DEPARTMENT OF INFORMATION TECHNOLOGY

Manual Ref. No:- PCE/IT/2020-21/4IT4-21

Session 2020-21

Name of the Lab: - Linux Shell Programming Lab

Year / Sem: <II/IV>

Lab Code: - 4IT4-21

Name of the Faculty:- Amol Saxena

Verified & Checked by:
Mr. Amol Saxena
(HOD-IT, PCE)

Approved by:
Dr. Mahesh M. Bundele
Principal & Director, PCE

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

TABLE OF CONTENTS

S. No.	Contents	Page No.
1	Lab Rules	3
2	Instructions	5
3	Syllabus	6
4	Marks Scheme	7
5	Lab Plan	8
6	Introduction to the subject (Zero Lab)	11
7	Experiment-1	15
8	Experiment-2	19
9	Experiment-3	24
10	Experiment-4	26
11	Experiment-5	30
12	Experiment-6	34
13	Experiment-7	35
14	Experiment-8	35
15	Experiment-9	37
16	Experiment-10	39
17	Experiment-11	40

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

LAB RULES

Responsibilities of Users
Users are expected to follow some fairly obvious rules of conduct:

Do's

- Enter the lab on time and leave at proper time.
- Wait for the previous class to leave before the next class enters.
- Keep the bag outside in the respective racks.
- Utilize lab hours in the corresponding.
- Turn off the machine before leaving the lab unless a member of lab staff has specifically told you not to do so.
- Leave the labs at least as nice as you found them.
- If you notice a problem with a piece of equipment (e.g. a computer doesn't respond) or the room in general (e.g. cooling, heating, lighting) please report it to lab staff immediately. Do not attempt to fix the problem yourself.

Don'ts

- Don't abuse the equipment.
- Do not adjust the heat or air conditioners. If you feel the temperature is not properly set, inform lab staff; we will attempt to maintain a balance that is healthy for people and machines.
- Do not attempt to reboot a computer. Report problems to lab staff.
- Do not remove or modify any software or file without permission.
- Do not remove printers and machines from the network without being explicitly told to do so by lab staff.
- Don't monopolize equipment. If you're going to be away from your machine for more than 10 or 15 minutes, log out before leaving. This is both for the security of your account, and to ensure that others are able to use the lab resources while you are not.
- Don't use internet, internet chat of any kind in your regular lab schedule.
- Do not download or upload of MP3, JPG or MPEG files.

[illegible]
Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

- No games are allowed in the lab sessions.
- No hardware including USB drives can be connected or disconnected in the labs without prior permission of the lab in-charge.
- No food or drink is allowed in the lab or near any of the equipment. Aside from the fact that it leaves a mess and attracts pests, spilling anything on a keyboard or other piece of computer equipment could cause permanent, irreparable, and costly damage. (and in fact *has*) If you need to eat or drink, take a break and do so in the canteen.
- Don't bring any external material in the lab, except your lab record, copy and books.
- Don't bring the mobile phones in the lab. If necessary then keep them in silence mode.
- Please be considerate of those around you, especially in terms of noise level. While labs are a natural place for conversations of all types, kindly keep the volume turned down.

If you are having problems or questions, please go to either the faculty, lab in-charge or the lab supporting staff. They will help you. We need your full support and cooperation for smooth functioning of the lab.

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

INSTRUCTIONS

Before entering in the lab

All the students are supposed to prepare the theory regarding the next experiment.

Students are supposed to bring the practical file and the lab copy.

Previous programs should be written in the practical file.

All the students must follow the instructions, failing which he/she may not be allowed in the lab.

While working in the lab

Adhere to experimental schedule as instructed by the lab in-charge.

Get the previously executed program signed by the instructor.

Get the output of the current program checked by the instructor in the lab copy.

Each student should work on his/her assigned computer at each turn of the lab.

Take responsibility of valuable accessories.

Concentrate on the assigned practical and do not play games

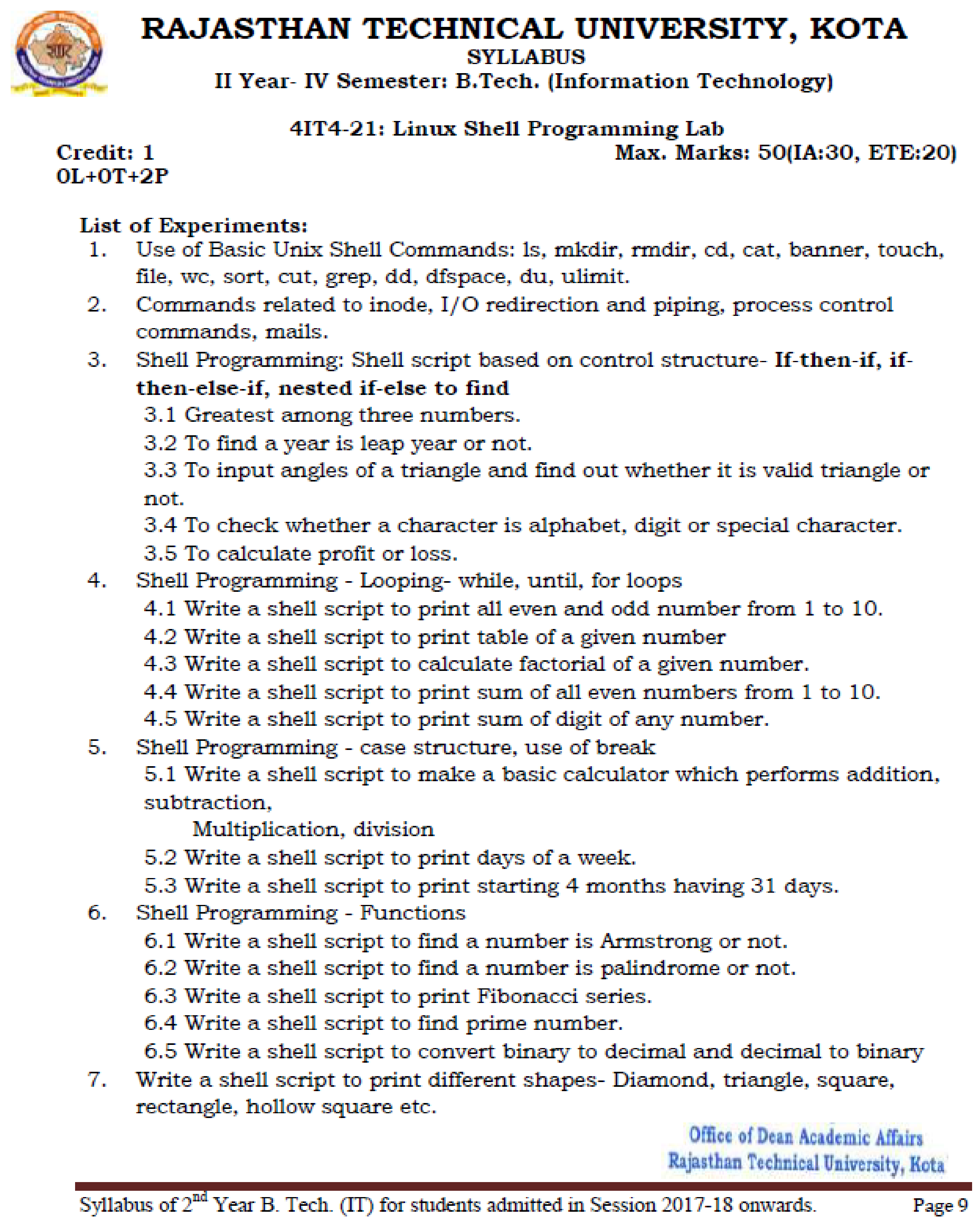
If anyone caught red handed carrying any equipment of the lab, then he/she will have to face serious consequences.

