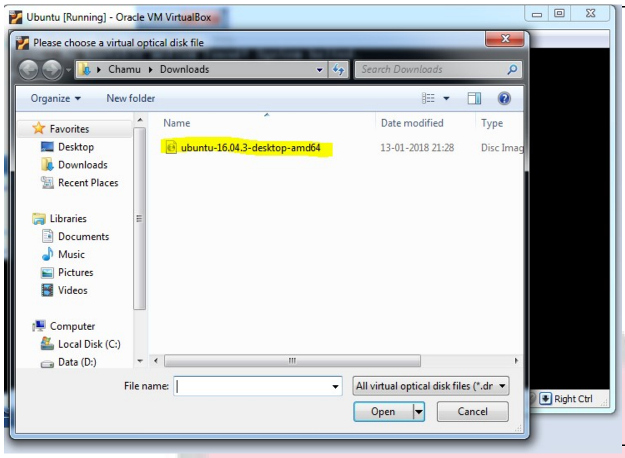
Please select a virtual optical disk file or a physical optical drive containing a disk to start your new virtual machine from.

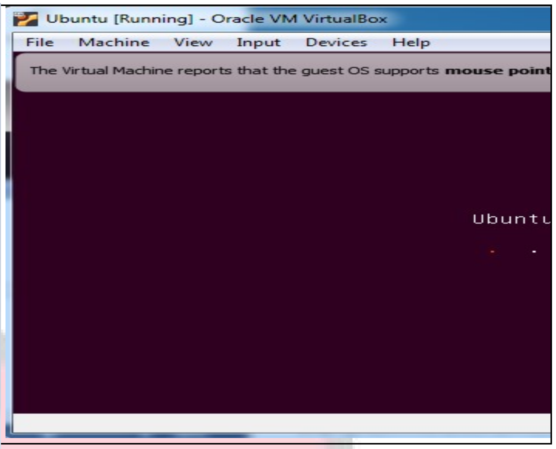
The disk should be suitable for starting a computer from and should contain the operating system you wish to install on the virtual machine if you want to do that now. The disk will be ejected from the virtual drive automatically next time you switch the virtual machine off, but you can also do this yourself if needed using the Devices menu.

Host Drive 'H:'



```
FATAL: No bootable m
_
```




Ubuntu [Running] - Oracle VM VirtualBox
File
Machine
View
Input
Devices
Help
The Virtual Machine reports that the guest OS supports mouse point
Ubuntu

Conclusion and Discussion: We have studied installation of Ubuntu through bootable USB and as well as through virtual machine also.

QUIZ / Viva Questions:

1. What is a operating system?
2. What are steps of installation?

Links:

https://askubuntu.com/questions/367248/how-to-install-virtualbox-from-command-line

Unix Lab

Experiment No. : 1(b)

Execution of various Unix general purpose utility commands

Experiment No. 1(b)

Aim: To study and execute various general purpose utility commands of Unix.

What will you learn by performing this experiment?

Identify the basic UNIX Commands

Software Required: Ubuntu

Theory:

UNIX is a multiuser, multiprocessing, portable operating system designed to facilitate programming, text processing, communication, and many other tasks that are expected for an operating system. It contains hundreds of simple, single-purpose functions that are combined to do virtually every processing task imaginable. It is flexible as it is used in three different computing environments: stand-alone personal environment, time- sharing system, and client/ server systems.

Commands:

Unix and Unix-like systems include a large core of standard utilities for editing text, writing, compiling, and controlling programs, processing, manipulating the user environment, and retrieving information about the system and its users. Users enter commands and arguments on the shell command line, and then the shell interprets them and passes them to the kernel for execution. Following is a very brief introduction to some UNIX commands, including examples of how to use each command. For more extensive information about any of these commands, use the man command as described below.

pwd :

pwd - Print Working Directory. pwd command prints the full filename of the current working directory.

SYNTAX: pwd [options]

cd :

cd command is used to change the director

SYNTAX: cd [directory | ~ | ./ | ../ | -]

ls :

ls command lists the files and directories under current working directory.

SYNTAX: ls [OPTIONS]... [FILE]

OPTIONS:

-l	lists all the files, directories and their mode, Number of links,owner of the file, file size, modified date and time and filename.
-t	lists in order of last modification time.
-a	lists all entries including hidden files.
-d	lists directory files instead of contents.
-p	puts slash at the end of each directories.
-u	list in order of last access time.
-i	Display inode information

rm:

rm linux command is used to remove/delete the file from the directory.

SYNTAX: rm [options..] [file | directory]

OPTIONS:

-f	Remove all files in a directory without prompting the user.
-i	Interactive. With this option, rm prompts for confirmation before removing any files.

mv:

mv command which is short for move. It is used to move/rename file from one directory to another. mv command is different from cp command as it completely removes the file from the source and moves to the directory specified, where cp command just copies the content from one file to another.

SYNTAX: mv [-f] [-i] oldname newname

OPTIONS:

-f	will move the file(s) without prompting even if it is writing over an existing target.
-i	-i Prompts before overwriting another file.

cat:

cat linux command concatenates files and print it on the standard output.

SYNTAX: cat [OPTIONS] [FILE]...

OPTIONS:

-A	Show all.
-b	Omits line numbers for blank space in the output.
-E	Displays a $ (dollar sign) at the end of each line.
-n	Line numbers for all the output lines.

cmp:

cmp linux command compares two files and tells you which line numbers are different.

SYNTAX: cmp [options..] file1 file2

OPTIONS:

-c	Output differing bytes as characters.
-l	Print the byte number (decimal) and the differing byte values (octal) for each difference.
-s	Prints nothing for differing files, return exit status only.

cp:

cp command copy files from one location to another. If the destination is an existing file, then the file is overwritten; if the destination is an existing directory, the file is copied into the directory (the directory is not overwritten).

