EXPERIMENT - 1
Basic Linux commands

1.a) INTRODUCTION TO LINUX/UNIX

AIM:
To study the basics of UNIX/LINUX.

UNIX:
In 1969, AT & T Bell Industries, USA developed UNIXand is a multi-user operating system.
Ken Thomson and Dennis Ritchie developed UNIX OSfeatures influenced from MULTICS (Multiplexed Information and Computing Service) OS. The development of UNIX operating systemstarted in 1969, and its code was rewritten in C in 1972.

LINUX:
Linux is an open-source software and is similar to UNIX, which was created by Linus Torualds. All UNIX commands worksin Linux. The features of Linux issimilar to other OS such as Windows and UNIX.

STRUCTURE OF A LINUX OPERATING SYSTEM:
The Linux operating system consists of three parts.
a) The UNIX Kernel
b) The Shell
c) The Programs - Tools and Applications

UNIX KERNEL:
Kernel is the core of the UNIX OS and is the lowest layer. It allocates processor time and memory to each program and determines when each program will run.Kernel is responsible for the control of all the tasks, schedules all the processes and carries out all the functions of the OS. Kernel decides when to stop an executing program and startanother program.

SHELL:
Shell is the command interpreter in the UNIX OS.The shell acts as an interface between the user (programs) and the kernel. It accepts commands from the user and analyses and interprets them.

UNIX is a multi-user, multi-tasking operating system in which a user can run more than one program or task at a time. The responsibility of kernel is to keep each process and user separate and to regulate access to system hardware, including CPU, memory, disk and other I/O devices.

UNIX is a layered operating system. – Layers are Hardware, Kernel and System calls, Programs
- **Hardware layer**is the innermost layer which provides the services for the operating system.
- The operating system, such as UNIX has two parts – **Kernel** and **Shell**

- **Kernel** interacts directly with the hardware resourcesand offers the services to the user programs.
- **User programs** interact with the kernel through a set of standard **system calls.** These system calls request services to be provided by the kernel.
- The services may be accessing a file: open, close, read, write, link, or execute a file; starting or updating accounting records; changing ownership of a file or directory; altering files to a new directory; creating, suspending, or killing a process; enabling access to hardware devices; and setting limits on system resources.

HISTORY OF UNIX

Year	Developments
Mid of 1960's (1965)	Multiplexed Operating and Computing System (MULTICS) is the system in which the Unix operating system was used at the beginning. The MULTICS project began as a joint venture of General Electric, Massachusetts Institute for Technology and AT&TBell Laboratories.
1969	Bell Laboratories pulled out of the project in 1969.In 1969, Ken Thompson wrote the first version of Unix, called UNICS. UNICS stands for Uniplexed Operating and Computing System.
1970	The development of UNIX rises with the acquisition of a DEC (Digital Equipment Corporation) PDP-11computer. PDP-11computer has 24 kilobytes of RAM and 512 kilobytes of disk space.
1971	Sixth version of UNIX was used only in Bell Labs.UNIX was used (with the assembly-language-coded *troff*) in the Bell Labs patent department as one of the first document-processing programs.
1973	In 1973 Dennis Ritchie and Ken Thompson rewrote the Unix kernel in C. A year later,the Fifth Edition of UNIX was first licensed to universities.
1978	The Seventh Edition of UNIX, released in 1978, served as a dividing point for two divergent lines of Unix development. The two branches are known as BSD andSVR4 (System V).

