CMR ENGINEERING COLLEGE

(Approved by AICTE- New Delhi, Affiliated to JNTUH)
Kandlakoya(V), Medchal Road, Hyderabad

Department of Computer Science & Engineering (Data Science)

LIFT Planner (Lab Improvement for Future Trends)

On

OPERATING SYSTEMS LAB

2025-2026

BY
M.Chandana
Assistant Professor

OPERATING SYSTEMS LAB

Sl. No	CONTENT
1	OBJECTIVES AND RELEVANCE
2	SCOPE
3	PREREQUISITES
4	SYLLABUS
5	LAB SCHEDULE
6	SUGGESTED BOOKS
7	WEBSITES (USEFUL LINKS)
8	EXPERT DETAILS
9	LEAD EXPERIMENTS
10	MAPPING OF LAB WITH PROJECT/CONSULTANCY/R & D PROPOSALS
11	PBL PROJECTS
12	INDUSTRIAL VISITS (IIP – INNOVATIVE INDUSTRIAL LEARNING PROGRAM)
13	SHADOW ENGINEERING
14	CALIBIRATION , TESTING AND INSPECTION
15	MAINTAINANCE SCHEDULES
16	TROUBLESHOOTING
17	ASSESSMENT AND ACCREDITATION PROCEDURE AS PER NABL
18	VIRTUAL LABS REFERENCE LINKS

1. OBJECTIVES AND RELEVANCE

Upon successful completion of this Lab the student will be able to:

- ✓ To able to know about design aspects of operating system concepts through simulation.
- ✓ To learn the principles of IPC Mechanisms.
- ✓ To practice the handling, I/O System calls of UNIX/LINUX Operating systems.
- ✓ Introduce basic Unix commands, system call interface.

2. SCOPE

This lab subject is cover with Scheduling Algorithms, deadlock management and handling of I/O System calls of UNIX/LINUX Operating systems

3. PREREQUISITES

This subject recommends continuous practice flexibility of C programming.

4. Autonomous Syllabus-R22

1. Write C programs to simulate the following CPU Scheduling algorithms

   a) FCFS b) SJF c) Round Robin d) priority

2. Write programs using the I/O system calls of UNIX/LINUX operating system (open, read, write, close, fcntl, seek, stat, opendir, readdir)

3. Write a C program to simulate Bankers Algorithm for Deadlock Avoidance and Prevention.

4. Write a C program to implement the Producer – Consumer problem using semaphores using UNIX/LINUX system calls.

5. Write C programs to illustrate the following IPC mechanisms

   a) Pipes b) FIFOs c) Message Queues d) Shared Memory

6. Write C programs to simulate the following memory management techniques

a) Paging b) Segmentation

7. Write a C program to simulate page replacement algorithms

a)FCFS

b)LRU

c) Optimal Page Replacement Algorithm

9.LEAD PROGRAMS

1. Write a C program to implement Dining-Philosophers problem using semaphore
2. Write a program to implement contiguous memory allocation using First fit.

LAB EXPERIMENTS:

Week No.	Name of the Program	No. of Hours required	UNIT NUMBER
1	Introduction & Basic Commands	1	I
2	1.Write C programs to simulate the following CPU Scheduling algorithms a) FCFS b) SJF	1	II
3	c) Round Robin d) priority	1	II
4	2.Write programs using the I/O system calls of UNIX/LINUX operating system (open, read, write, close, fcntl, seek, stat, opendir, readdir)	1	I
5	3. Write a C program to simulate Bankers Algorithm for Deadlock Avoidance and Prevention.	2	II
6	4. Write a C program to implement the Producer – Consumer problem using semaphores using UNIX/LINUX system calls	1	II
7	5.Write C programs to illustrate the following IPC mechanisms	1	III

	a) Pipes b) FIFOs c)Message Queues d) Shared Memory		
8	6. .Write C programs to simulate the following memory management techniques a) Paging b) Segmentation	1	IV
9	7. Write a C program to simulate page replacement algorithms a)FCFS b)LRU c) Optimal Page Replacement Algorithm	2	IV
10	LEAD Experiments	3	
	Total no of HOURS required to complete syllabus	**14**	

EXPERIMENT NO.1

Introduction of CPU scheduling algorithms – FCFS, SJF, Round Robin, Priority.

OBJECTIVE:

The main objective is to understand clearly about all the process scheduling algorithms.