SYNTAX: cp [OPTIONS]... SOURCE DEST

echo:

echo command prints the given input string to standard output.

SYNTAX: echo [options..] [string]

mkdir:

mkdir command is used to create one or more directories.

SYNTAX: mkdir [options] directories

rmdir:

rmdir command is used to delete/remove a directory and its subdirectories.

SYNTAX: rmdir [options..] Directory

OPTIONS:

-p	Allow users to remove the directory dir name and its parent directories which become empty.

who:

who command lets you display the users that are currently logged into your Unix computer system. ‘who’ is the basic command with no command-line arguments. It shows the names of users that are currently logged in, and may also show the terminal they're logged in on, and the time they logged in.

SYNTAX: who [options..][files]

Conclusion and Discussion: In this way we can run different file and directory handling commands and see the output on standard output window.

QUIZ / Viva Questions:

1. List all commands of Unix?
2. Explain options of ls command.
3. What is who command?

Text Books:

1. Unix, concepts and applications by Sumitabha Das, McGraw-Hill

2. Mastering Shell Scripting, Randal. K. Michael , Second Edition, Wiley Publication

References:

1. Unix Shell Programming by Yashwant Kanetkar

2. Unix shell programming by forozun

Unix Lab

Experiment No. : 2

Study and Execution of Unix networking commands

Experiment No. 2

Aim: To study and execute basic UNIX networking commands.

What will you learn by performing this experiment?

Networking commands of UNIX and its usage

Theory:

Networking is an essential part of UNIX and it offer lots of tools and command to diagnose any networking problem. Some of them are listed below:

- finding host/domain name and IP address - **hostname**
- test network connection – **ping**
- getting network configuration – **ifconfig**
- Network connections, routing tables, interface statistics – **netstat**
- query DNS lookup name – **nslookup**
- communicate with another hostname – **telnet**
- outing steps that packets take to get to network host – **traceroute**
- view user information – **finger**
- checking status of destination host – **telnet**

Example of Networking commands in Unix

Some example of various networking command in UNIX are shown here. Some of them are quite basic e.g. ping and telnet and some are more powerful e.g. nslookup and netstat. When you used these commands in a combination of find and grep you can get anything you are looking for e.g. hostname, connection endpoints, connection status etc.

hostname

hostname *with no options displays the machine's hostname*

hostname –d *displays the domain name the machine belongs to*

hostname –f *displays the fully qualified host and domain name*

hostname –i *displays the IP address for the current machine*

ping

It sends packets of information to the user-defined source. If the packets are received, the destination device sends packets back. Ping can be used for two purposes:

1. To ensure that a network connection can be established.

2. Timing information as to the speed of the connection.

If you do ping www.yahoo.com it will display its IP address. Use ctrl+C to stop the test.

ifconfig

One of the most basic networking commands is ifconfig. It will tell you about your network interfaces, the state that they're in, your assigned IP addresses, and even provide some counts of packets that have crossed the interface since the system was last booted. These days, you may see both ipv4 and ipv6 addresses.

To view your network settings, you would use the ifconfig command. If you try this command and get an error that claims the command cannot be found, it is likely that /sbin is not on your search path. You can try typing /sbin/ifconfig instead or you can modify your search path and then try ifconfig again.

```
$ ifconfig

-bash: ifconfig: command not found

$ PATH=$PATH:/sbin

$ ifconfig

eth0     Link encap:Ethernet  HWaddr 00:16:35:69:BD:79

         inet addr:192.168.0.11  Bcast:192.168.0.255  Mask:255.255.255.0

         inet6 addr: fe80::212:ff35:fe69:bd12/64 Scope:Link

         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

         RX packets:16791460 errors:0 dropped:0 overruns:0 frame:0

         TX packets:30066639 errors:0 dropped:0 overruns:0 carrier:0
```

```
        collisions:0 txqueuelen:1000

        RX bytes: 2182259898 (2.0 GiB)   TX bytes: 3951206845 (3.6 GiB)

        Interrupt:209 Memory:fdef0000-fdf00000
```

Notice the RX (receive) and TX (transmit) packets counts as well as the same basic statistics shown in bytes. These numbers may seem huge but the system was been running for roughly a year. The uptime command shows you this along with the load averages that indicate how hard this system is working to keep up with the load. In this case, the system isn't breaking a sweat. The 0.10 one minute load basically tells you that every 10th time the system checks, there is one process having to wait for access to the CPU. The 0.01 fifteen minute numbers tells you we're seeing something waiting only one time in a hundred.

```
$ uptime 09:56:19 up 325 days, 18:16, 1 user, load average: 0.10, 0.06, 0.01
```

netstat

Most useful and very versatile for finding a connection to and from the host. You can find out all the multicast groups (network) subscribed by this host by issuing "netstat -g"

netstat -nap | grep port will display process id of application which is using that port

netstat -a or netstat –all will display all connections including TCP and UDP

netstat --tcp or netstat –t will display only TCP connection

netstat --tcp or netstat –t will display only TCP connection

netstat --udp or netstat –u will display only UDP connection

netstat -g will display all multicast network subscribed by this host.

netstat -r will displays the routing table on your system.

nslookup

If you know the IP address it will display hostname. To find all the IP addresses for a given domain name, the command nslookup is used. You must have a connection to the internet for this utility to be useful, e.g.

$ **nslookup blogger.com**

you can also use the **nslookup** to convert hostname to IP Address and from IP Address to the hostname.

traceroute

A handy utility to view the number of hops and response time to get to a remote system or website is traceroute. Again you need an internet connection to make use of this tool.