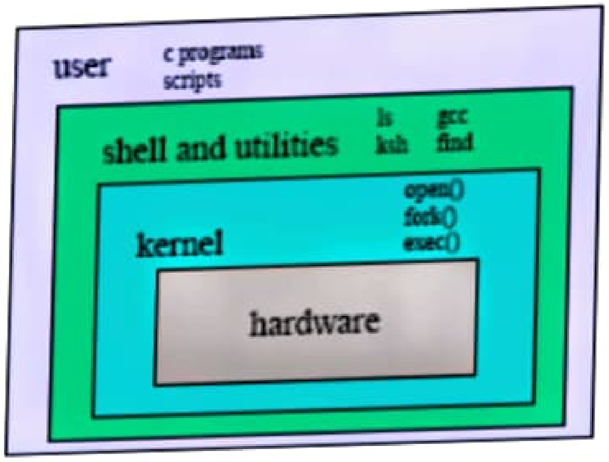
FEATURES OF UNIX Operating System

The UNIX operating system is the most popular operating system. The key features that made the UNIX operating system popularare:

- Multi-user operating system
- Hierarchical file system
- Time sharing
- Background processing
- Multitasking
- Security and Protection
- Better communication
- Includescompilers, tools and utilities

- Portability
- Shell programming

THE KERNEL AND THE SHELL



Kernelis the main control program of UNIX operating system. The kernel does not allow the user to execute its commands directly; instead, the user need to type commands on another program in the operating system called a **shell. Shell program** parses the commands, checks the syntax, translates and passes them to the kernel for execution.

The kernel will send the response when the command has been correctly interpreted and executed.The response may merely be a **prompt** for the entering next command, or displays output in the monitor. If the kernel cannot execute the requested command due to wrong syntax, then the response will be an error message; and the user must re-enter the command with correct syntax.

Commonly used Unix operating systemshells are:

- The Bourne Shell or POSIX shell – /bin/shis a symlink to a Bourne shell.
- C shell – /bin/cshis a symlink to the C shell, or a compatible shell
- Enhanced C Shell – /bin/tcsh - Tcsh is an enhanced version of the csh
- Bash shell – /bin/bash - Bash shell interpreter is located.BourneAgain shell (called bash)
- Korn shell – /bin/ksh - Korn shell offers much more programming features and is superior in comparison to the Bash shell.

Files and processes

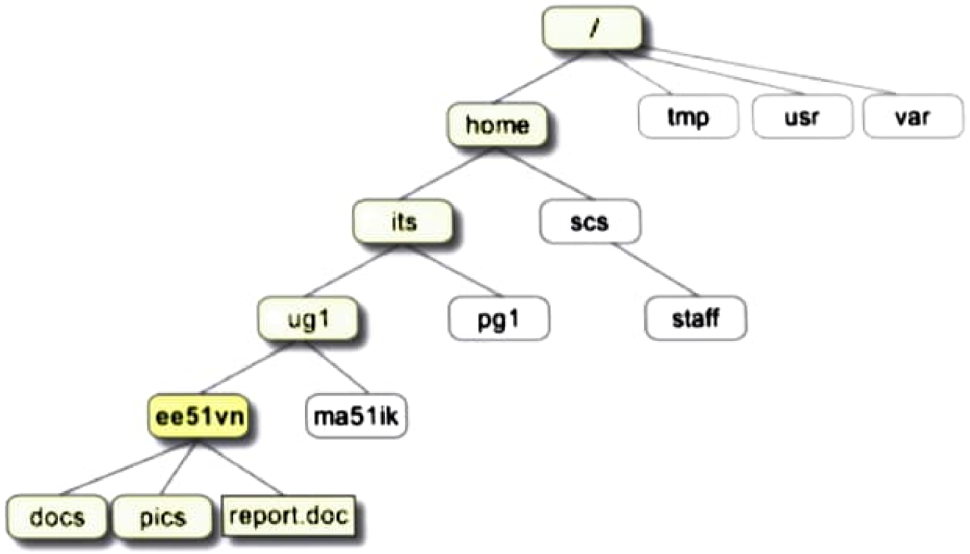
In UNIX operating system, everythingis represented or stored either as a file or as a process.

- A **file** is a collection of data. Files are created by users using text editors, running compilers etc.
    - Examples of files are:
    - a typed document (report, essay etc.) prepared by word or text editors.
    - the text of a program written in some high-level programming language
    - instructions comprehensible directly to the machine and incomprehensible to a casual user, for example, a collection of binary digits (an executable or binary file);
    - a directory, containing information about its contents, which may be a mixture of other directories (subdirectories) and ordinary files.

