MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL

(A constituent unit of MAHE, Manipal)

LABORATORY MANUAL

Department	:	Department of Data Science and Computer Applications
Course Name & code	:	LINUX PROGRAMMING LAB & MCA 4243
Semester & branch	:	II Semester MCA
Name of the faculty	:	Tojo Thomas, Akshay Bhat and Abhilash K Pai
No of contact hours/week	:	L: 0, T: 0, P: 3, C: 1

COURSE OUTCOMES

	At the end of this course, the student should be able to:	**No. of Contact Hours**	**Marks**
CO1:	Illustrate the usage of basic and advanced Linux commands, shell scripting using appropriate Linux system calls.	18	40
CO2:	To implement the process scheduling, synchronization and deadlocks.	12	30
CO3:	To implement the page replacement, file handling and disk scheduling algorithms.	6	30
	Total	36	100

ASSESSMENT PLAN

1. **Continuous Evaluation** 60%

   Two Evaluations and One Mid-Term Examination.

   **Each of the two evaluation is conducted for 20 marks, where:
   - Observation book : 6 marks
   - Execution check : 7 marks
   - Test/Quiz : 7 marks

   **Mid-Term Evaluation will be conducted in Week 6 for 20 marks (portion Week 1-5).

   The test/quiz details are as follows:
   Test/Quiz 1: In the 4th Week (portion Week 1-3).
   Test/Quiz 2: In the 10th Week (portion Week 6-9).

   Total Internal marks = ((6+7+7)*2) + 20 = 60 marks

2. **Lab Examination** 40%

   Examination of 2 hours duration (Max. Marks: 40). Program Write up: 15 Marks; Program Execution: 25; Marks Total: 15+25 =40 Marks

LESSON PLAN

L No	TOPICS	Course Outcome Addressed
L1	LINUX COMMANDS and Vi-Editor	CO1
L2	SHELL PROGRAMS-1	CO1
L3	SHELL PROGRAMS-2 with FILE-RELATED COMMANDS	CO1
L4	SHELL PROGRAMS-3 with ADDITIONAL LINUX COMMANDS	CO1
L5	PROCESSES	CO1
L6	MID-TERM LAB EXAMINATION	CO1
L7	PROCESS SCHEDULING-1	CO2
L8	PROCESS SCHEDULING-2	CO2
L9	PROCESS SYNCHRONIZATION AND DEADLOCKS	CO2
L10	PAGE REPLACEMENT	CO3
L11	DISK SCHEDULING	CO3
L12	END-TERM LAB EXAMINATION	

INSTRUCTIONS TO STUDENTS

1. **Attending the laboratory sessions**:
   a) You should be regular and come prepared for implementing the specified programs.
   b) In case you miss a session, it is your responsibility to complete the pending work before coming to the next session.

2. **Implementing the programs**:
   a) You should implement the programs individually.
   b) Prescribed text / reference books and class notes can be kept ready for reference, if required.
   c) The programs should meet the following criteria.
      - Programs should be interactive with appropriate messages for inputs and descriptive messages for outputs.
      - Programs should perform input validation (data type, range error, etc.) and give appropriate error messages and suggest corrective actions.
      - Comments should be used to specify the statement of the problem and the purpose, inputs and outputs of every function.
      - Statements within the program should be properly indented.
      - Meaningful names should be used for variables and functions.
      - Constants and type definitions should be used wherever needed.

3. **Observation book**:
   a) You should maintain an observation book exclusively for the laboratory work and should bring it to the laboratory during every session.
   b) You should reserve first few pages of the book for writing the index for the programs that follow. The index should consist of program serial number, complete title of the program, date of completing the program, page number where the program starts. Sufficient space must be left in the last column for getting the signature of the faculty.
   c) Once the programs get correctly executed, you should copy the same in your observation book and appropriately fill in the index entries.

4. **Continuous evaluation**: You should submit the completed observation book during every evaluation, show the execution of the programs asked by the faculty and answer the viva-voce questions.

5. **End-semester examination**: Questions for end-semester laboratory examination need not necessarily be limited to the questions in the manual, but could involve some variations and / or combinations of the questions.

PATTERN OF WRITING OBSERVATIONS

Linux Shell Commands

- Syntax: Write the syntax of the command specifying all argument delimiters, punctuation characters etc. without fail.
- Description: Write the complete description of the command – meaning of the command, use of various arguments etc.
- Illustration: Write examples of using the command. There should be more than one example for commands involving arguments and multiple possibilities.