The traceroute command will attempt to provide a list of all the routers your connections cross when reaching out to a remote system. The output also provides some information on how long each segment of the path takes, thus giving you some notion of the quality of a connection.

```
$ traceroute fermion

  traceroute to fermion (10.10.10.10), 30 hops max, 40 byte packets

 1 coresys (10.10.10.2) 0.965ms 1.010 ms   1.006 ms

 2 rtr2 (10.10.8.10)    0.613ms 0.627 ms 0.642 ms

 3 10.210.1.14 (10.20.1.11) 78.226 ms 77.779 ms 77.734 ms

 4 fermion (10.1.2.3) 77.257 ms! X 81.914 ms! X 73.626 ms! X
```

If traceroute gets to aa point at which it is unable to provide further information, you will start to see asterisks where you used to see system names and timing.

finger

View user information, displays a user’s login name, real name, terminal name and write status. This is pretty old Unix command and rarely used nowadays.

telnet

Connects destination host via the telnet protocol, if telnet connection establishes on any port means connectivity between two hosts is working fine.

$ telnet hostname port

will telnet hostname with the port specified. Normally it is used to see whether the host is alive and the network connection is fine or not.

Conclusion and Discussion: Thus we have studied UNIX networking commands and its other functions.

QUIZ / Viva Questions:

1. What are Unix Networking command?
2. What is the use of ping and netstat command?

Text Books:

1. Unix, concepts and applications by Sumitabha Das, McGraw-Hill

2. Mastering Shell Scripting, Randal. K. Michael , Second Edition, Wiley Publication

References:

1. Unix Shell Programming by Yashwant Kanetkar

2. Unix shell programming by forozun

Unix Lab

Experiment No. : 3(a)

Study and implementation of vi editor

Experiment No.3(a)

Aim: Programs to study and implementation of vi editor.

What will you learn by performing this experiment?
vi editor and its usage method

Theory:

The *vi* editor is one of the most common text editors on Unix. It was developed starting around 1976 by Bill Joy. Compared to most of its many clones, the traditional *vi* is a rather small program (the binary size is approximately 160 kBytes on i386) just with its extremely powerful editing interface, but lacking fancy features like multiple undo, multiple screens, or syntax highlighting.

"vi" is a screen oriented text editor originally created for the Unix OS. The name "vi" is derived from the shortest unambiguous abbreviation for the ex command visual, which switches the ex line editor to visual mode. vi is included in the most popular Linux like Ubuntu, Linux Mint or Debian. Also you can use it in another UNIX based systems like FreeBSD, OpenBSD or MINIX. Vi dosen't hold your hand and provide a list of keyboard shortcuts on the screen.

vi editor has the following operation modes-

1. **Command mode:** This enable to perform administrative tasks such as saving files, executing commands, moving the cursor, cutting and pasting lines or words, and finding and replacing.
2. **Text mode:** This mode enables you to insert text into the file.

Steps to write a very small C program using vi:

1. Open a terminal
2. To create a new file, type:

   ```
   vi filename.txt
   ```

3. vi is in command mode by default, if you want to write something, you can't do it.
4. Type "i", it enable you to write your code.
5. Now in insert mode, if we want to delete then we can't.
6. Type ESC to change to command mode.
7. In command mode, type "x", which delete character under cursor.
8. You can insert a character to the left of the cursor Typeing "i"
9. You can insert a character to the right of the cursor Typeing "a"
10. Insert ">" in the text
11. Go back to the command mode and type :wq, then type return
12. Now you're in the prompt again, see the content of your file using cat:
13. Open your file again (vi file.txt) and write the following code:

```
File.txt
Welcome to Unix lab
Performing all Unix commands and shell scripting
```

14. Save your file and use cat command to see the content of the file as cat file.txt

vi common options to open a file:

vi filename

It creates a new file if it already does not exist, otherwise opens existing file.

vi -R filename

It will create file in read only mode

Vi common commands (you must be in command mode):

Command	**Description**
I	Inserts text before current cursor
L	Inserts text at beginning of current line
A	Inserts text after current cursor
A	Inserts text at end of current line
O	Creates a new line for text entry below cursor
O	Creates a new line for text entry above cursor
X	Deletes the character under the cursor
X	Deletes the character before the cursor

dd	Deletes the line the cursor is on
cc	Removes contents of the line, leaving you in insert mode
r	Replaces the character under the cursor

Conclusion and Discussion: Thus we have studied use of vi editor and its other functions.

QUIZ / Viva Questions:

4. What is a vi editor?
5. What are operation modes of vi editor?
6. Explain commands of vi editor.

Text Books:

1. Unix, concepts and applications by Sumitabha Das, McGraw-Hill

2. Mastering Shell Scripting, Randal. K. Michael , Second Edition, Wiley Publication

References:

1. Unix Shell Programming by Yashwant Kanetkar

2. Unix shell programming by forozun

Unix Lab

Experiment No. : 3(b)

Study of File management system in Unix

Experiment No.3 (b)

Aim: Study of File management system in Unix.

What will you learn by performing this experiment?
File management commands and its theory.
Theory:

In UNIX systems user data is organised and stored in files. These files are subsequently organised into a management structure comprising directories and sub-directories. These directories and sub-directories are organised into a tree-like structure called the file system.

In UNIX, there are three basic types of files:

- **Ordinary Files:** An ordinary file actually holds the user's data or a set of program instructions.
- **Directories:** Directories store the users files in a "folder" type of structure.
- **Special Files:** Special files control access to certain types of hardware such as CD-ROM drives, Ethernet adapters. etc.

You can create directory files from your home directory and other parts of the file store structure that you have permission to write to, for example **/tmp**. When you have a number of files named in series (for example, *chap1* to *chap12*) or filenames with common characters (such as *aegis*, *aeon*, and *aerie*), you can use *wildcards* to specify many files at once. These special characters are * (asterisk), ?(question mark), and [ ] (square brackets).

* - An asterisk stands for any number of characters in a filename.