- A **process** is an program under execution which can be identified by a unique PID. PID stands for process identifier.

The Directory Structure

In UNIX like operating system, all the files are usually organized in the directory structure. The UNIX filesystem is represented as a hierarchical structure, similarto an inverted tree. The top of the file system hierarchy is called **root** (written as a "/")



In the diagram above, the directory of the undergraduate student **"ee51vn"** contains two sub-directories (**docs** and **pics**) and a file called **report.doc**.

The full path to the file **report.doc** is **"/home/its/ug1/ee51vn/report.doc"**

The current directory (.)

In UNIX-like OS, (**.**) means the current directory, so typing
$cd .
NOTE: there is a space between cd and the dot

The parent directory (..)

In UNIX-like OS,(**..**) means the parent of the current directory, hence
$cd ..
will change one directory up the hierarchy (back to the home directory).
Note: typing cd with no argument always returns to the home directory. It is very useful when you are lost in the file system.

A **Shell** provides an interface to the Unix. It gathers input from the user and executes programs based on the user input. When a program finishes itsexecution, it displays that output of the program.

Shell provides an environment in which a user can run commands, execute programs and shell scripts..

Shell Prompt

The prompt, **$**, which is called the **command prompt**, is issued by the shell. While the prompt is displayed, you can type a command.
Shell reads your input after you press **Enter**. It determines the command you want executed by looking at the first word of your input. A word is an unbroken set of characters. Spaces and tabs separate words.
Following is a simple example of the **date** command, which displays the current date and time:

```
$date
Thu Jun 25 08:30:19 MST 2009
```

Shell Types

In Unix, there are two major types of shells −

- **Bourne shell** − If you are using a Bourne-type shell, the **$** character is the default prompt.
- **C shell** − If you are using a C-type shell, the % character is the default prompt.

The Bourne Shell has the following subcategories −

- Bourne shell (sh)
- Korn shell (ksh)
- Bourne Again shell (bash)
- POSIX shell (sh)

The different C-type shells follow −

- C shell (csh)
- TENEX/TOPS C shell (tcsh)

The original Unix shell was written in the mid-1970s by Stephen R. Bourne while he was at the AT&T Bell Labs in New Jersey.

Bourne shell was the first shell to appear on Unix systems, thus it is referred to as "the shell".
Bourne shell is usually installed as **/bin/sh**on most versions of Unix. For this reason, it is the shell of choice for writing scripts that can be used on different versions of Unix.

Shell Scripts

The basic concept of a shell script is a list of commands, which are listed in the order of execution. A good shell script will have comments, preceded by # sign, explaining the steps.

Shell scriptmay contain conditional tests, such as value X is greater than value Y, loops need to go through massive amount of data, files to read and store data, and variables to read and store data, and the shell script may include functions.

In Unix-like operating systems such as Linux, the **chmod**command is used to change the access mode of a file. The command **chmod**stands for**change mode**.

Syntax of chmod:

chmod [reference][operator][mode] file

The **reference** option is used to represent the users to whom the permissions are given. Itwill be either a single letter or a list of letters which specifies whomhas the permissions. The references can be represented by one or more of the following letters:

Reference	Class	Description
u	owner	Permission is granted only to the owner of the file.
g	group	Permission granted to users who are members of the file's group.
o	others	users who are neither the file's owner nor members of the file's group
a	all	All three of the above, i.e., a is same as ugo

The **operator** is used to specify how the permission modes of a file must be adjusted. The three operators are as follows:

Operator Description

+	Adds the specified permission modes to the specified classes.
-	Removes the specified permission modes from the specified classes
=	Themodes specified are to be made the exact modes for the specified classes

The **modes** indicate which permissions are to be granted or removed from the specified classes. There are three basic modes which correspond to the basic permissions:

r	Permission to read accessto the specified file.
w	Permission to write access to the specified file. Also allows to delete the file.
x	Permission to execute the specified file, or, if it's a directory, then permission to search it.