Linux Shell Programs and Implementation of OS concepts using C/C++

- Program description: Write the complete description of the program.
- Inputs required and Outputs expected: Write about all the inputs required for the execution and also about the outputs expected for each possible type of input.
- Program Logic: Write the complete description of each module used in the program.
- Coding: Write the actual code that is implemented and executed.
- Conclusion: Write your observations about the implementation of the program, the inputs and outputs, and various possible alternatives with regard to inputs and the corresponding outputs.

Note:

- Write the correct serial number (as written in the index page) in the beginning of each set of commands / programs.
- Write the page number at the top of each page aligned at the center

WEEK 1: LINUX COMMANDS and Vi-Editor

1) Basic Linux Commands: Test the function of each of the following.
   a) Online help: man, help
   b) Directory related commands: pwd, mkdir, cd, rmdir
   c) File related commands: ls, cp, mv, rm, touch
   d) Others: who, whoami, whatis, date, wc

2) File and directory permissions:
   a) Users and ownership
   b) Groups and Changing group ownership
   c) File permissions and Changing file permissions

3) Additional and advanced Linux commands:
   a) cat, echo
   b) Wildcards: *, ? , [ ] , &&
   c) Input/output redirection: |, >, >>, <, <<

4) Vi editor commands.

<u>Week-1 Exercise 1:</u>

i. Write shell commands for the following.
ii. Create 11 subdirectories in your home directory under OSxxx.
iii. Create subdirectory **TestFolder** under ***Week1***
iv. Create 3 different empty files in the TestFolder.
v. Add different contents in each of them using *vi editor (say f1, f2 and f3)*
vi. Move back to Week1 folder and copy *f1* to this location.
vii. Create one more file with the command listing the **number of users** and **number of files.**
viii. Check content of this file.
ix. Use command to list all the files which starts with either ‘a’ or ‘A’.
x. Create a newfile called **month_names** and use sort command to sort them.
xi. Display the first five and last five lines of a given file.
xii. Redirect the output of commands ‘pwd’, ‘date’ and ‘ls’ in succession to a file.
xiii. Exercise for vi editor commands: *(Refer to next page)*

Looking into the Linux kernel

The core of the Linux system is the kernel. The kernel controls all of the hardware and software on the computer system, allocating hardware when necessary, and executing software when required.

*****

If you've been following the Linux world at all, no doubt you've heard the name Linus Torvalds. Linus is the person responsible for creating the first Linux kernel software while he was a student at the University of Helsinki. He intended it to be a copy of the Unix system, at the time a popular operating system used at many universities.

*****

After developing the Linux kernel, Linus released it to the Internet community and solicited suggestions for improving it. This simple process started a revolution in the world of computer operating systems. Soon Linus was receiving suggestions from students as well as professional programmers from around the world.

For the above exercise (**Ex. xiii**) perform the following editor operations:

a) Remove blank lines wherever the *** is found

b) Replace all occurrences of the word "Linux" with "Ubuntu"

c) Save and return to command line prompt

d) Use command to view "Linux.txt" at the command prompt without using vi

e) Rename "Linux.txt" with the name "Ubuntu.txt"

f) Make a copy of the same text file in the name "Linux1.txt" at command prompt

Week-1 Exercise 2:

i. Write a command to redirect the output of the ls command to a file named file_list.txt.
ii. Append the output of the date command to the file file_list.txt.
iii. Redirect both the output and error messages of the command ls non_existent_file to a file named errors.log.
iv. Use the << redirection operator to create a file named notes.txt with the following content:
This is the first line.
This is the second line.
EOF
v. Use the > operator to overwrite a file named output.txt with the result of the pwd command
vi. Write a command to display today's date in the format YYYY-MM-DD.
vii. Write a command to display the date 7 days ago.
viii. Use the date command to display the current time only in the format HH:MM:SS.
ix. Use the ls and wc commands with a pipe to count the number of files in the current directory.
x. Write a command to list files in the current directory, sort them, and display only the top 5 results using a pipe.

WEEK 2: SHELL PROGRAMS-1

1) Write and execute shell scripts for the following (Using simple shell commands):

a) Input two values. Perform the arithmetic operations +, -, * and / on them. Display the results.

b) Input a file name. Display its attributes and contents.

c) Input a file name. Copy it to another file and then rename it (The names of the second and third files should be input).