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

SYLLABUS

.



RAJASTHAN TECHNICAL UNIVERSITY, KOTA

SYLLABUS

II Year- IV Semester: B.Tech. (Information Technology)

4IT4-21: Linux Shell Programming Lab

Credit: 1 **Max. Marks: 50(IA:30, ETE:20)**

0L+0T+2P

List of Experiments:

1. Use of Basic Unix Shell Commands: ls, mkdir, rmdir, cd, cat, banner, touch, file, wc, sort, cut, grep, dd, dfspace, du, ulimit.
2. Commands related to inode, I/O redirection and piping, process control commands, mails.
3. Shell Programming: Shell script based on control structure- **If-then-if, if-then-else-if, nested if-else to find**
   3.1 Greatest among three numbers.
   3.2 To find a year is leap year or not.
   3.3 To input angles of a triangle and find out whether it is valid triangle or not.
   3.4 To check whether a character is alphabet, digit or special character.
   3.5 To calculate profit or loss.
4. Shell Programming - Looping- while, until, for loops
   4.1 Write a shell script to print all even and odd number from 1 to 10.
   4.2 Write a shell script to print table of a given number
   4.3 Write a shell script to calculate factorial of a given number.
   4.4 Write a shell script to print sum of all even numbers from 1 to 10.
   4.5 Write a shell script to print sum of digit of any number.
5. Shell Programming - case structure, use of break
   5.1 Write a shell script to make a basic calculator which performs addition, subtraction,
   Multiplication, division
   5.2 Write a shell script to print days of a week.
   5.3 Write a shell script to print starting 4 months having 31 days.
6. Shell Programming - Functions
   6.1 Write a shell script to find a number is Armstrong or not.
   6.2 Write a shell script to find a number is palindrome or not.
   6.3 Write a shell script to print Fibonacci series.
   6.4 Write a shell script to find prime number.
   6.5 Write a shell script to convert binary to decimal and decimal to binary
7. Write a shell script to print different shapes- Diamond, triangle, square, rectangle, hollow square etc.

Office of Dean Academic Affairs
Rajasthan Technical University, Kota

Syllabus of 2nd Year B. Tech. (IT) for students admitted in Session 2017-18 onwards. Page 9

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

RAJASTHAN TECHNICAL UNIVERSITY, KOTA

SYLLABUS

II Year- IV Semester: B.Tech. (Information Technology)

8. Shell Programming – Arrays
   - 8.1 Write a C program to read and print elements of array.
   - 8.2 Write a C program to find sum of all array elements.
   - 8.3 Write a C program to find reverse of an array.
   - 8.4 Write a C program to search an element in an array.
   - 8.5 Write a C program to sort array elements in ascending or descending order.

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

MARKS SCHEME

RTU Marks Scheme

Maximum Marks Allocation		
Sessional	End-Term	Total
30	20	50

Marks Division

1st / 2nd Midterm and End-term		
Experiment	Viva	Total
15	5	20
Attendance Performance (Internal)		
Attendance	Performance	Total
5	15	20

Assessment of an Experiment

Total Marks – 10

Attendance	Discipline	Performance	Record	Viva	Total
2	2	**3**	1	2	10

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

LAB PLAN

Total number of experiments – 11

Total number of turns required -12

Exp. No.	Description of Experiment
0	Zero Lab Introduction Linux Shell Programming
1	Use of basic Unix shell commands: ls, mkdir, rmdir, cd, cat, banner, touch, file, wc
2	Use of basic Unix shell command sort, cut, grep
3	Basic commands - dd, dfspace, du, ulimit. Commands related to inode
4	I/O redirection and piping. Process control commands and mails.
5	**Shell Programming: Shell script based on control structure- If-then-if, if-then-else-if, nested if-else to find** 5.1 Greatest among three numbers. 5.2 To find a year is leap year or not.
6	6.1 To input angles of a triangle and find out whether it is valid triangle or not. 6.2 To check whether a character is alphabet, digit or special character.
7	**Shell Programming - Looping- while, until, for loops** 7.1 Write a shell script to print all even and odd number from 1 to 10 7.2 Write a shell script to print table of a given number 7.3 Write a shell script to calculate factorial of a given number. 7.4 Write a shell script to print sum of all even numbers from 1 to 10.
8	**Shell Programming - case structure, use of break** 8.1 Write a shell script to make a basic calculator which performs addition, subtraction, Multiplication, division 8.2 Write a shell script to print days of a week. 8.3 Write a shell script to print starting 4 months having 31 days.
9	**Shell Programming – Functions** 9.1 Write a shell script to find whether a number is palindrome or not. 9.2 Write a shell script to print Fibonacci series. 9.3 Write a shell script to find prime number. 9.4 Write a shell script to convert binary to decimal and decimal to binary.
10	Write a shell script to print different shapes- Diamond, triangle, square, rectangle, hollow square etc
11	**C programming** 11.1 Write a C program to read N elements in an array and then find sum of all array elements. 11.2 Write a C program to search an element in an array. 11.3 Write a C program to sort array elements in ascending or descending order