Types of permissions can be changed using chmodcommand:
In Linux, ls-l command is used to list all the permissions assigned to different files. Open terminal and type ls-l command, it will list the files in the current working directory (in long format).

1.b) Basic UNIX/LINUX Commands

AIM:
To study basic UNIX commands.

THEORY:

a) date
–It is used to display the date and time.

Syntax: $date

Format	Purpose	Example	Result
+%m	Displays monthonly (in number)	$date+%m	06
+%h	Displays month name	$date+%h	June
+%d	Displays day of month	$date+%d	01
+%y	Print last two digits of year	$date+%y	09
+%H	Print hours	$date+%H	10
+%M	Displays minutes	$date+%M	45
+%S	Displays seconds	$date+%S	55

b) cal
-This command is used to show the calendar.

Syntax: $cal 2 2009

c) echo
- This command is used to print a message on the screen.

Syntax: $echo "text"

d) ls
- This command is used to list the files and directories of the current working directory.

Syntax: $ls
Lists all the files and directories

Command	**Usage**
ls -s (the s is lowercase)	to list files and directories with their sizes
ls -S (the S is uppercase)	to list files and directories in descending order of file size (biggest to smallest).
ls–l	Displays the list in a long list format. (Provide file statistics)
ls–t	Displays the list of files in the order of its creation time.
ls– u	Sort by access time (or show when last accessed together with –l)
ls–r	Displays the list in reverse order
ls–f	Mark directories with /, executable with* , symbolic links with @, local sockets with =,named pipes (FIFOs)with
ls– h	Displays size of the file in human readable format.
ls[a-m]*	to list all the files with filename starts with alphabets "a" to "m".
ls[a]*	to list all the files whose name begins with "a" or "A"
Eg:$ls>filelist	Output of"ls" command is written or stored to a file named "filelist".

e) lp
- This command submits a file for printing.

Syntax: $lp filename

f) man
- This command is used to provide manual help for every UNIX command.

Syntax: $man unix command

$man cat

g) who &whoami
- This command displays the data about all the users who have currently logged into the system.

whoami -This command displays the current user only.

Syntax: $who
$whoami

h) uptime
- This command tells you how long the computer has been running since its last reboot or power-off.

Syntax: $uptime

i) uname

- This commandprints the system information such as system name,processor,hardware platform and OS type.

Syntax: $uname–a

j) hostname

- This command displays and set system host name.

Syntax: $hostname

k) bc

–This command stands for "**basic calculator**"

$bc 10/2*3 15	$ bc scale =1 2.25+1 3.35 quit	$ bc ibase=2 obase=16 11010011 89275 1010 Ā quit	$ bc sqrt(196) 14 quit

$bc
for(i=1;i<3;i=i+1)I
1
2
3 quit
	$ bc-l
	scale=2
	s(3.14)
	0		

FILE MANIPULATION COMMANDS

a) cat – This command can be used to create, view and concatenate files.

Creation:

Syntax:

$cat>filename

Viewing:

Syntax:

$cat filename

Add text to an existing file:

Syntax:

$cat>>filename

Concatenate:

Syntax:

$cat f1 f2>f3

$cat f1 f2>>f3 (f1, f2 and f3 are three filenames and no over writing of file f3)

b) grep – This command is used to search a particular word or pattern from a file.
Syntax: $grep search word filename
Eg: $grep anu student

c) rm – This command removesthe specified file from the file system.
Syntax: $rm filename

d) touch – This command is used to create a blank file.
Syntax: $touch filename

e) cp – This command copies the files or directories
Syntax: $cp sourcefiledestinationfile
Eg: $cp student stud

f) mv – This command is used to rename the file or directory.
Syntax: $mv old file new file
Eg: $mv–i student studentlist(-i prompt when overwrite)

g) cut– This command cuts or picks up a given number of characters or fields of the file.
Syntax:$cut<option><filename>
Eg: $cut –c filename
$cut–c1-10 student
$cut–f 3, 5 student
$ cut –f 3-8student
-c cutting columns
-f cutting fields

h) head – This command displays 10 lines from the top (head) of the file.
Syntax: $head filename
Eg:$head employee
Command to print the top two lines:
Syntax: $head-2 employee

i) tail – This command displays last 10 lines of the file
Syntax: $tail filename
Eg:$tail student