2) Write and execute shell scripts for the following (Using *if*, *while*, *until* and *break* statements):

d) Input a number. Output whether it is odd or even.

e) Input two numbers. Compare them using *if* statement and output their relative magnitudes.

f) Input two numbers. Display all numbers between (including) them using *while* statement. Calculate their sum using *until* statement and display it.

3) Write and execute shell scripts for the following (Using *case* statement):

g) Input a number. Output whether it is zero or non-zero.

h) Input a character. Output whether is an upper-case alphabet, a lower-case alphabet, a digit or a special character.

WEEK 3: SHELL PROGRAMS-2 with FILE-RELATED COMMANDS

1) File-related commands: Test the function of each of the following.
   a) File permissions: –r, –w, –x
   b) File existence: –f, –d, –e, –s
2) Write and execute shell scripts for the following (Using file related commands):
   c) List all files in the directory which have a read permission.
   d) Input a file name. Output whether it has read, write and execute permissions or not.
   e) Accept a file name. Check whether it has write permission. If yes, append some new text to it, otherwise display an error message.
3) Write and execute shell scripts for the following (Using file related commands):
   f) Count and display the number of ordinary files, hidden files and sub-directories present in the working directory.
   g) Display the names of all files in the working directory which have size greater than 0.

WEEK 4: SHELL PROGRAMS-3 with ADDITIONAL LINUX COMMANDS

1) Additional Linux Commands: Test the function of each of the following.

a) Transforming data: cut, sort, tr

b) Pattern matching: grep

c) Viewing parts of a file: head, tail

2) Write and execute shell scripts for the following (Using the above commands).

d) Display the names of all files whose names start with the word 'hello'.

e) Accept a file name and a pattern. Output whether the pattern exists in the file or not.

f) Accept a file name. Display its access privileges for all types of users.

WEEK 5: PROCESSES

1) Process related Commands: Study the meaning / function of the following.

a) Process identification: PID, getpid()

b) Background process: &

c) Parent and child processes: fork(), sleep(), wait()

2) Write and execute C programs for the following.

a) Create a child process. Display different messages in the parent and child process. Also display their process ID.

b) Accept an array of integers. Display the unsorted array in the parent process. Create a child process. Sort and display the sorted array in the child process.

WEEK 6: MID-TERM LAB EXAMINATION

WEEK 7: PROCESS SCHEDULING-1

1) Write and execute C programs for simulating the following scheduling algorithms. In each case, calculate the waiting time and turn-around time for each process and the average waiting and turn-around times.

a) FCFS

b) SJF

WEEK 8: PROCESS SCHEDULING-2

1) Write and execute C programs for simulating the following scheduling algorithms. In each case, calculate the waiting time and turn-around time for each process and the average waiting and turn-around times.

a) Pre-emptive SJF

b) Priority

WEEK 9: PROCESS SYNCHRONIZATION AND DEADLOCKS

1) Write and execute C programs for simulating the following.

a) Semaphore operations (for producer-consumer problem)

b) Banker's algorithm (for deadlock avoidance)

WEEK 10: PAGE REPLACEMENT

1) Write and execute C programs for simulating the following page replacement algorithms. In each case, calculate the number of page faults.

a) FIFO

b) LRU

c) Optimal

WEEK 11: DISK SCHEDULING

1) Write and execute C programs for simulating the following disk scheduling algorithms.

a) FCFS
b) SSTF
c) SCAN
d) LOOK

WEEK 12: End Semester Laboratory Examination

ADDITIONAL EXERCISES

SL. NO.	PROGRAM DESCRIPTION
1	Write Linux shell scripts for the following: a) Accept an integer and find its reverse. Also check for palindrome. b) Accept an integer „n" and find the sum of the series 1! + 2! + 3! + …… + n! c) Accept an integer and check whether it is a prime or not. d) Write a shell script to accept „n" integers and count the number of +ves, -ves and zeroes separately. Also calculate the sum of +ves and - ves.
2	Write Linux shell scripts for the following: a) Accept a word and check whether it begins with lower case vowel or capital vowel, ends with a digit or whether it is a three letter word b) Accept a string. Check whether it is a palindrome or not. c) Accept a string. Count the number of vowels, consonants, digits, spaces and special characters in it. d) Write a shell script to accept student name and marks in 3 subjects through command line arguments. Find the total marks, result and grade (depending on the total marks).
3	Write a menu driven shell script for the following.

a) Rename a file (check for the existence of the source file)
b) Display the current working directory
c) List the users logged in.
d) Append the contents of a file to another file (Display the message if the file doesn‟t exist in the directory).