PRE REQUISITES:

Basic scheduling process technique and communication techniques

DESCRIPTION:

1. Introduction to process techniques.
2. Introduction to scheduling techniques.
3. Implementation of scheduling algorithms.

APPLICATIONS:

1. These mechanisms can be practically applied in various Operating Systems.
2. These techniques are useful to improve the throughput.

EXPERIMENT NO.2

Introduction of I/O System calls of UNIX/LINUX

OBJECTIVE:

The main objective is to understand clearly about all the basic I/O system calls in UNIX/LINUX.

PRE REQUISITES:

Basic of system calls.

DESCRIPTION:

1. Introduction to I/O system calls.
2. Implementation of open, close, read, write, etc.

APPLICATIONS:

1. These mechanisms can be practically applied in various Operating Systems.
2. These techniques are useful to improve the throughput.

EXPERIMENT NO. 3

Deadlocks

OBJECTIVE:

The main objective is to understand clearly about Deadlocks technique

PREREQUISITES:

Basic knowledge of Deadlock characteristics.

DESCRIPTION:

1. Introduction to Deadlocks.
2. Deadlock detection
3. Deadlock Avoidance
4. Data Recovery from Deadlocks.
5. Deadlock prevention.

APPLICATION:

1. It gives the correct idea of Deadlocks.
2. It helps the student to know about Deadlock detection, Deadlock avoidance & prevention techniques.

EXPERIMENT NO.4 & 5

Introduction to Producer Consumer problem & Pipes, Inter-processor communication and semaphores

OBJECTIVE:

The main objective is to understand clearly about IPC Mechanisms.

PRE REQUISITES:

Basic communication techniques

DESCRIPTION:

1. Introduction to communication techniques.
2. Implementation of inter process communication with pipes
3. Implementation of semaphores

APPLICATIONS:

3. These mechanisms can be practically applied in various Operating Systems.
4. These techniques are useful to improve the throughput.
5. These techniques are used to solve communication problems and Producer - Consumer problem

EXPERIMENT NO. 6

Introduction of paging techniques and page replacement algorithms

OBJECTIVE:

The main objective is to understand clearly about all the page replacement algorithms and

Allocation techniques.

PRE REQUISITES:

Basic paging techniques.

DESCRIPTION:

1. Introduction to paging techniques.
2. Introduction to page replacement and allocation technique.
3. Implementation of page replacement algorithms.

APPLICATIONS:

1. These mechanisms can be practically applied in various secondary storage techniques such as disk storage.

EXPERIMENT NO. 7

Write C programs to simulate Page replacement policies a) FCFS

b) LRU

c) Optimal

DESCRIPTION:

1. Introduction to page replacement algorithms.
2. Introduction to page faults.
3. Implementation of page replacement algorithms.

5.A) LAB SCHEDULE:

CYCLE 1:

Batches	week-1	week-2	week-3	week-4	week-5	week-6	week-7	week-8
B1	Demo	Exp.1	Exp.1	Exp.2	Exp.3	Exp.4	Test	Exp.5
B2	Demo	Exp.1	Exp.1	Exp.2	Exp.3	Exp.4	Test	Exp.5

Batches	week-9	week-10	week-11	week-12	week-13	week-14
B1	Exp.5	Exp.6	Exp.6	Test	LEAD 1 &2	Lab internal
B2	Exp.5	Exp.6	Exp.6	Test	LEAD 1 & 2	Lab internal

(B) VIVA SCHEDULE: The viva schedule should be planned prior to the lab experiment.

CYCLE 1: **ROUND -** 1

Batches	week-1	week-2	week-3	week-4	week-5	week-6	week-7	week-8	week-9	week-10	week-11	week-12	week-13	week-14
B1,B2	viva	viva	viva	viva	viva	viva	Viva	viva	viva	viva	Viva	Viva	viva	viva

ROUND - 2

Batches	week-1	week-2	week-3	week-4	week-5	week-6	week-7	week-8	week-9	week-10
SG1	viva									
SG2			viva							

SG3					viva					

LAB EXTERNAL:			
S.NO	Write up	Final Evaluation	Viva
1	**1. Aim 2. Procedure 3. Program 4. Expected output.**	**Based on correctness of the program and Results**	**Based on understanding of Experiment and theoretical questions in the related subjects**
Marks	**35**	**15**	**10**
Total Marks-60			