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

Number of turns required for

Experiment Number	Turns	Scheduled Day
Zero Lab	1	Day 1
Exp. 1	1	Day 2
Exp. 2	1	Day 3
Exp. 3	1	Day 4
Exp. 4	1	Day 5
Exp. 5	1	Day 6
Exp. 6	2	Day 7
Exp. 7	1	Day 8
Exp. 8	1	Day 10
Exp. 9	1	Day 11
Exp. 10	1	Day 12
Exp. 11	1	Day 13

Distribution of lab hours

Attendance	05 minutes
Explanation of experiment	30 minutes
Performance of experiment	45 minutes
Record Checking	15 minutes
Viva / Quiz / Queries	25 minutes
Total	**120 minutes**

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

Introduction to the subject (Zero Lab)

Course Objectives:

The objective of this lab is to make students familiar with the Linux command-line environment, develop the skills of shell scripting and the process of developing C/C++ programs in Linux. This course introduces the UNIX/Linux operating system, including: command line tools, input/output processing, internal and external commands and shell configuration. The course explores the use of operating system utilities such as text editors, electronic mail, file management, scripting, and C/C++ compilers.

Lab Outcomes

On completion of this course the student should be able to:

1. **Use** UNIX/Linux utilities to create and manage simple file processing operations, organize directory structures with appropriate security, process control, I/O redirection and piping, and write shell scripts.
2. **Identify** appropriate Linux/Unix commands and use them to perform advanced shell programming.
3. **Create** C/ C++ programs and execute them in Linux environment.
4. **Demonstrate** skills to use various Linux tools and utilities in command line environment.
5. **Write** lab records in professional manner and submit timely without copying from other sources.
6. Orally **present** the lab work in unambiguous manner with proper linkage with required concepts.

LO-PO & PSO Mapping

LO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PS O1	PS O2	PS O3
LO1	3	-	-	-	-	-	-	-	-	-	-	-	3	-	-
LO2	-	3	-	-	-	-	-	-	-	-	-	-	-	3	-
LO3	-	-	3	-	-	-	-	-	-	-	-	-	-	-	3
LO4	-	-	-	-	3	-	-	-	-	-	-	-	-	-	2
LO5	-	-	-	-	-	-	-	3	-	-	-	-	-	-	-
LO6	-	-	-	-	-	-	-	-	-	3	-	-	-	-	-

Software required: -

Linux OS - Ububtu/ RedHat/ Caldera

Students may also use Cygwin Linux environment on top of Windows OS.

Hardware required (optional):-

Only PCs are required.

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

Software / Platform / Tool Installation Procedure

Cygwin Installation Guide

Cygwin is a POSIX-compatible programming and runtime environment that runs natively on Microsoft Windows. Under Cygwin, source code designed for Unix-like operating systems may be compiled with minimal modification and executed.

Use the following link to Installing and Updating Cygwin for 64-bit versions of Windows.

https://www.cygwin.com/install.html

Linux Introduction

Linux is a free open source UNIX based OS for PCs that was originally developed in 1991 by Linus Torvalds, a Finnish undergraduate student. Linux conforms to a set of IEEE standards called POSIX (Portable Operating System Interface).

The open source nature of Linux means that the source code for the Linux kernel is freely available so that anyone can add features and correct deficiencies. This approach has been very successful and what started as one person's project has now turned into a collaboration of hundreds of volunteer developers from around the globe.

The open source approach has not just successfully been applied to kernel code, but also to application programs for Linux.

As Linux has become more popular, several different development streams or distributions have emerged, **e.g. Redhat, Slackware, Mandrake, Debian, Ubuntu and Caldera.** A distribution comprises a prepackaged kernel, system utilities, GUI interfaces and application programs.

Redhat is one of the most popular distribution because it has been ported to a large number of hardware platforms (including Intel, Alpha, and SPARC), it is easy to use and install and it comes with a comprehensive set of utilities and applications including the X Windows graphics system, GNOME and KDE GUI environments, and the Star Office suite (an open source MS-Office clone for Linux).

Architecture of the Linux Operating System

Linux has all of the components of a typical OS

Kernel

The Linux kernel includes device driver support for a large number of PC hardware devices (graphics cards, network cards, hard disks etc.), advanced processor and memory management features, and support for many different types of file systems.

Shells and GUIs

Linux supports two forms of command input: through textual command line shells similar to those found on most UNIX systems (e.g. sh - the Bourne shell, bash - the Bourne again shell and csh - the C shell) and through graphical interfaces (GUIs) such as the KDE and GNOME window managers.

System Utilities

Virtually every system utility that we would expect to find on standard implementations of UNIX (including every system utility described in the POSIX.2 specification) has been ported to Linux. This includes commands such as ls, cp, grep, awk, sed, bc, wc, more, and so on. These system utilities are designed to be powerful tools that do a single task extremely well (e.g. grep finds text inside files while wc counts the number of words, lines and bytes inside a