4 Write a shell script to accept your option for deleting (-d) or for copying (-c) a file and file name(s) through command line arguments

(e.g. for deletion: $sh filename –d file1
for copying: $sh filename –c file1 file2)

and check for the following:

a) Check whether the given arguments are sufficient for the selected option.
b) File to be copied or deleted must be present in the directory.
c) While copying, if the destination file already exists, prompt for overwriting.

5 Write a shell script to accept many filenames through command line. Do the following for each filenam

a) If it is an ordinary file, display its content and also check whether it has „execute‟ permission.
b) If it is directory, display the number of files in it.
c) If the file/directory does not exist, display a message.

6 Write a menu driven shell script for the following.

a) List of directory having all the permission.
b) Count the number of directories and files separately.
c) List the names of all files / directories in the present working directory which have the specified pattern.

7 Write C program to simulate Round-Robin Scheduling

8 Write and execute C programs for the following.

a) Accept two file names (first one is an existing file and the second one it to be created). Copy the content of the existing file to the new file. Display the contents of both the files and also the number of characters transferred.

b) Accept a file name. Display the content of the file starting from the given position specified by the user.

c) Create a file with the following permissions: Read-Write-Execute for the user, Read-Write for the group, and Read for others. Input a sequence of characters from the keyboard until # is read and write them into the file. Display the content of the file.

9 Write and execute C programs for simulating the following disk scheduling algorithms.

a) C-SCAN
b) C-LOOK

REFERENCE BOOKS

1. Abraham Silberschatz and Peter Baer Galvin, "Operating Systems Concepts", 5th Edition, Addison Wesley Publications, 2002.
2. Yashvant P Kanetkar, "Unix Shell Programming", 1st Edition, BPB Publications, 2008.
3. Meetha Gandhi, Tilak Shetty and Rajiv Shah, "The C Odyssey UNIX – The Open Boundless C", 1st Edition, BPB Publications, 2008.
4. Eric Foster-Johnson, John C Welch and Micah Anderson, "Beginning Shell Scripting – Covering Linux, Unix, Windows and Mac", 1st Edition, Wiley Publishing Inc., 2005.
5. Richard Blum, "Linux – Command Line and Shell Scripting", Wiley India Pvt. Ltd., 2011.

Guide to using the vi editor

The **vi editor** is a powerful text editor available on almost all Unix/Linux systems. It's lightweight, fast, and highly efficient for editing text files, scripts, and configuration files.

Modes of vi Editor

1. **Command Mode** (default mode):
    - Used for navigation, text manipulation, and executing commands.
    - Press Esc to return to Command mode from any other mode.
2. **Insert Mode**:
    - Used for inserting text.
    - Enter Insert mode by pressing i (insert at cursor), a (append after cursor), or o (open a new line below).
3. **Visual Mode**:
    - Used for selecting and manipulating blocks of text.
    - Enter Visual mode by pressing v (character-wise selection), V (line-wise selection), or Ctrl+v (block selection).
4. **Command-Line Mode**:
    - Used for saving files, quitting, searching, and more.
    - Access by typing : from Command mode.

Basic vi Commands

File Operations:

- :w – Save the file.
- :q – Quit vi.
- :wq or ZZ – Save and quit.
- :q! – Quit without saving.
- :w filename – Save the file with a new name.
- :r filename – Insert the content of another file.

Navigation:

- h – Move left.
- l – Move right.
- j – Move down.
- k – Move up.
- gg – Go to the beginning of the file.
- G – Go to the end of the file.
- :n – Go to line number n.

Text Insertion & Deletion

- i – Insert before the cursor.
- I – Insert at the beginning of the line.
- a – Append after the cursor.
- A – Append at the end of the line.
- x - Deletes the character under the cursor.

- X- Deletes the character before the cursor.
- dd - Deletes the entire current line.
- n dd - Deletes n lines starting from the current line.
-

Copy-paste:

- yy : Yanks (copies) the entire current line.
- n yy : Yanks n lines starting from the current line
- yw: Yanks from the cursor to the end of the current word.
- yW: Yanks the entire word (ignoring punctuation).
- p: Puts (pastes) the yanked or deleted text after the cursor.
- P: Puts (pastes) the yanked or deleted text before the cursor.

Show line numbers: inside the vi editor command mode type **":set number"**