SG4							viva			
SG5									viva	

*SG: Selected Group with a maximum of 6 or 12 students

(C) Scheme of Evaluation:

The scheme of evaluation for internal and external exams is as follows:

LAB INTERNAL:							
Day to Day Evalution-15					**Internal Exam-15**		
Uniform	**Observation & Record**	**Performance Of the Experiment**	**Result**	**Viva**	**Write up**	**Execution & Results**	**Viva**
Marks-2	**Marks-2**	**Marks-3**	**Marks-3**	**Marks-5**	**Marks-5**	**Marks-5**	**Marks-5**
Total Marks-30							

6. SUGGESTED BOOKS:

1.Operating System Principles- Abraham Silberchatz, Peter B. Galvin, Greg Gagne 7th Edition, John Wiley

2. Advanced programming in the Unix environment, W.R.Stevens, Pearson education.

7 . WEBSITES

1. https://en.wikipedia.org/wiki/Operating_system
2. https://nptel.ac.in/courses/106/108/106108101/#

8. EXPERTS DETAILS

The Expert Details which have been mentioned below are only a few of the eminent ones known Internationally, Nationally and Locally. There are a few others known as well.

INTERNATIONAL

1.Martin C. Rinard, MIT Computer Science and Artificial Intelligence Laboratory ,The Stata Center, Building 32-G82832 Vassar Street, Cambridge, MA 02139 Email: rinard@lcs.mit.edu .

2. Abraham Silberschatz, Professor and Chair of Computer Science at Yale University.

3.Andrew S. Tanenbaum, American computer scientist and professor of computer science,

NATIONAL

1. Matthew Jacob T. Ph.D. (SERC) Professor. Performance aspects of Computer Architecture, Operating Systems, Parallel Processing. Email: mjt@serc.iisc.ernet.in.

2. Mrs. A. Kalaivani MCA., M. Phil Asst. Professor, NIITM, NEW DELHI.

REGIONAL

1. Dr Raja shekar, professor,CSE VNRVJIET Hyderabad, Email: rajasekarm9@gmail.com.

9849296429

2. Dr prashanth Rao Professor, IT Anurag

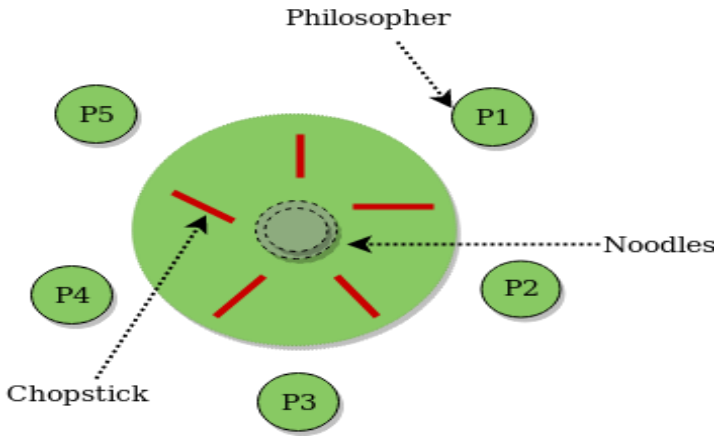
university,Hyderabad,Email:prashanthrao@cvsr.ac.in,9490232922

9. LEAD EXPERIMENTS

1. Dining Philosopher Problem Using Semaphores

DESCRIPTION:

The Dining Philosopher Problem – The Dining Philosopher Problem states that K philosophers seated around a circular table with one chopstick between each pair of philosophers. There is one chopstick between each philosopher. A philosopher may eat if he can pickup the two chopsticks adjacent to him. One chopstick may be picked up by any one of its adjacent followers but not both.



Semaphore Solution to Dining Philosopher –
Each philosopher is represented by the following pseudocode:

```
process P[i]
 while true do
   {  THINK;
      PICKUP(CHOPSTICK[i], CHOPSTICK[i+1 mod 5]);
      EAT;
      PUTDOWN(CHOPSTICK[i], CHOPSTICK[i+1 mod 5])
   }
```