Command to print the bottom two lines;
Syntax: $tail -2 student

j) chmod – This command is used to change the permission of a file or directory.
Syntax: $chmod category operation permission file

category–It specifies the user type.
operation– It is used to assign or remove permission
Permission– It specifies the type of permission – read, write or execute
File– It specifies the filename to assign or remove permission all

Examples:
$chmod u-wx student
This command removes permission to write and execute for users on student.
$chmodu+rw,g+rw student
This command assigns read and write permission for users and groups
$chmod g=rwx student
This command assigns or changesthe permission for groups and grant read, write and execute permissions.

k) wc – This commandfinds the number of lines, word count, character count in a specified file or files. The wc stands for word count and the options of **wc command**are :–l,-w,-c
wc command displays four-columnar output.

Let file1.txt be a file contains names of fiveIndian states.
$cat file1.txt
Kerala
Assam
Arunachal Pradesh
Madhya Pradesh
Tamil Nadu

wc file1.txt
5 853file1.txt
First column - number of lines, second column - number of words, third column - number of characters present in file, fourth column - file name itself.

Category		Operation		Permission	
u	users	+	assign	r	read
g	group	-	remove	w	write
o	others	=	assign absolutely	x	execute

Syntax:
$wc –l filename
$wc –w filename
$wc–c filename

l)touch : Create a new empty file.

Syntax: touch <filename>

15/83

1.c) UNIX EDITORS

AIM:
To study various UNIX editors such as vi, ed, ex and EMACS.

THEORY:
Editor is a program that allows user to see a portion a file on the screen and modify characters and lines by simply typing at the current position. UNIX supports a variety of Editors. They are: - ed ex vi

EMACS
Vi - vi stands for "visual". vi is the most important and powerful editor. vi is a full screen editor that allows user to view and edit entire document at the same time.vi editor was written in the University of California, at Berkley by Bill Joy, who is one of the co-founder of Sun Microsystems.

Features of vi:
It is easy to learn and has more powerful features.
It works great speed and is case sensitive. vi has powerful undo functions and has 3 modes:

1) Command mode
2) Insert mode
3) Escape or ex mode

In command mode, no text is displayed on the screen.
In Insert mode, it permits users to edit, insert or replace text.
In escape mode, it displays commands at the command line.
Moving the cursor with the help of h, l, k, j, I, etc

EMACS Editor

Cursor movements / EMACS Commands:
C- means "control key", M- means "meta key"

M->	Move to end of file
M-<	Move to beginning of file
M-f	Move cursor forward one word
M-b	Move cursor backwards one word
M-v	Go backward one screen
C-v	Go forward one screen
C-n	Move cursor to next line
C-p	Move cursor to previous line
C-a	Move cursor to beginning of line
C-e	Move cursor to end of line

C-f	Move cursor forward one character
C-b	Move cursor backwards one character
C-x C-s	Save file

Deletion Commands:

C-d	Delete the character by the cursor.
delete key	Delete the character preceding the cursor
C-x DEL	deletes the previous sentence
M-k	delete the rest of the current sentence
C-k	Delete from the cursor to the end of the line
C-x u	Undo (in order to undo several previous commands)

Search and Replace in EMACS:

y	To replace the occurrence of the pattern with new string
n	Don't change the occurrence, but to skip to the next occurrence without replacing this one.
!	to replace all remaining occurrences of the file without asking again.

17/83

RESULT:

The Linux commands are executed and the output is verified.

EXPERIMENT - 2
Shell programming

AIM:

To write simple shell programs by using conditional, branching and looping statements.

a) EVEN OR ODD
b) LEAP YEAR OR NOT
c) BIGGEST OF THREE NUMBERS
d) FACTORIAL OF NUMBER
e) FIBONACCI SERIES
f) SWAP TWO VARIABLES

2.a) Write a Shell program to check if the given number is even or odd.