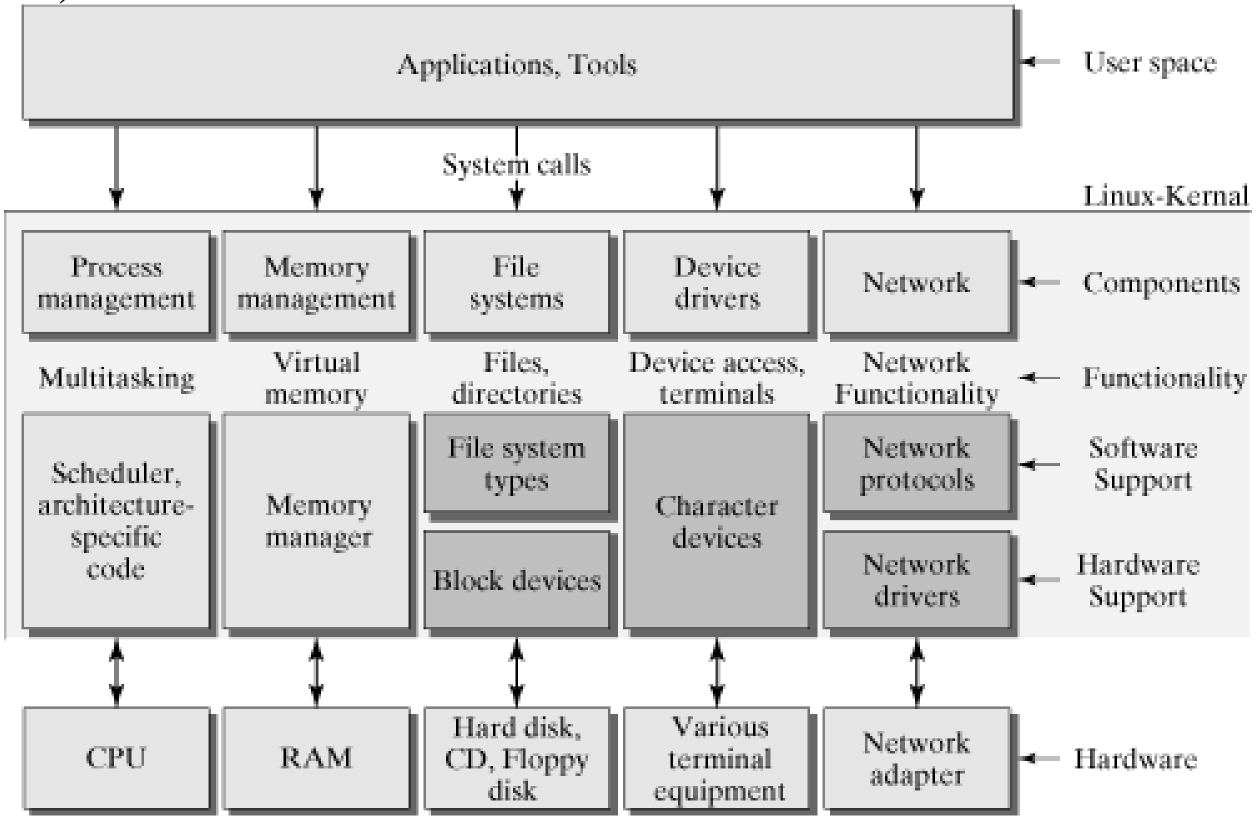
[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

file). Users can often solve problems by interconnecting these tools instead of writing a large monolithic application program.

Application programs

Linux distributions typically come with several useful application programs as standard. Examples include the emacs editor, xv (an image viewer), gcc (a C compiler), g++ (a C++ compiler), xfig (a drawing package), latex (a powerful typesetting language) and soffice (Star Office, which is an MS-Office style clone that can read and write Word, Excel and PowerPoint files).



Architecture of the Linux Operating System

Linux Documentation or Help

Distributions of Linux do not typically come with hardcopy reference manuals. However, its online documentation has always been one of Linux's strengths. The man (or manual) and info pages have been available via the man and info utilities since early releases of the operating system.

The --help Option

Most GNU utilities provide a —**help** option that displays information about the utility. Non-GNU utilities may use a –**h** or –**help** option to display help information.

```
$ cat --help
Usage: cat [OPTION] [FILE]...
Concatenate FILE(s), or standard input, to standard output.
```

If the information that —**help** displays runs off the screen, send the output through the less pager using a pipe:

```
$ ls --help | less
```

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

man: Displays the System Manual

The man utility displays (man) pages from the system documentation in a textual environment. This documentation is helpful when you know which utility you want to use but have forgotten exactly how to use it.

```
$ man passwd
$ man ls
```

References

1. A Practical Guide to Linux Commands, Editors, and Shell Programming, Mark G Sobel, Pearson Education
2. Linux: The Complete Reference by Richard Petersen, McGrawa Hill

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

Experiment-1

Objective:
Use of basic Unix shell commands: ls, mkdir, rmdir, cd, cat, banner, touch, file, wc

Theoretical Description including Algorithm:
1. ls command
ls -l command
Columns above indicate specific things:
Column 1 indicates information regarding file permission.
Column 2 indicates the number of links to the file.
Column 3 & 4 indicates the owner and group information.
Column 5 indicates size of the file in bytes.
Column 6 shows the date and time on which the file was recently modified.
Column 7 shows the file or directory name.
Linux ls -l --block-size=[SIZE]
If you want to display the file size of your list in a particular format or size, then you can use this command. Just put the size in place of [SIZE] as per your requirement.
Syntax:
ls -l --block-size=[SIZE]
Example:
ls -l --block-size=M (M for megabytes, K=Kilo, G=Giga, T=Tera)
ls -d */
If you only want to display the sub-directories excluding all other files, you can use this command.
Example:
ls -d */
Linux ls -g
If you don't want to display the owner information in your list, then you can exclude this column with the help of this command.
Example:
ls -g
ls -lG
If you don't want to display the group information in your list then you can exclude this column with the help of this command.
Linux ls ~
Linux ls ~ command shows the contents of the home directory. Let us see the example of ls ~ command.
Example:
ls ~
Linux ls ../
This command contains the list of the parent directory.
In the given example, our current directory is Downloads, and by using ls ../ command, we have listed out the content of its parent directory "home directory".
Example:

[illegible]
Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

ls ../
ls -F (or --classify on GNU) to show indicators after each entry that identify the kind of file it is.
A slash (/) denotes a directory (or "folder").
An asterisk (*) denotes an executable file. This includes a binary file (compiled code) as well as scripts (text files that have executable permission).
An at sign (@) denotes a symbolic link (or "alias").

2. mkdir – make directories
create directory[ies] if they are not already exists.
Usage: mkdir <DIREECTORY NAME>
eg. mkdir mydir
3. rmdir- Remove directories
remove directory[ies] if they are empty.
Usage: rmdir [DIRECTORY NAME]
Eg. rmdir mydir remove mydir directory if this is empty.
rm-remove files or directories
remove file[s] or directory[ies]
Usage: rm –[option] [DIRECTORY NAME OR FILE NAME]
Eg.
rm –i [FILENAME] remove file interactively. This will ask before removing file.
rm –f [FILENAME] remove file forcefully.
rm –r [FILENAME] recursively remove non empty directory.

