MANIPAL INSTITUTE OF TECHNOLOGY

(A constituent Institute of Manipal University, Manipal)

DEPARTMENT OF MASTER OF COMPUTER APPLICATION

LABORATORY COURSE PLAN

Department	:	Master of Computer Application
Subject	:	Linux Programming Lab (MCA 4162)
Semester & branch	:	I Semester M.C.A.
Name of the faculty	:	Dr. Sandhya Parasnath Dubey
No. of contact hours/week	:	3

Submitted by:	Approved by:
Dr. Sandhya Parasnath Dubey	Dr. Karunakar A. Kotegar
(Signature of the faculty)	(Signature of HOD)
Date: 19-11-2021	Date: 19-11-2021

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. **Bi-weekly 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

- 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.

EVALUATION PLAN

Continuous Evaluation: **3 x 20 = 60 marks**

- **3 bi-weekly evaluations for 20 marks each.**

Observation Book – 5 marks
Program Execution – 5 marks
Test – 10 marks

End-Semester Examination 40 marks

Total 100 marks

COURSE OBJECTIVES

At the end of the course, the student should

- Have a complete understanding of the use of various Linux commands.
- Be able to write shell programs for any given problem.
- Have a thorough knowledge of the meaning of various Operating System concepts and be able to implement / simulate them.

WEEK 1: BASIC LINUX and vi EDITOR COMMANDS

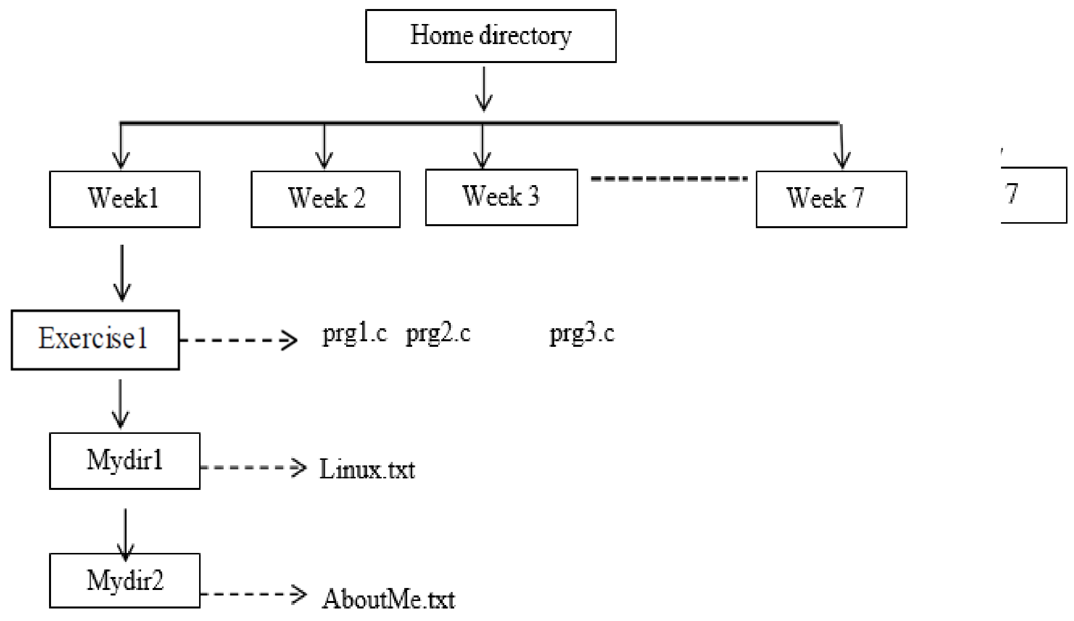
1 Basic Linux Commands: Test the function of each of the following:

a) Online help: man, help
b) Standard output: echo
c) Date
d) clear
e) Directory related commands: pwd, mkdir, cd, rmdir, mvdir
f) File related commands: ls, cat, more, cp, mv, rm, touch

2 Exercises on vi editor and Linux commands

Exercise1: Linux & vi editor

We will be using the following file structure during your OS lab sessions.
Home directory example: /OSmca001/



1. Create a sub-folder under your Week1 folder called **Exercise1**.

1. Create a subfolder under Exercise1 called **Mydir1**.
2. Create a subfolder under Mydir1 called **Mydir2.**
3. Create a file called **AboutMe.txt** using vi editor. Type out 10 lines on the topic“About Me” in this text file. (You must be in Mydir2 sub directory)
4. Now go back to subdirectory **Mydir1.**
5. Create a new file called **Linux.txt** with the following information: (You must be in sub-directory Mydir1).

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. 6) perform the following editor operations:

6. Remove blank lines wherever the *** is found
7. Replace all occurrences of the word "Linux" with "Ubuntu"
8. Save and return to command line prompt
9. Use command to view **"Linux.txt"** at the command prompt without using vi
10. Rename **"Linux.txt"** with the name **"Ubuntu.txt"**
11. Make a copy of the same text file in the name **"Linux1.txt"** at command prompt
12. Now return back to the folder "Exercise1" under Week1. All the remaining exercises must be in Exercise1 sub directory.

WEEK 2: ADDITIONAL LINUX COMMANDS

1 Linux Commands: Test the function of each of the following:

a) Wildcards: *, ?
b) Grep
c) Input/output redirection: >, >>, <, <<
d) Pipes |
e) Wc
f) Command substitution ``
g) Sequencing ;
h) Conditional sequence; && and ||
i) Sort
j) File and directory permissions (chmod):
   a. Users and ownership
   b. Groups and Changing group ownership
   c. File permissions and Changing file permission

2 Exercises on above commands
Write shell commands for the following.

a. Create a directory in your home directory having 2 subdirectories.
b. In the first subdirectory, create 3 different files with different content in each of them.
c. Copy the first file from the first subdirectory to the second subdirectory.

d. Create one more file in the second subdirectory, which contains the command listing the number of users and number of files.
e. To list all the files which starts with either ‘a’ or ‘A’.
f. Display the output if the compilation of a sort program succeeds else display an
error message.
a. Display the first five and last five lines of a given file.
b. Redirect the output of commands ‘pwd’, ‘date’ and ‘ls’ in succession to a file.

WEEK 3: SIMPLE SHELL PROGRAMS

1 Linux Commands: Test the function of each of the following:

a) Who
b) Whoami
c) Logname
d) Uname
e) Tty
f) Time
g) Bc
h) expr

2 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).

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

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

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

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

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

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

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

WEEK 4: FILE-RELATED COMMANDS AND SHELL PROGRAMS

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):

a) List all files in the directory which have a read permission.

b) Input a file name. Output whether it has read, write and execute permissions or not.

c) 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):

a) Count and display the number of ordinary files, hidden files and sub-directories present in the working directory.

b) Display the names of all files in the working directory which have size greater than 0.

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: PROCESS SCHEDULING

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 7: 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 8: END SEMESTER LABORATORY EXAMINATION

ADDITIONAL EXERCISES

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

a) Line, word and character counts: wc

b) Transforming data: cut, sort, tr

c) Pattern matching: grep

d) Viewing parts of a file: head, tail

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

a) Display the names of all files whose names start with the word „hello‟.

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

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

PROCESS SCHEDULING

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

PROCESS SYNCHRONIZATION AND DEADLOCKS

2 Write and execute C programs for simulating the following.

a) Semaphore operations (for producer-consumer problem)

b) Banker’s algorithm (for deadlock avoidance)

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

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 filename
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.
d) Exit.

7 Write a 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.

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.