? - A question mark stands for any single character

[ ] - Square brackets can surround a choice of single characters

Some basic commands related to files in Unix

Creating Files: You can use the **vi** editor to create ordinary files on any Unix system. You simply need to give the following command −

```
$ vi filename
```

Editing Files: Once the file is opened, you can come in the edit mode by pressing the key 'i' and then you can proceed by editing the file. If you want to move here and there inside a file, then first you need to come out of the edit mode by pressing the key **Esc**. After this, you can use the following keys to move inside a file −

- **l** key to move to the right side.
- **h** key to move to the left side.
- **k** key to move upside in the file.
- **j** key to move downside in the file.

Once you are done with the editing in your file, press **Esc** and finally two keys **Shift + Z** together to come out of the file completely.

Display Content of a File: This helps us to display all details of given filename.

```
$ cat filename
```

You can display the line numbers by using the **-b** option along with the **cat** command as follows −

```
$ cat -b filename
```

Counting Words in a File: You can use the **wc** command to get a count of the total number of lines, words, and characters contained in a file.

```
$ wc filename
```

In output first column represents the total number of lines in the file, second column gives total number of words in the file, and third column represents the total number of bytes in the file. You can give multiple files and get information about those files at a time. Following is simple syntax −

```
$ wc filename1 filename2 filename3
```

Copying Files: To make a copy of a file use the **cp** command. The basic syntax of the command is −

```
$ cp source_file destination_file
```

Renaming Files: To change the name of a file, use the **mv** command. Following is the basic syntax −

```
$ mv old_file new_file
```

Deleting Files: To delete an existing file, use the **rm** command. Basic syntax:

```
$ rm filename
```

You can remove multiple files at a time with the command given below −

```
$ rm filename1 filename2 filename3
```

Standard UNIX Streams: **U**nder normal circumstances, every Unix program has three streams (files) opened for it when it starts up −

- **stdin** − This is referred to as the standard input and the associated file descriptor is 0. This is also represented as STDIN. The Unix program will read the default input from STDIN.
- **stdout** − This is referred to as the standard output and the associated file descriptor is 1. This is also represented as STDOUT. The Unix program will write the default output at STDOUT
- **stderr** − This is referred to as the standard error and the associated file descriptor is 2. This is also represented as STDERR. The Unix program will write all the error messages at STDERR.

Listing Filenames: To see the names of all your files in your current working directory ls command is used. Basic syntax is

```
$ ls
```

ls command lists the contents of a directory. It can take multiple options, some of those are explained below.

commands	**Explanation**
Ls	list a directory
ls –l	list a directory in detailed format including file sizes and permissions
ls –a	List the current directory including hidden files. Hidden files start with "."
ls -ld *	List all the file and directory names in the current directory using long format. Without the "d" option, ls would list the contents of any sub-directory of the current. With the "d" option, ls just list them like regular files.
ls –lh	list detailed format this time file sizes are human readable not in bytes

Conclusion: In this we studied how to manage files in unix operating systems.

QUIZ / Viva Questions:

1. What is meant by File management in UNIX?

2. Explain commands related to files?

Text Books:

1. Unix, concepts and applications by Sumitabha Das, McGraw-Hill

2. Mastering Shell Scripting, Randal. K. Michael, Second Edition, Wiley Publication

References:

1. Unix Shell Programming by Yashwant Kanetkar

2. Unix shell programming by forozun

Unix Lab

Experiment No. : 4

Implementation of

i)User Management

ii)Process Management

iii)Memory Management

in Unix

Experiment No.4

Aim: Perform a program for implementation of

i)User Management

ii)Process Management

iii)Memory Management in Unix

What will you learn by performing this experiment?
Performing different commands related to user, process we will learn their working and memory management in UNIX.

Theory:

User Management:

Linux is a multi-user operating system, which means that more than one user can use Linux at the same time. Linux provides a beautiful mechanism to manage users in a system. One of the most important roles of a system administrator is to manage the users and groups in a system. A user or account of a system is uniquely identified by a numerical number called the UID (unique identification number).

There are two types of users –

- The root or super user
- Normal users.

A root or super user can access all the files, while the normal user has limited access to files. A super user can add, delete and modify a user account. The full account information is stored in the */etc/passwd* file and a hash password is stored in the file */etc/shadow*.

User Permissions:

Permissions or access rights are methods that direct users on how to act on a file or directory. There are three basic access rights viz., read, write, and execute.

- *Read* – read permission allows the contents of a file to be viewed. Read permission on a directory allows you to list the contents of a directory.
- *Write* – write permission on a file allows you to modify contents of that file. Write permission allows you to edit the contents of a directory or file.
- *Execute* – for a file, execute permission allows you to run the file as an executable program or script. For a directory, the execute permission allows you to change to a different directory and make it your current working directory.

User related commands:

User creation: This commands is used to create user in system. Basic syntax is

```
$ useradd
```

After creating the user, set a password using the *passwd* utility, as follows:

```
$ useradd user1

$ passwd user1

Changing password for user user1.

New password:

Retype new password:

passwd: all authentication tokens updated successfully.
```

The system automatically assigns a UID, creates the home directory (*/home/<username>*) and sets the default shell to */bin/bash*. The *useradd* command creates a user private group whenever a new user is added to the system and names the group after the user. Specifying a user's full name when creating a user: A systems administrator can use the –c option with *useradd* to specify the user's full name, as shown below:

```
$ useradd -c "ABC XYZ" user1
```

Creating a user with the UID: You can create a user with a custom UID with the *–u* option, as follows:

```
$ useradd -u 1036 user1
```

Creating a user with non-default home directory: A non-default home directory can be set by executing the following command:

```
$ useradd –d /home/test user1
```

Adding a user to a primary group and supplementary group: A systems administrator can specify a primary group and a supplementary one by specifying the *–g* and *–G* option, respectively.

```
$ useradd -g "head" -G "faculty" user1
```

Locking and unlocking a user: A super user can lock and unlock a user account. To lock an account, one needs to invoke *passwd* with the *-l* option.

```
$ passwd -l user1

Locking password for user user1.

passwd: Success
```

The *–u* option with *passwd* unlock an account, as
shown below:

```
$ passwd -u user1

Unlocking password for user user1.

passwd: Success
```

Changing a user name: The *–l* option with the *usermod* command changes the login (user) name, as shown below:

```
$ usermod -l "user2" user1
```

Removing a user: Combining *userdel* with the *–r* option drop a user and the home directory associated with that user, as shown below:

```
$ userdel -r user2
```

Processes Management:

A process, in simple terms, is an instance of a running program. The operating system tracks processes through a five-digit ID number known as the pid or the process ID.
Each process in the system has a unique pid.