4. cd – changes directories
Usage: cd [DIRECTORY NAME]
Eg. cd mydir
5. pwd – print working directory
Shows what directory (folder) you are in.
In Linux, your home directory is /home/username.
Usage: pwd
eg. pwd show present working directory
/home/username
6. mv-move
move or rename files or directories
Usage: mv [SOURCE DIRECTORY] [DESTINATION DIRECTORY]
mv [OLD FILENAME] [NEW FILENAME]
Eg. mv linixdir mydir renaming or moving directory linixdir as mydir. After execution of command, the destination files are only available.
7. cp – copy
copy files and directories
Usage: cp [OPTION] SOURCE FILE] [DESTINATION FILE]
eg. cp sample.txt sample_copy.txt After execution of command, the both source and destination files are available.
8. touch command
To make a new blank file or changing the file timestamps
Creating a blank file

[illegible]
Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

e.g. touch newfile
In fact the touch command real function is to change the modification and/or access timestamps of a file.
-a option will change the access time,
-m option can be used to change the last modified time
-t to specify the new time
-r option to set the value to be the same as another file
Note: If the time is not specified using -t or -r, then the current time will be used
touch -m -t 201205251700 thisfile
will set the last modified date of thisfile to 25th may 2012, 17:00.
touch -m -r referencefile thisfile - will set the modified date to be the same as another file

9. file command

To check the type of a file
file filename
will tell us if the file is a text file, a command file or a directory etc.
file file.txt
file.txt: ASCII text
To show just the file type pass the -b option
file -b file.txt
ASCII text
The file command can be useful as filenames in UNIX bear no relation to their file type. So a file called somefile.csv could actually be a zip file. This can be verified by the file command.
file somefile.csv
somefile.csv: Zip archive data, at least v2.0 to extract
file command cont…
How to determine the file type of multiple files
file *.txt
The output will display the information on all .txt files in the current directory.
file *
Below is an example of what may appear when running file with a wildcard for all files:
shutdown.htm: HTML document text
si.htm: HTML document text
side0.gif: GIF image data, version 89a, 107 x 18
robots.txt: ASCII text, with CRLF line terminators
myprog.c: C source ASCII text
file /dev/hda1 or file /dev/sda1
/dev/hda1: block special (0/0)

10. **cat** – concatenate files and print on the standard output
Usage: cat [OPTION] [FILE]...
eg. cat file1.txt file2.txt
cat when supplied with more than one file will concatenate the files without any header information.
cat used to display the contents of a small file on terminal
usage: cat [file name]
cat- To create file
Usage: cat > [file name]

[illegible]
Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

NOTE: Press and hold CTRL key and press D to stop or to end file (CTRL+D)
cat – to apend text at the end of file
Usage: cat >> [file name]
NOTE: Press and hold CTRL key and press D to stop or to end file (CTRL+D)
11. **echo** – display a line of text
Usage: echo [OPTION] [string] ...
eg. echo I love India
echo $HOME
12. **wc** -command is used to count lines, words and characters, depending on the option used.
Usage: wc [options] [file name]
You can just print number of lines, number of words or number of charcters by using following options:
-l: Number of lines
-w : Number of words
-c : Number of characters
Eg. wc file.txt
count number of lines, words and character (including whitespaces , newline etc) in a file.

Supporting File/Dataset:

Program:

Output:

Conclusion:

VIVA Questions:
Q. How do you list the attributes of a directory?

[illegible]
Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

Experiment-2

Objective:
Use of basic Unix shell command sort, cut, grep

1. grep – print lines matching a pattern

Usage: grep [OPTION] PATTERN [FILE]...
eg. grep -i apple sample.txt
Options:
-i case-insensitive search
-n show the line# along with the matched line
-v invert match, e.g. find all lines that do NOT match
-w match entire words, rather than substrings

2. sort command

SORT command is used to sort a file, arranging the records in a particular order. By default, the sort command sorts file assuming the contents are ASCII. Using options in sort command, it can also be used to sort numerically.
Examples
Suppose you create a data file with name file.txt
Command :

```
$ cat > file.txt
abhishek
chitransh
satish
rajan
naveen
divyam
harsh
```

Now use the sort command

```
sort file.txt
```

Output :

```
abhishek
chitransh
divyam
harsh
naveen
rajan
satish
```

Note: This command does not actually change the input file, i.e. file.txt
Sort function with mix file i.e. uppercase and lower case :When we have a mix file with both uppercase and lowercase letters then first the lower case letters would be sorted following with the upper case letters .
Options with sort function
-o Option : Unix also provides us with special facilities like if you want to write the output to a new file, output.txt, redirects the output like this or you can also use the built-in sort option -o,

[illegible]
Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

which allows you to specify an output file. Using the -o option is functionally the same as redirecting the output to a file.

```
sort inputfile.txt > filename.txt
sort -o filename.txt inputfile.txt
$ sort file.txt > output.txt
$ sort -o output.txt file.txt
$ cat output.txt
```

Output :

```
abhishek
chitransh
divyam
harsh
naveen
rajan
satish
```

-k Option : Unix provides the feature of sorting a table on the basis of any column number by using -k option.
Use the -k option to sort on a certain column. For example, use "-k 2" to sort on the second column.
Example :
Let us create a table with 2 columns

```
$ cat > employee.txt
manager  5000
clerk    4000
employee  6000
peon     3500
director 9000
guard     3000
sort -k 2n employee.txt
guard    3000
peon     3500
clerk    4000
manager  5000
employee 6000
director 9000
```

-r Option: Sorting In Reverse Order

```
sort -r file.txt
```

Output :

```
Satish
Rajan
naveen harsh
Divyam
Chitransh
abhishek
```

-n Option : To sort a file numerically use –n option.

[illegible]
Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

This option is used to sort the file with numeric data present inside.

```
cat > file1.txt
50
39
15
89
200
sort -n file1.txt
Output :
15
39
50
89
200
```

3. cut command

The cut command in UNIX is a command for cutting out the sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by **byte position, character and field**. Basically the cut command slices a line and extracts the text.

Syntax:

cut OPTION... [FILE]...

Let us consider a files having name **state.txt** that contains 5 names of the Indian states.

```
$ cat state.txt
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
```

Options and their Description with examples:

1. -b(byte): To extract the specific bytes, you need to follow -b option with the list of byte numbers separated by comma. Range of bytes can also be specified using the hyphen(-). It is necessary to specify list of byte numbers otherwise it gives error. **Tabs and backspaces** are treated like as a character of 1 byte.

List without ranges

```
$ cut -b 1,2,3 state.txt
And
Aru
Ass
Bih
Chh
```

List with ranges

```
$ cut -b 1-3,5-7 state.txt
Andra
Aruach
Assm
Bihr
```

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

```
Chhtti
```

It uses a special form for selecting bytes from beginning upto the end of the line:
In this, 1- indicate from 1st byte to end byte of a line

```
$ cut -b 1- state.txt
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
```

In this, -3 indicate from 1st byte to 3rd byte of a line

```
$ cut -b -3 state.txt
And
Aru
Ass
Bih
Chh
```

2.1.1. -b(byte): To extract the specific bytes, you need to follow -b option with the list of byte numbers separated by comma. Range of bytes can also be specified using the hyphen(-).