Code

```
#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#define N 5

#define THINKING 2

#define HUNGRY 1

#define EATING 0

#define LEFT (phnum + 4) % N

#define RIGHT (phnum + 1) % N

int state[N];

int phil[N] = { 0, 1, 2, 3, 4 };

sem_t mutex;

sem_t S[N];
```

```
void test(int phnum)
{
    if (state[phnum] == HUNGRY
        && state[LEFT] != EATING
        && state[RIGHT] != EATING) {
        // state that eating
        state[phnum] = EATING;
        sleep(2);
        printf("Philosopher %d takes fork %d and %d\n",
                    phnum + 1, LEFT + 1, phnum + 1);
        printf("Philosopher %d is Eating\n", phnum + 1);
        // sem_post(&S[phnum]) has no effect
        // during takefork
        // used to wake up hungry philosophers
        // during putfork
        sem_post(&S[phnum]);
    }
}
// take up chopsticks
void take_fork(int phnum)
```

```
{
    sem_wait(&mutex);
    // state that hungry
    state[phnum] = HUNGRY;
    printf("Philosopher %d is Hungry\n", phnum + 1);
    // eat if neighbours are not eating
    test(phnum);
    sem_post(&mutex);
    // if unable to eat wait to be signalled
    sem_wait(&S[phnum]);
    sleep(1);
}
// put down chopsticks
void put_fork(int phnum)
{
    sem_wait(&mutex);
    // state that thinking
    state[phnum] = THINKING;
    printf("Philosopher %d putting fork %d and %d down\n",
          phnum + 1, LEFT + 1, phnum + 1);
```

```
    printf("Philosopher %d is thinking\n", phnum + 1);
    test(LEFT);
    test(RIGHT);
    sem_post(&mutex);
}
void* philospher(void* num)
{
    while (1) {
        int* i = num;
        sleep(1);
        take_fork(*i);
        sleep(0);
        put_fork(*i);
    }
}
int main()
{
    int i;
    pthread_t thread_id[N];
    // initialize the semaphores
```

```
    sem_init(&mutex, 0, 1);
    for (i = 0; i < N; i++)
     sem_init(&S[i], 0, 0);
    for (i = 0; i < N; i++) {
        // create philosopher processes
        pthread_create(&thread_id[i], NULL,
                       philospher, &phil[i]);
        printf("Philosopher %d is thinking\n", i + 1);
    }
    for (i = 0; i < N; i++)
        pthread_join(thread_id[i], NULL);
}
```

2) **Write a program to implement contiguous memory allocation using First Fit**

OBJECTIVE: To implement contiguous memory allocation using First Fit.

Program

```
#include&lt;stdio.h&gt;
void main()
{
int bsize[10], psize[10], bno, pno, flags[10], allocation[10], i, j;
```

```
for(i = 0; i &lt; 10; i++)

{

flags[i] = 0;

allocation[i] = -1;

}

printf(&quot;Enter no. of blocks: &quot;);

scanf(&quot;%d&quot;, &amp;bno);

printf(&quot;\nEnter size of each block: &quot;);

for(i = 0; i &lt; bno; i++)

scanf(&quot;%d&quot;, &amp;bsize[i]);

printf(&quot;\nEnter no. of processes: &quot;);

scanf(&quot;%d&quot;, &amp;pno);

printf(&quot;\nEnter size of each process: &quot;);

for(i = 0; i &lt; pno; i++)

scanf(&quot;%d&quot;, &amp;psize[i]);

for(i = 0; i &lt; pno; i++) //allocation as per first fit

for(j = 0; j &lt; bno; j++)

if(flags[j] == 0 &amp;&amp; bsize[j] &gt;= psize[i])

{

allocation[j] = i;
```

```
flags[j] = 1;

break;

}

//display allocation details

printf(&amp;quot;\nBlock no.\tsize\t\tprocess no.\t\tsize&amp;quot;);

for(i = 0; i &amp;lt; bno; i++)

{

printf(&amp;quot;\n%d\t\t%d\t\t&amp;quot;, i+1, bsize[i]);

if(flags[i] == 1)

else

} }
```

10. MAPPING OF LAB WITH PROJECT/CONSULTANCY/R & D:

PROPOSAL 1: Scheduling Techniques

To implement this simplified process management scheme using 2 different scheduling techniques: prioritized round robin and shortest remaining time.

Your project is to implement this simplified process management scheme using 2 different scheduling techniques: prioritized round robin and shortest remaining time. You also need to keep track of various statistcal data for each scheduling technique and job: total time each process was in each state, number of jobs run, total elasped time, total elapsed time per job (throughput), and longest and shortest time taken for any job in the system.