Types of Processes:

- Foreground Processes: They run on the screen and need input from the user. For example Office Programs.
- Background Processes: They run in the background and usually do not need user input. For example Antivirus.

Commands related to Process Management:

PS

This command stands for 'Process Status'. It is similar to the "Task Manager" that pop-ups in a Windows Machine when we use Cntrl+Alt+Del.

To check all the processes running under a user, use the command -

```
$ ps ux
```

You can also check the process status of a single process, use the syntax –

```
$ ps pid
```

Kill

This command terminates running processes on a Linux machine. To use these utilities you need to know the PID (process id) of the process you want to kill

```
$ kill pid
```

To find the PID of a process

```
$pidof  process _name
```

NICE

Linux can run a lot of processes at a time, which can slow down the speed of some high priority processes and result in poor performance. To avoid this, you can tell your machine to prioritize processes as per your requirements. This priority is called Niceness in Linux, and it has a value between -20 to 19. The lower the Niceness index, the higher would be a priority given to that task. The default value of all the processes is 0. To start a process with a niceness value other than the default value use the following syntax

```
$nice -n 'Nice value' process name
```

If there is some process already running on the system, then you can 'Renice' its value using syntax.

```
$renice 'nice value' -p 'PID'
```

DF

This utility reports the free disk space(Hard Disk) on all the file systems.

Free

This command shows the free and used memory (RAM) on the Linux system.

Memory Management in UNIX:

Memory is an important resource in computer. Memory management is the process of managing the computer memory which consists of primary memory and secondary memory. The goal for memory management is to keep track of which parts of memory are in use and which parts are not in use, to allocate memory to processes when they need it and de-allocate it when they are done. UNIX memory management scheme includes swapping and demand paging.

Memory Partitioning

The simplest form of memory management is splitting up the main memory into multiple logical spaces called partition. Each partition is used for separate program. There are 2 types of memory partitioning:-

Single Partition Allocation:

Single partition allocation only separates the main memory into operating system and one user process area. Operating system will not able to have virtual memory using single partition. Using single partition is very ineffective because it only allows one process to run in the memory at one time.

Multiple Partition Allocation:

Most of the operating system nowadays is using multiple partitions because it is more flexible. Multiple partition allocation enabled multiple programs run in the main memory at once. Each partition is used for one process. There are two different forms of multiple partition allocation, which is fixed partitioning and variable partitioning. Fixed partitioning divides memory up into many fixed partitions which cannot be change. However, variable partitioning is more flexible because the partitions vary dynamically in the later as processes come and go. Variable partitioning (Variable memory) has been used in UNIX. Virtual memory does more than just make your computer's memory go further. The memory management subsystem provides:

Large Address Spaces: The operating system makes the system appear as if it has a larger amount of memory than it actually has. The virtual memory can be many times larger than the physical memory in the system,

Protection: Each process in the system has its own virtual address space. These virtual address spaces are completely separate from each other and so a process running one application cannot affect another. Also, the hardware virtual memory mechanisms allow areas of memory to be protected against writing. This protects code and data from being overwritten by rogue applications.

Memory Mapping: Memory mapping is used to map image and data files into a processes address space. In memory mapping, the contents of a file are linked directly into the virtual address space of a process.

Fair Physical Memory Allocation: The memory management subsystem allows each running process in the system a fair share of the physical memory of the system,

Shared Virtual Memory: Although virtual memory allows processes to have separate (virtual) address spaces, there are times when you need processes to share memory. For example there could be several processes in the system running the bash command shell. Rather than have several copies of bash, one in each processes virtual address space, it is better to have only one copy in physical memory and all of the processes running bash share it. Dynamic libraries are another common example of executing code shared between several processes. Shared memory can also be used as an Inter Process Communication (IPC) mechanism, with two or more processes exchanging information via memory common to all of them.

Swapping:

Virtual memory is divided up into pages, chunks that are usually either 4096 or 8192 bytes in size. The memory manager considers pages to be the atomic (indivisible) unit of memory. For the best performance, we want each page to be accessible in Main memory as it is needed by the CPU. When a page is not needed, it does not matter where it is located. The collection of pages which a process is expected to use in the very near future is called its resident set. The process of moving some pages out of main memory and moving others in is called swapping. A page fault occurs when the CPU tries to access a page that is not in main memory, thus forcing the CPU to wait for the page to be swapped in. Since moving data to and from disks takes a significant amount of time, the goal of the memory manager is to minimize the number of page faults.

Conclusion: Thus we have seen how user is getting created and remove, how files are managed and about memory management also.

QUIZ / Viva Questions:

1. What is mean by User, process and memory management?

2. What are commands related to user management?

3. What is swapping?

Text Books:

1. Unix, concepts and applications by Sumitabha Das, McGraw-Hill

2. Mastering Shell Scripting, Randal. K. Michael , Second Edition, Wiley Publication

References:

1. Unix Shell Programming by Yashwant Kanetkar

2. Unix shell programming by forozun

Unix Laboratory

Experiment No. : 5(a)

Write a shell script program to display ”Hello World”

Experiment No.5(a)

Aim: Write a shell script program to display hello world.