$cut -c [(k)-(n)/(k),(n)/(n)] filename

Here,**k** denotes the starting position of the character and **n** denotes the ending position of the character in each line, if *k* and *n* are separated by "-" otherwise they are only the position of character in each line from the file taken as an input.

```
$ cut -c 2,5,7 state.txt
nr
rah
sm
ir
hti
```

Above cut command prints second, fifth and seventh character from each line of the file.

```
$ cut -c 1-7 state.txt
Andhra
Arunach
Assam
Bihar
Chhatti
```

Above cut command prints first seven characters of each line from the file.
Cut uses a special form for selecting characters from beginning upto the end of the line:

```
$ cut -c 1- state.txt
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
```

Above command prints starting from first character to end. Here in command only starting position is specified and the ending position is omitted.

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

$ cut -c -5 state.txt

```
Andhr
Aruna
Assam
Bihar
Chhat
```

Above command prints starting position to the fifth character. Here the starting position is omitted and the ending position is specified.

3. -f (field): -c option is useful for fixed-length lines. Most unix files doesn't have fixed-length lines. To extract the useful information you need to cut by fields rather than columns. List of the fields number specified must be separated by comma. *Ranges are not described with -f option*. **cut** uses **tab** as a default field delimiter but can also work with other delimiter by using **-d** option.

Note: Space is not considered as delimiter in UNIX.

Syntax:

$cut -d "delimiter" -f (field number) file.txt

Like in the file **state.txt** fields are separated by space if -d option is not used then it prints whole line:

$ cut -f 1 state.txt

```
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
```

If -d option is used then it considered space as a field separator or delimiter:

$ cut -d " " -f 1 state.txt

```
Andhra
Arunachal
Assam
Bihar
Chhattisgarh
```

Command prints field from first to fourth of each line from the file.

Command:

```
$ cut -d " " -f 1-4 state.txt
```

Output:

```
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
```

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

Experiment-3

Objective

Basic commands - dd, dfspace, du, ulimit. Commands related to inode

1. 'dd' command

The dd command stands for "data duplicator" and used for copying and converting data. It is very powerful low level utility of Linux which can do much more like;

- Backup and restore the entire hard disk or partition.
- Backup of MBR (Master Boot Record)
- It can copy and convert magnetic tape format, convert between ASCII and EBCDIC formats, swap bytes and can also convert lower case to upper case.
- It can also be used by Linux kernel make files to make boot images.
- Only superuser can run this command

Syntax

dd if=<source file name> of=<target file name> [Options]

if=<source> –This is a source from where you want to copy data and 'if' stands for input-file.

of=<destination> –This is a source from where you want to write/paste data and 'of' stands for output-file.

Example 1: Clone one hard disk to another hard disk. This is useful when we are building many machines with same configuration. We no need to install OS on all the machines.

dd if=/dev/sda of=/dev/sdb

Example 2: We can take backup of a partition/complete HDD for future restoration.

Backing up a partition to a file(to home directory as hdadisk.img)

dd if =/dev/sda2 of=~/hdadisk.img

Restoring this image file in to other machine

dd if=hdadisk.img of=/dev/sdb3

Example 3:

dd command can be used as file copier as well

If we don't have cp command use dd command to copy a file from one location to other.

dd if=/home/imran/abc.txt of=/mnt/abc.txt

2. 'du' command

du [option(s)] [path]

This command, when executed without any parameters, shows the total disk space occupied by files and subdirectories in the current directory.

-a

Displays the size of each individual file

-h

Output in human-readable form

-s

Displays only the calculated total size

3. 'df' command

df [option(s)] [directory]

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

The **df** (disk free) command, when used without any options, displays information about the total disk space, the disk space currently in use, and the free space on all the mounted drives. If a directory is specified, the information is limited to the drive on which that directory is located.
-H or -h
shows the number of occupied blocks in gigabytes, megabytes, or kilobytes
(human-readable format)
-T
Type of file system (ext2, nfs, etc.)

4. 'ulimit' command

To see, set, or limit the resource usage of the current user. It is used to return the number of open file descriptors for each process. It is also used to set restrictions on the resources used by a process.
ulimit –a   All current limits are reported
ulimit -u
To display maximum users process or for showing maximum user process limit for the logged-in user.
ulimit –f
For showing the maximum file size a user can have.
ulimit –v
For showing maximum memory size limit.

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

Experiment-4

Objective

I/O redirection and piping. Process control commands and mails.

Process?

- A **process** is a program in execution.
- Every time you invoke a system utility or an application program from a shell, one or more "child" processes are created by the shell in response to your command.
- All Linux processes are identified by a unique process identifier or PID.
- An important process that is always present is the init process. This is the first process to be created when a UNIX system starts up and usually has a PID of 1

Piping

- $ cat hello.txt | sort | uniq
- creates three processes (corresponding to cat, sort and uniq) which execute concurrently.
- $ cat hello.txt | grep "dog" | grep -v "cat"
- finds all lines in hello.txt that contain the string "dog" but do not contain the string "cat".

Redirecting input and output

- Processes usually write to **standard output** (the screen) and take their input from **standard input** (the keyboard).
- There is in fact another output channel called **standard error**, where processes write their error messages; by default error messages are also sent to the screen.
- To redirect standard output to a file instead of the screen, we use the > operator:

-
  ```
  $ echo hello
  hello
  $ echo hello > output
  $ cat output
  hello
  ```
- In this case, the contents of the file output will be destroyed if the file already exists. If instead we want to append the output of the echo command to the file, we can use the >> operator:

-
  ```
  $ echo bye >> output
  $ cat output
  hello
  bye
  ```
- To capture standard error, prefix the > operator with a 2 (in UNIX the file numbers 0, 1 and 2 are assigned to standard input, standard output and standard error respectively), e.g.:

-
  ```
  $ cat nonexistent 2>errors
  $ cat errors
  cat: nonexistent: No such file or directory
  ```
- Standard input can also be redirected using the < operator, so that input is read from a file instead of the keyboard:

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

- $ cat < output
  hello
  bye
- $ cat < output > output
- will destroy the contents of the file output. This is because the first thing the shell does when it sees the > operator is to create an empty file ready for the output.

Process control commands

Whenever we issue a command in UNIX, it creates, or *starts*, a new process. When we tried out the ls command to list directory contents, you started a process (the ls command).