The system will have the following behavior. You must have a way to keep time intervals (clock ticks) during the execution of the system. A simple approach to this problem might be to use a loop and for each iteration, increment the clock. Seven things should be done in the scheduling loop during each cycle, the first six of which pretty much map directly to the transitions of the diagramabove. For this simulation jobs will do no "real" work; i.e. your simulation will implement what happens when the CPU invokes the operating system process scheduler. You can think of this project as a function call inside the operating system. A function call that updates all job states and sets up the next job to run, so that when the function call completes, the CPU has next job and can begin to run it.

Since our simulated jobs are doing no "real work", we have a little problem. If they are not actually executing code, they cannot request I/O. If they cannot request I/O, we can't transition from *running* to *sleeping on I/O*. And since there is no real I/O going on, I/O can never complete, so we cannot transition from *sleeping on I/O* to a *ready to run*. This is a problem. But we have a solution. When a job is in the *running* state you will have a function that randomly decides whether or not the job has requested I/O. When a job is in the "waiting for I/O" state, you will have a function that randomly decides whether or not I/O has completed for that job.

PROPOSAL 2: A "Slower" File System

Objectives

There are two objectives to this assignment:

1. To understand how file systems work, specifically the directory hierarchy and storage management.
2. To understand some of the performance issues file systems deal with.

Overview

In this project, we will build a user-level library, libFS, that implements a good portion of a file system. Your file system will be built inside of a library that applications can link with to access files and directories. Your library will in turn link with a layer that implements a "disk"; we provide this library, LibDisk, which you must use.

LibFS Specification

We'll start by describing the LibFS API to the file system. There are three parts to the API: two generic file system calls, a set of calls that deal with file access, and a set of calls that deal with directories, and

Applications (e.g., your own test applications, and certainly our test applications) will link with LibFS in order to test out your file system.

Your library will be tested on how functions and also particularly on how it handles errors. When an error occurs (each possible error is specified below in the API definition), your library should set the global variable **osErrno** to the error described in the API definition below and return the proper error code. This way, applications that link with your library have a way to see what happened when the error occurred.

11. PBL PROJECTS

1. Redundant array of independent Disks

2. Implementing a basic system call interface

12. PROPOSALS INDUSTRIAL VISIT

Relevant IV (Industrial Visit) (VIP= Visit of Industrial Proposals)

If students visited different software development companies like TCS, WIPRO, INFOSYS.

S.No	Type of industry	Nature of industry	Date of visit (Tentative Date)	No. of students participated	Year/branch	Remarks
1	MNC	software		20 to 30	CSE	
2	MNC	software		20 to 30	CSE	

13. SHADOW ENGINEERING

GUIDELINES FOR SHADOW ENGINEERING (VIP)

INDUSTRIAL VISITS (IIP – INNOVATIVE INDUSTRIAL LEARNING PROGRAM)

OBJECTIVES OF SHADOW ENGINEERING:

1. To uplift the knowledge of the students related to laboratories.
2. To improve the industry-college interactions.
3. To create industry like environment for all the students in order to make future Assignment.
4. This program leads to matrixing with the students.

TABLE 2: INDUSTRIAL TRAINING (Shadow Engineering)
(Career Visit Approval)

S.No	Name of the Course	Nature of industry	Duration of Training	Authority	Date of Training/Certificate No.	Remarks
1	Operating Systems	Software	10 hours			

14. Calibration, Testing & Inspection (Method & Tools to be used)

Calibration

There is no need of any equipment to be calibrated.

Testing & Inspection:

Installation: Aim of this concept is to make and maintain installation procedure for a new equipment or already existing equipment

Linux OS Installation steps :(Red Hat Linux GUI)

Step 1 – Insert the Red Hat Linux DVD into the DVD-drive of your computer. As soon as the following screen pops up, press 'Enter' to install Red Hat Enterprise Linux (RHEL) through GUI mode.

Step 2- RHEL installer would then prompt you to conduct a check as to whether the CD media from which you're installing is functioning correctly or not. Choose 'Skip', press enter and the installation would begin.

Step 3- Next, we need to select the language – English or any other language as per your preference, and then press 'Next'.

Step 4- In this step, the RHEL installer would ask you about the appropriate type of keyboard for the system. We take the 'US English' keyboard; you can pick any other option depending on the type of your keyboard. Then press 'Next' to move to the next step.