What will you learn by performing this experiment?
Shell scripting in Unix

Software Required: Unix OS

Theory:

A shell script is a file that contains ASCII text. Many types of shells available in unix are bourne shell, kourne shell, C shell,To create a shell script, you use a text editor. A text editor is a program, like a word processor, that reads and writes ASCII text files. There are many, many text editors available for your Linux system, both for the command line environment and the GUI environment. Here is a list of some common ones: Vi,Nano editor

Create file First.sh

```
$ vi First.sh
```

Vi editor gets open. Default mode of vi editor is command mode.press “i” to change to insert mode.

Vi editor

```
#this is my first script

echo “welcome to unix lab”
```

Press **esc** to change to again command mode. Enter **:wq** to save and return to shell again.

To run the script

```
sh First.sh
```

Welcome to Unix lab is displayed

Algorithms:

1. Go to vi editor

2. Create file

3. Add content to file using echo display “Hello world”

4.Run the script

Conclusion: In this we studied different shell types of Unix and displayed shell script program of Hello World

QUIZ / Viva Questions:

1.What is shell and its types?

2.What needs to be done before you can run a shell script from the command line prompt?

References:

1.Unix concepts and applications by Sumitabha Das,Mc Graw Hill

2. Unix shell programming by Forozun

3.Unix shell programming by Yashwant kanetkar

Unix Laboratory

Experiment No. : 5(b)

Write a shell script program to check whether given number is even or odd

Experiment No. 5(b)

Aim: Write a shell script program to check whether given number is even or odd

What will you learn by performing this experiment?
Performing shell if loops

Software Required: Unix OS
Theory:

To write a shell script to find given number even or odd. BY definition a number is even if the remainder of the division of the number by 2 is zero. IF it is not zero it is called as odd. In program we shall use the modulus operator % to find the remainder of the division. As input we shall ask for number from the user. In order to check whether the remainder is zero or some-other we shall use the if else statement. The syntax of the if else statement is

```
 if[expr]

then

        //expression to be evaluated

else

        // expression to be evaluated
```

Algorithm:

1. Get the input number from the user.
2. Read n
3. Check whether the number is % by 2 gives zero
4. If n%2=0;then even ,if not odd
5. End if
6. Display the number is whether odd or even

Conclusion: In this we studied how to accept input from user and checked whether the number is odd or even

QUIZ / Viva Questions:

1.What is use of echo statement?

2.What is the syntax of if else?

3.How to get input from user in shell script?

References:

1.Unix concepts and applications by Sumitabha Das,Mc Graw Hill

2. Unix shell programming by Forozun

3.Unix shell programming by Yashwant kanetkar

Unix Lab

Experiment No. : 6

Write a shell script program to copy contents of one file to another

Experiment No.6

Aim: Write a shell script program to copy contents of one file to another

What will you learn by performing this experiment?
Performing create , read and copy file operation using shell script

Software Required: Unix OS
Theory:

Create File

The most common tool to crete a text file is text editor such as vi.Other utility such as cat to create small file.
Read
Takes the input from user through keyboard. It takes input and assigns it to a variable.
echo -n "Enter some text > "
read text
echo "You entered: $text"

Copy

The copy cp command creates a duplicate of file ,a set of files ,or directory. If the source is directory, all the files in the directory is copied to destination.
cp file 1 file2
Check for the source and destination file exist to copy contents.if not display error message file not exist. If file is exist copy the contents from source to destination

Algorithm:
1. Create a file “copying file” in vi editor.
2. Display user to “enter source file”
3. Read source file name input from user
4. Display user to “enter destination file”.
5. Read destination file name input from user

6. Check for source file exist ,if not throw error message ”file not exist”
7. Check for destination file,if not throw error message “file cannot overwrite”
8. If both exist copy contents using cp command
9. Display message copied successfully,if not display error message as “problem in copying”

Conclusion: In this we learned the concept of copying contents from one file to another file using shell scrip in Unix

QUIZ / Viva Questions:

1. What command is used to get input from user?

2. What is the default mode of vi editor?

3. What is the command to copy contents ?

References:

1. Unix concepts and applications by Sumitabha Das,Mc Graw Hill

2. Unix shell programming by Forozun

3. Unix shell programming by Yashwant kanetkar

Unix Lab

Experiment No. : 7(a)

Write a program using sed command to print duplicate lines of input

Experiment No. 7(a)

Aim: Write a program using sed command to print duplicate lines of input

What will you learn by performing this experiment?
Stream editor (sed) concepts in Unix

Software Required: Unix OS
Theory:

Sed is a Stream Editor used for modifying the files in unix (or linux). Whenever you want to make changes to the file automatically, sed comes in handy to do this. Most people never learn its power; they just simply use sed to replace text. You can do many things apart from replacing text with sed.

Consider the below text file as an input.

```
$cat > geekfile.txt
unix is great os. unix is opensource. unix is free os.
learn operating system.
```

Replacing or substituting string:
Sed command is mostly used to replace the text in a file. The below simple sed command replaces the word “unix” with “linux” in the file.

```
$sed 's/unix/linux/' geekfile.txtOutput:
```

```
unix is great os. linux is opensource. unix is free os.
learn operating system.
```

Replacing all the occurrence of the pattern in a line :

The substitute flag /g (global replacement) specifies the sed command to replace all the occurrences of the string in the line.

```
$sed 's/unix/linux/g' geekfile.txtOutput :
```

```
linux is great os. linux is opensource. linux is free os.
learn operating system.
```

Duplicating the replaced line with /p flag

The /p print flag prints the replaced line twice on the terminal. If a line does not have the search pattern and is not replaced, then the /p prints that line only once.

```
>sed 's/unix/linux/p' file.txt
linux is great os. unix is open source. unix is free os.
linux is great os. unix is open source. unix is free os.
learn operating system.
Linux linux which one you choose.
Linux linux which one you choose
```

Algorithm:

1. Create a text file file1.txt

2. Add contents to the file using **cat** command

3. Replace and substitute string with /p flag of sed command

4. Duplicate lines gets print in the file

Conclusion : In this we studied Stream editor and sed command to print duplicate lines in a file

QUIZ / Viva Questions:

1. What is sed?
2. What are the different features of sed with examples?

References:

1. Unix concepts and applications by Sumitabha Das,Mc Graw Hill

2. Unix shell programming by Forozun

3. Unix shell programming by Yashwant kanetkar

Unix Lab

Experiment No. : 7(b)

Use a pipeline and command substitution to set the length of a line in file to a variable

Experiment No. 7(b)

Aim: Use a pipeline and command substitution to set the length of a line in file to a variable using shell script.