ALGORITHM:

```
1) Start the program
2) Read the number, n.
3) Calculate "r=expr $n%2".
4) If the value of r equals 0 then print the number is even.
5) If the value of r not equal to 0 then print the number is odd.
6) Stop the program.
```

PROGRAM:

```
echo "Enter the Number"
read n
r=`expr $n % 2`
if [ $r -eq 0 ]
then
echo "$n is Even number"
else
echo "$n is Odd number"
fi
```

OUTPUT:

```
Enter the Number
4
4 is Even number
Enter the Number
5
5 is Odd number
```

2.b) Write a Shell program to check if the given year is leap year or not.

ALGORITHM:

1) Start the program.
2) Read year in y.
3) Calculate "x=expr $y%4".
4) If the value of x equals 0 then print the year is a leap year.
5) If the value of x not equal to 0 then print the year is not a leap year.
6) Stop the program.

PROGRAM:

```
echo "Enter the year"
read y
x=`expr $y % 4`
if [ $x -eq 0 ]
then
echo "$y is a leap year"
else
echo "$y is not a leap year"
fi
```

OUTPUT:

```
Enter the year
2004
2004 is a leap year
Enter the year
2005
2005 is not a leap year
```

2.c) Write a Program to Find Biggest in Three Numbers.

ALGORITHM:

1) Start the program.
2) Read three numbers.
3) If xis greater than yand xis greater than zthen print "x"is Big.
4) Else If "y" is greater than "z"then "z"is Big.
5) Else print "z"is Big.
6) Stop the program.

PROGRAM:

```
echo "Enter three numbers""
read x y z
if [ $x -gt $y ]&& [ $x -gt $z ]
then
```

```
echo "x is big"
else if [ $y -gt $z ]
then
echo "y is big"
else
echo "z is big"
fi
fi
```

OUTPUT:

```
Enter three numbers
23 54 78
zis big.
```

2.d) Write a Shell program to find the factorial of a number.

ALGORITHM:

1) Start the program.
2) Read the number, n.
3) Calculate "i=expr $n-1".
4) If the value of i is greater than 1 then calculate "n=expr $n * $i" and "i=expr $i – 1"
5) Print the factorial of the given number.
6) End program

PROGRAM:

```
echo -n "Enter a Number"
read n
i=`expr $n - 1`
while [ $i -ge1 ]
do
n=`expr $n \* $i`
i=`expr $i - 1`
done
echo "The Factorial of the given Number is $n"
```

OUTPUT:

```
Enter a Number5
The Factorial of the given Number is 120
```

2.e) Write a Shell program to print the Fibonacci series.

ALGORITHM:

1) Start the program.
2) Read the Limit, N.

```
3) Print First two terms of fibonacci series - a and  b.
4) For loop from i=0 to i<N
    4.1) Print $a
    4.2) fn = a+b
    4.3) a=$b
    4.4) b=$fn
    4.5) Increment i by 1.
5) Print the Fibonacci seriesupto the given limit.
6) Stop the program.
```

PROGRAM:

```
# Limit N
N=7
# First number of the Fibonacci Series
a=0
# Second number of the Fibonacci Series
b=1
echo "The Fibonacci series is :"
for (( i=0; i<N; i++ ))
do
  echo -n "$a "
  fn=$((a + b))
  a=$b
  b=$fn
done
```

OUTPUT:

```
The Fibonacci series is :
0 1 1 2 3 5 8
```

2.f) Write a Shell program to swap the two integers.

ALGORITHM:

```
1) Start the program.
2) Read two integers a and b.
3) Calculate the swapping of two integers by using a temporary variable temp.
4) Print the swapped values of a and b.
5) Stop the program
```

PROGRAM:

```
echo "Enter two numbers"
read a b
temp=$a
a=$b
```

```
b=$temp
echo "after swapping"
echo $a $b
```

OUTPUT:

```
Enter Two Numbers
4 5
after swapping
5 4
```

RESULT:

Thus, the programshave been executed successfully.