Each process in the system has a unique pid (process id)

When you start a process (run a command), there are two ways you can run it--in the *foreground* or *background*.

Foreground Processes

By default, every process that you start runs in the foreground.

Background Processes

A background process runs without being connected to your keyboard. If the background process requires any keyboard input, it waits.

The advantage of running a process in the background is that you can run other commands; you do not have to wait until it completes to start another!

The simplest way to start a background process is to add an ampersand (&) at the end of the command.

```
vi a.txt &
ls –R / | more
```

Moving a Foreground Process to the Background

In addition to running a process in the background using &, you can move a foreground process into the background.

While a foreground process runs, the shell does not process any new commands. Before you can enter any commands, you have to suspend the foreground process to get a command prompt.

The suspend key on most UNIX systems is Ctrl+Z

When a foreground process is suspended, a command prompt enables you to enter more commands; the original process is still in memory but is not getting any CPU time.

To resume the foreground process, you have two choices--background and foreground

Moving a Foreground Process to the Background

The bg command enables you to resume the suspended process in the background; the fg command returns it to the foreground.

Moving a Foreground Process to the Background

```
$ long_running_process
^Z[1] + Stopped (SIGTSTP) long_running_process
$ long_running_process2
^Z[2] + Stopped (SIGTSTP) long_running_process2
$
```

To move the first one to the background, I use the following:

```
$ bg %1
```

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

```
[1] long_running_process &
$
```

The second process is still suspended and can be moved to the background as follows:

```
$ bg %2
[2] long_running_process2 &
$
```

Moving a Background Process to the Foreground (fg Command)

When you have a process that is in the background or suspended, you can move it to the foreground with the fg command.

By default, the process most recently suspended or moved to the background moves to the foreground.

You can also specify which job, using its job number, you want to make foreground.

Moving a Background Process to the Foreground (fg Command)

```
$ long_running_process
^Z[1] + Stopped (SIGTSTP) long_running_process
$ bg
[1] long_running_process &
$
```

You can move it back to the foreground as follows:

```
$ fg %1
long_running_process
```

Listing Running Processes

jobs Command

The jobs command shows you the processes you have suspended and the ones running in the background.

```
$ jobs
[3] + Running first_one &
[2] - Stopped (SIGTSTP) second_one
[1] Stopped (SIGTTIN) third_one &
```

In the above example, I have three jobs. The first one (job 3) is running, the second (job 2) is suspended (a foreground process after I used Ctrl+Z), and the third one (job 1) is stopped in the background to wait for keyboard input:

ps Command(Process Status)

shows all running processes

```
$ ps
PID TTY TIME CMD
6738 pts/6 0:00 first_one
6739 pts/6 0:00 second_one
3662 pts/6 0:00 ksh
8062 pts/6 0:00 ps
6770 pts/6 0:01 third_one
```

Killing a Process (kill Command)

The kill command *kills*, or ends, a process

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

To kill job number 1 in the earlier example regarding waiting for keyboard input, I use the following:

```
$ kill %1
[1] - Terminated third_one &
$
```

We can also kill a specific process by specifying the process ID on the command line without the percent sign used with job numbers. To kill job number 2 (process 6738) in the earlier example using process ID, we use the following:

```
$ kill 6739
$
```

Killing a Process (kill Command)

In reality, kill does not physically kill a process; it sends the process a signal. By default, it sends the TERM (value 15) signal.

A process can choose to ignore the TERM signal or use it to begin an orderly shut down (flushing buffers, closing files, and so on).

If a process ignores a regular kill command, you can use kill -9 or kill -KILL followed by the process ID or job number (prefixed with a percent sign).

This forces the process to end.

[illegible]
Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

Experiment-5

Objective

Shell Programming: Shell script based on control structure- If-then-if, if-then-else-if, nested if-else to find

5.1 Greatest among three numbers.

5.2 To find a year is leap year or not.

Shell Scripts

A shell is a program which reads and executes commands for the user. Shells also usually provide features such job control, input and output redirection and a command language for writing *shell scripts*. A shell script is simply an ordinary text file containing a series of commands in a shell command language (just like a "batch file" under MS-DOS).

There are many different shells available on UNIX systems (e.g. sh, bash, csh, ksh, tcsh etc.), and they each support a different command language.

Here we will discuss the command language for the Bourne shell sh since it is available on almost all UNIX systems (and is also supported under bash and ksh).

Shell Variables and the Environment

A shell lets you define variables (like most programming languages). Once you have assigned a value to a variable, you access its value by prefixing a $ to the name.

```
$ str='hello world'
$ echo $str
hello world
```

Variables created within a shell are local to that shell, so only that shell can access them.

The set command will show you a list of all variables currently defined in a shell. If you wish a variable to be accessible to commands outside the shell, you can *export* it into the *environment*:

```
$ export str
```

Simple Shell Scripting

```
#!/bin/sh
# this is a comment
echo "The number of arguments is $#"
echo "The arguments are $*"
echo "The first is $1"
echo "My process number is $$"
echo "Enter a number from the keyboard: "
read number
echo "The number you entered was $number"
```

The shell script begins with the line "#!/bin/sh" . Usually "#" denotes the start of a comment, but #! is a special combination that tells UNIX to use the Bourne shell (sh) to interpret this script. The #! must be the first two characters of the script. The arguments passed to the script can be accessed through $1, $2, $3 etc.

$* stands for all the arguments, and $# for the number of arguments.

The process number of the shell executing the script is given by $$. The read number statement assigns keyboard input to the variable number.

How to execute a shell script

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

To execute this script, we first have to make the file first executable:

```
$ chmod +x first
$ ls -l first
$ ./first hello world
```

Shell scripts are able to perform simple conditional branches:

```
if [ test ]
then
   commands-if-test-is-true
else
   commands-if-test-is-false
fi
```

The *test* condition may involve file characteristics or simple string or numerical comparisons. There must be spaces before and after it as well as before the closing bracket.
Some common test conditions are:

-s *file*
   true if *file* exists and is not empty
-f *file*
   true if *file* is an ordinary file
-d *file*
   true if *file* is a directory
-r *file*
   true if *file* is readable
-w *file*
   true if *file* is writable
-x *file*
   true if *file* is executable
$X -eq $Y
   true if X equals Y
$X -ne $Y
   true if X not equal to Y
$X -lt $Y
   true if X less than $Y
$X -gt $Y
   true if X greater than $Y
$X -le $Y
   true if X less than or equal to Y
$X -ge $Y
   true if X greater than or equal to Y
"$A" = "$B"
   true if string A equals string B