Step 5- Next, the installer would ask for an 'installation number' if you wish to install full set of Red Hat functionalities. Enter the installation number and press 'OK' if you have an officially licensed installation number (for corporate clients that buy Red Hat's backup support and full features).

Step 6- The Red Hat installer would then require you to create partitions in your computer's hard disk for the installation. You can do it in four ways but the simplest way is to select 'Use free space on selected drives and create default layout' as this option will not affect any other OS residing in your system.

Step 7- In this step you must create the required system partitions and mount points such as '/boot', '/home', 'swap' etc which are required for the Linux's proper functioning.

Step 8- This step pertains to the default OS that will be loaded by the GRUB loader.

Step 9- This step pertains to the network settings of the Linux system that you are going to install. You can select the Ethernet devices through which the system would communicate with other devices in the network.

Step 10- The next step is to adjust the system clock to your particular time zone. Select your time zone and then click 'Next'.

Step 11 – This is a very important step that deals with the root (super-user) password for the system. Type the password and confirm it and then click next.

Step 12 – The RHEL installer would then prompt you about if you wish to install some extra 'Software Development' or 'Web Server' features. By default, keep it at 'Customize later' and press 'Next'.

Step 13- This next step will initiate the installation of Red Hat Linux, press 'Next' to begin the process.

Step 14- Upon the completion of installation you should the following screen. Press Reboot and you'd be ready to use your newly installed Red Hat Linux OS.

Commands in Linux for creating, running, testing C programs

Creating program:

vi program name .c

vi commands:

for saving the commands is: esc: wq

for editing the command is: esc: i

for compiling the command is :gcc programname.c –o programname

for running the command is: ./programname

15 & 16. MAINTAINANCE SCHEDULES AND TROUBLESHOOTING

Maintenance:

Maintenance and trouble shooting of each equipment in a laboratory must follow the following guidelines:

Maintenance Schedules:

(1) Preventive Maintenance Schedules of lab will be decided by lab in charge along with concerned HOD. The details of schedule should be recorded in the following template of format.

S.No.	Software	Date of Maintenance	Type of Activity	Remarks
1	Operating System		Soft wares installed(PUTTY)	

(2) Maintenance Reports should be duly signed by in charges as well as HODs and duly approved by Principal periodically.

17. ASSESSMENT AND ACCREDITATION PROCEDURE AS PER NABL

Accreditation is the formal recognition, authorization and registration of a laboratory that has demonstrated its capability, competence and credibility to carry out the tasks. It provides the feedback to laboratories as to whether they are performing according to the technical competence

as per guidelines of NABL (National Accreditation Board for Testing and Calibration Laboratories).

The laboratory should carry out the following important tasks towards getting ready for accreditation from NABL.

1. Preparation of methodology for each experiment
2. Preparation of Standard Operating procedure for each equipment
3. Preparation of Laboratory Manual as per the guidelines specified by Combined Lab Team(CLT) headed by Principal/HOD/Dean/in charge
4. Ensure Effective environmental conditions (temperature, humidity, storage and placement) in the laboratories by implementing proper housekeeping and cleaning of the equipments from dust, dirt etc.
5. Ensure Calibration of instruments/equipment (Only NABL accredited authorized laboratories provide calibration).
6. All the details of Calibration should be included in the format specified exclusively for calibration procedure.
7. Ensure proper implementation of all the documents, formats to be included in the lab manual.
8. Impart training for all the technicians working in labs about the importance of documentation, log sheets, operating procedure of the lab.
9. Incorporate Internal Lab audits for effective functioning of the laboratories. Audits may be once in a month or 3 months or at the end of the semester. The audit schedule will be decided by the Chairman and Principal of the CLT team.
10. Auditors should submit the detailed report of each lab duly signed by them to the Principal.
11. Each lab should maintain all the bills/invoices of each instrument or equipment in a separate file.
12. All the stock registers either consumable or non consumable should be updated whenever any purchases of consumables or equipment takes place.
13. All the safety precautions should be properly displayed in front of each lab.
14. All the LEAD experiments should be maintained separately in a record / recorded in a separate folder.
15. Based on the Pre Assessment report submitted by the auditor, corrective actions should be carried out by each lab in charge and that must be forwarded to concerned HOD and Principal.

18. VIRTUAL LABS REFERENCE LINKS

1.http://vlabs.iitb.ac.in/vlabs-dev/vlab_bootcamp/bootcamp/CRUX/labs/exp1/references.html

2. http://www.oslab.ch/modules.php