What will you learn by performing this experiment?
Pipelining and shell scripting
Software Required: Unix OS

Theory:
Pipe command:
The Pipe is a command in Linux that lets you use two or more commands such that output of one command serves as input to the next. In short, the output of each process directly as input to the next one like a pipeline. The symbol '|' denotes a pipe.
fold command :
The fold command in UNIX is a command line utility for folding contents of specified files, or standard input. By default it wraps lines at a maximum width of 80 columns. It also supports specifying the column width and wrapping by numbers of bytes.

To limit column width:

To change the column width from the default 80 pass the -w option. In the following example this is limited to 20
Output:

```
fold -w 20 lorum.txt
Lorem ipsum dolor si
t amet, consectetur
adipiscing elit. Int
eger at accumsan ips
um, ut sagittis dolo
r. Vivamus erat tell
us, ullamcorper ut a
liquam nec, maximus
at turpis.
```

Algorithm:

1. Assign the length of a file to a variable
2 .Using fold command set the length of the file
3. Use pipe command to give input of variable value to fold command
4. Use Command substitution to set length of file

Conclusion: Using a pipeline and command substitution to set the length of a line in file to a variable using shell script is executed successfully

QUIZ / Viva Questions:

1. What is the use of pipe command ?
2. Which command provides to set length of a line in file?
3. What is the use of command substitution**?**

References:

1. Unix concepts and applications by Sumitabha Das,Mc Graw Hill

2. Unix shell programming by Forozun

3. Unix shell programming by Yashwant kanetkar

Unix Lab

Experiment No. : 8

Write a grep/egrep program to find number of words character, words, line in file

Experiment No- 8

Aim: Write a grep/egrep program to find number of words character, words, line in file

What will you learn by performing this experiment?
Performing grep/egrep commands in shell script

Software Required: Unix OS

Theory:

Grep stands for global regular expression print. It is used to search the input file for all lines that match a specified regular expression and write them to standard output file.

Grep operations:

1. Copies the next input line into the pattern space
2. Applies the regular expression to the pattern space
3. If there is a match,line is copied from pattern space to standard output.

Grep options:

Options	**Descriptions**
B	Display the block number at the beginning of each line.
-c	Display the number of matched lines.
-h	Display the matched lines, but do not display the filenames.
-i	Ignore case sensitivity.
-l	Display the filenames, but do not display the matched lines.
-n	Display the matched lines and their line numbers.
-s	Silent mode.

Algorithm:

1. Enter the file name

2. Read fine name as input from user

3. Fetch number of words in a file using wc-w and assign to a variable

4. Fetch number of characters in a file using wc-c and assign to a variable

5. Using grep-c option and get count of the number of lines

6. Display all the count of words ,character and number of lines in a file

Conclusion: Thus we have studied the use of grep commands and its options

QUIZ/Viva Questions:

1. What is grep?
2. What are the grep types?
3. Which option in grep to display ignores case in matching text?

References:

1. Unix concepts and applications by Sumitabha Das,Mc Graw Hill

2. Unix shell programming by Forozun

3. Unix shell programming by Yashwant kanetkar

Unix Lab

Experiment No. : 9

Write a perl script to check number is prime or not

Experiment No.9

Aim: Write a perl script to check number is prime or not.

What will you learn by performing this experiment?
How to work with Perl Script.
Software Required: Ubuntu OS

Theory:

Perl is a programming language developed by Larry Wall, especially designed for text processing. It stands for Practical Extraction and Report Language. Perl is a general-purpose programming language originally developed for text manipulation and now used for a wide range of tasks including system administration, web development, network programming, GUI development, and more.

Perl is a stable, cross platform programming language. Though Perl is not officially an acronym but few people used it as Practical Extraction and Report Language. It is used for mission critical projects in the public and private sectors. Perl is an *Open Source* software, licensed under its *Artistic License*, or the *GNU General Public License (GPL)*.

Perl takes the best features from other languages, such as C, awk, sed, sh, and BASIC, among others. Perls database integration interface DBI supports third-party databases including Oracle, Sybase, Postgres, MySQL and others. Perl works with HTML, XML, and other mark-up languages. Perl supports Unicode. Perl supports both procedural and object-oriented programming. Perl interfaces with external C/C++ libraries through XS or SWIG. Perl is an interpreted language, which means that your code can be run as is, without a compilation stage that creates a non portable executable program. Traditional compilers convert programs into machine language. When you run a Perl program, it's first compiled into a byte code, which is then converted (as the program runs) into machine instructions. So it is not quite the same as shells, or Tcl, which are strictly interpreted without an intermediate representation.

Perl Environment:

Before we start writing our Perl programs, let's understand how to setup our Perl environment. Perl is available on a wide variety of platforms Unix (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX etc.), Win 9x/NT/2000, Macintosh (PPC, 68K), Solaris (x86, SPARC), OpenVMS and many more...

This is more likely that your system will have perl installed on it. Just try giving the following command at the $ prompt –

```
$perl –v
```

If you have perl installed on your machine, then you will get a message something as

This is perl and its version, sub-version and design for what and other details.

Otherwise install it from https://www.perl.org/get.html.

Running Perl

There are two ways in which you can write code.

