LAB PLAN

Subject: Operating System

S.N.	Lab Topics	Lab hours
1	Introduction to VIM editor, GCC compiler, Linux Terminal	4
2	Interacting with the execution environment	4
3	Using “System” system call	1
4	Process creation using fork() system call	4
5	Demonstration of process switching and process termination	2
6	Implementation of Threads	2
7	Demonstration of IPC using lock variable	2
8	Demonstration of IPC using strict alternation	2
9	Implementation of SJF scheduling algorithm	2
10	Implementation of RR scheduling algorithm	2
11	Implementation of optimal page replacement algorithm	1

Note: Lab1 through Lab8 will be carried out in Linux Environment. Lab 9 through Lab11 will be carried out in Windows environment. Students should submit a lab sheet of each lab work on the date specified by the faculty.

Instructions for Students

Each lab assignment should be submitted with following format. The front page should contain the topic and lab number of each lab. Each lab assignment should contain the topic, objective, program code, output and output analysis.

Sample format: -

OPERATING SYSTEM

LAB ASSIGNMENT #6

Introduction to Interprocess communication.

……..

……..

Implementation of mutual exclusion methods with busy waiting.

6.1 Run the following program and analyze the output

OBJECTIVE:- To use strict alternation method to achieve mutual exclusion.

Program Code:

```
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>
#include<stdio.h>


void *thread1f(void *arg);
void *thread2f(void *arg);

int turn=1;

int main()
 {
  pthread_t thid1;
  pthread_t thid2;

  pthread_create(&thid1,NULL,&thread1f,NULL);
  pthread_create(&thid2,NULL,&thread2f,NULL);

  pthread_join(thid1,NULL);
  pthread_join(thid2,NULL);

  return 0;
 }


void *thread1f(void *arg)
 {
  int a=0;

  while(a++<20)
     {
      while(turn!=1);
       fputc('b',stderr);
       turn=0;
     }

 }

void *thread2f(void *arg)
   {
    int b=0;

    while(b++<20)
       {
        while(turn!=0);

        fputc('a',stderr);
```

```
            turn=1;
            }
      }
```

OUTPUT:-

```
[root@localhost lab6]# gcc -o strictalter strictalter.c -lpthread
strictalter.c:61:12: warning: no newline at end of file
[root@localhost lab6]# chmod a+x strictalter
[root@localhost lab6]# $strictalter
[root@localhost lab6]# strictalter
babababababababababababababababababababa
[root@localhost lab6]#
```

Output Analysis:-

This is an example of how strict alternation works in IPC. Here we have two processes. Each process have a task of printing the letter 'a' and 'b' twenty times. Here we have used the “turn” variable which keeps track of which process is currently in its CR. A process Pi which wants to enter CR is allowed only when turn=i, otherwise it goes in a tight loop until turn becomes i. That’s why this method is called strict alternation. We see the output as slowly printing 'b' and 'a' alternately. This method greatly wastes cpu time.