[illegible]
Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

"$A" != "$B"
    true if string A not equal to string B
$X ! -gt $Y
    true if string X is not greater than Y
$E -a $F
    true if expressions E and F are both true
$E -o $F
    true if either expression E or expression F is true

for loops

Sometimes we want to loop through a list of files, executing some commands on each file. We can do this by using a for loop:

for *variable* in *list*
do
    statements (referring to $*variable*)
done

while loops
Another form of loop is the while loop:

while [*test*]
do
    statements   (to be executed while *test* is true)
done

[illegible]
Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

3.1 Greatest among three numbers.

```
# greatest among three numbers
echo enter three numbers
read a
read b
read c
if [ $a -gt $b -a $a -gt $c ]
then
        max=$a
elif [ $b -gt $c ]
then
        max=$b
else
        max=$c

fi
echo "max = $max"
```

3.2 To find a year is leap year or not.

```
echo -n "Input year (yyyy): "
read y
a=`expr $y % 4`
b=`expr $y % 100`
c=`expr $y % 400`
if [ $a -eq 0 -a $b -ne 0 -o $c -eq 0 ]
then
echo "$y is a leap year"
else
echo "$y is not a leap year"
fi
```

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

Experiment-6

Shell Programming: Shell script based on control structure- If-then-if, if-then-else-if, nested if-else to find

6.1 To input angles of a triangle and find out whether it is valid triangle or not.

6.2 To check whether a character is alphabet, digit or special character.

6.3 To calculate profit or loss.

6.1 To input angles of a triangle and find out whether it is valid triangle or not.

```
#exp 3.3
echo "input angle a "
read a
echo "input angle b "
read b
echo "input angle c "
read c
d=`expr $a + $b + $c`
if [ $a -eq 0 -o $b -eq 0 -o $c -eq 0 ]
then
echo "Invalid angle..."
else
if [ $d == 180 ]
then
echo "Valid triangle"
else
echo "Invalid triangle"
fi
fi
```

6.2 To check whether a character is alphabet, digit or special character.

```
echo "Enter a single character "
read ch
echo $ch | grep [a-zA-Z]>/dev/null
if [ $? -eq 0 ]
then
echo "Alphabet"
else
echo $ch | grep [0-9] >/dev/null
if [ $? -eq 0 ]
then
echo "Digit"
else
echo "Special Character..."
fi
fi
```

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

Experiment 7: Write a shell script to print sum of all even numbers from 1 to 10.

```
s=0
i=1
while [ $i -le 10 ]
do
        r=`expr $i % 2`
        if [ $r -eq 0 ]
        then
                s=`expr $s + $i`
        fi
        i=`expr $i + 1`
done
echo "Sum of even numbers= $s"
```

Experiment 8:
8.1 Write a shell script to make a basic calculator which performs addition, subtraction, Multiplication, division

```
echo "enter first num: "
read n1
echo "enter second num: "
read n2
echo "enter operator (+, -, *, /): "
read ch
case $ch in
        +)
        ans=`expr $n1 + $n2`
        ;;
        -)
        ans=`expr $n1 - $n2`
        ;;
        \*)
        ans=`expr $n1 \* $n2`
        ;;
        /)
        ans=`expr $n1 / $n2`
        ;;
esac
echo "Answer is: $ans"
```

8.2 Write a shell script to print days of a week.

```
echo "enter day of week"
read day
case $day in
```

[illegible]
Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

```
        "sunday")
        echo "7"
        ;;
        "monday")
        echo "1"
        ;;
        "tuesday")
        echo "2"
        ;;
        "wednesday")
        echo "3"
        ;;
        "thursday")
        echo "4"
        ;;
        "friday")
        echo "5"
        ;;
        "saturday")
        echo "6"
        ;;
        *)
        echo "invalid input"
        ;;
esac
```

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

Experiment 9
Some examples of simple functions

(i)
```
lsl()
{
	echo "We are in lsl()"
	ls -l
	echo $1
}
#function call
lsl z1
```
(ii)
```
function foo()
{
	echo $1
	echo $2
}
foo 1 2   #function call
```
(iii)
```
function logmsg
{
	echo "`date '+%F %T'`:$1"
}
logmsg "this is a text msg"
```

9.1 Write a shell script to find whether a number is palindrome or not.
```
#to check whether a num is palindrome or not
function palindrome
{
	s=0
	n=$1
	while [ $n -ne 0 ]
	do
		r=`expr $n % 10`
		s=`expr $s \* 10 + $r`
		n=`expr $n / 10`
	done
	if [ $s -eq $1 ]
	then
		return 1
	else
		return 0
	fi
}
echo "Enter a num.: "
```

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

```
read z
palindrome "$z"
echo $?
```

9.2 Write a shell script (using function) to print Fibonacci series.

```
function fib
{
	x=0
	y=1
	i=2
	echo "serief upto $n"
	echo $x
	echo $y
	while [ $i -lt $n ]
	do
		i=`expr $i + 1`
		z=`expr $x + $y`
		echo $z
		x=$y
		y=$z
	done
}
echo "enter number of terms: "
read n
fib $n
```

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

Experiment 10

Write a shell script to print different shapes- Diamond, triangle, square, rectangle, hollow square etc

```
function hollowsquare
{
	n=$1
	i=1
	while [ $i -le $n ]
	do
		j=1
		while [ $j -le $n ]
		do
			if [ $i -eq 1 -o $i -eq $n ]
			then
				echo -n "*"
			else
				if [ $j -eq 1 -o $j -eq $n ]
				then
					echo -n "*"
				else
					echo -n " "
				fi
			fi
			j=`expr $j + 1`
		done
		i=`expr $i + 1`
		echo
	done
}
echo "enter no of terms: "
read n
hollowsquare $n
```

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR

Experiment 11

C programming

11.1 Write a C program to read N elements in an array and then find sum of all array elements.

```
#include<stdio.h>
int main()
{
	int arr[10];
	int i,n;
	sum=0;
	printf("Enter no of terms: ");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("enter an element ");
		scanf("%d",&arr[i]);
	}
	for (i=0;i<n;i++)
	{
		printf("%d  ", arr[i]);
		sum=sum+arr[i];
	}
	printf("Sum of all array elements = %d",sum);
	return 0;
}
```

[illegible]

Dr. Mahesh Bundele
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-6, RIICO Institutional Area
Sitapura, JAIPUR