- **Interactive Interpreter:**

You can enter perl and start coding right away in the interactive interpreter by starting it from the command line. You can do this from Unix, DOS, or any other system, which provides you a command-line interpreter or shell window.

```
$perl  -e <perl code>
```

- **Script from the Command-line:**

A Perl script is a text file, which keeps perl code in it and it can be executed at the command line by invoking the interpreter on your application, as in the following −

```
$perl  script.pl
```

As a Perl convention, a Perl file must be saved with a .pl or .PL file extension.

First Perl Program

1. **Interactive Mode Programming:**

You can use Perl interpreter with -e option at command line, which lets you execute Perl statements from the command line. Let's try something at $ prompt as follows −

```
$perl -e 'print "Hello World\n"'
```

This execution will produce the following result −

```
Hello, world
```

2. **Script Mode Programming**

Assuming you are already on $ prompt, let's open a text file hello.pl using vi or vim editor and put the following lines inside your file.

```
#!/usr/bin/perl

# This will print "Hello, World"

print "Hello, world\n";
```

Here /usr/bin/perl is actual the perl interpreter binary. Before you execute your script, be sure to change the mode of the script file and give execution priviledge, generally a setting of 0755 works perfectly and finally you execute the above script as follows −

```
$chmod 0755 hello.pl

$./hello.pl
```

This execution will produce the following result −

```
Hello, world
```

Now we have to use perl for checking number is prime or not.

Algorithm :

```
Step 1: Start
Step 2: Read the number say n.
Step 3: Initialize flag to 0.
Step 4: if n is equal to 2, print number is prime and goto 8
Step 5: Apply loop to check prime number. Loop variable start from 2 to n-1
Step 6: if n%loop_variable=0, set flag=1
Step 7: if flag=1, break from loop n print not prime, else print prime.
Step 8: Stop
```

Conclusion and Discussion:

In this we learned the concept of perl and we have seen how to write a perl program for prime number.

QUIZ / Viva Questions:

What is Perl?

What are ways of writing perl programs?

What is condition of prime number?

Text Books:

1. Unix, concepts and applications by Sumitabha Das, McGraw-Hill

2. Mastering Shell Scripting, Randal. K. Michael , Second Edition, Wiley Publication

References:

1. Unix Shell Programming by Yashwant Kanetkar

2. Unix shell programming by forozun

Unix Lab

Experiment No. : 10

Write a awk script to develop a Fibonacci series

Experiment No. 10

Aim: Write an awk script to develop a Fibonacci series.

What will you learn by performing this experiment?
We will learn to use **awk** scripting language by using awk command

Software Required: Ubuntu

Theory:
Awk

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 matches with the specified patterns and then performs the associated actions. Simple awk commands can be run from the command line. More complex tasks should be written as awk programs (so-called awk scripts) to a file.

The basic format of an awk command looks like this:

```
awk 'BEGIN {start_action} {action} END {stop_action}' filename
```

Here the actions in the begin block are performed before processing the file and the actions in the end block are performed after processing the file. The rest of the actions are performed while processing the file.
Create a file input_file with the following data. This file can be easily created using the output of ls -l.

```
-rw-r--r-- 1 center center  0 Dec  8 21:39 p1
-rw-r--r-- 1 center center 17 Dec  8 21:15 t1
-rw-r--r-- 1 center center 26 Dec  8 21:38 t2
```

```
-rw-r--r-- 1 center center 25 Dec  8 21:38 t3
-rw-r--r-- 1 center center 43 Dec  8 21:39 t4
```

From the data, you can observe that this file has rows and columns. The rows are separated by a new line character and the columns are separated by a space characters. We will use this file as the input for the examples discussed here.

1. awk '{print $1}' input_file

Here $1 has a meaning. $1, $2, $3, represents the first, second, third columns and so on in a row respectively. This awk command will print the first column in each row as shown below.

```
-rw-r--r--
-rw-r--r--
-rw-r--r--
-rw-r--r--
-rw-r--r--
-rw-r--r—
```

To print the 4th and 6th columns in a file use awk '{print $4,$5}' input_file

Here the Begin and End blocks are not used in awk. So, the print command will be executed for each row it reads from the file. In the next example we will see how to use the Begin and End blocks.

2. awk 'BEGIN {sum=0} {sum=sum+$5} END {print sum}' input_file

This will prints the sum of the value in the 5th column. In the Begin block the variable sum is assigned with value 0. In the next block the value of 5th column is added to the sum variable. This addition of the 5th column to the sum variable repeats for every row it processed. When all the rows are processed the sum variable will hold the sum of the values in the 5th column. This value is printed in the End block.

3. In this example we will see how to execute the awk script written in a file. Create a file sum_column and write the below script in that file

```
#!/usr/bin/awk -f
BEGIN {sum=0}
{sum=sum+$5}
END {print sum}
```

Now execute the script using awk command as

```
awk -f sum_column input_file.
```

This will run the script in sum_column file and displays the sum of the 5th column in the input_file.

4. To print the squares of first numbers from 1 to n say 6:

```
$ awk 'BEGIN { for(i=1;i<=5;i++) print "square of", i, "is",i*i; }'
```

Output:-

```
square of 1 is 1
square of 2 is 4
square of 3 is 9
square of 4 is 16
square of 5 is 25
square of 6 is 36
```

ALGORITHM

```
1. Create a file using awk –f
2. Declare variables a,b
   Declare a function f (n)
       {
           Print a;
            c=a+b;
            a=b;
            b=c;
       }
3. Using for-loop satisfy the condition as for(i=0;i<`$ {NUM}`;i++)
4. Execute the script written by using awk command.
```

Conclusion: We have studied how to use awk script by using awk command as well as by writing awk script in a file.

QUIZ/Viva Questions:

1. What is awk?
2. What is basic format of awk?

Text Books:

1. Unix, concepts and applications by Sumitabha Das, McGraw-Hill

2. Mastering Shell Scripting, Randal. K. Michael , Second Edition, Wiley Publication

References:

1. Unix Shell Programming by Yashwant Kanetkar

2. Unix shell programming by forozun