Operating Systems (CSL 3030)

Readme file

Lab Assignment	07
Roll No.	B20CS018
Name	Harshita Gupta

Ques1.

Part-1: no_deadlock

Command lines: gcc no_deadlock.c
./a.exe

Approach:

1. I used both Pthreads mutex locks and conditional variables making sure that a philosopher will only eat if both forks are available aka picking them up in the critical region.
2. There will be 5 philosophers created and each philosopher will assign a thread. A function pickup_forks() will be invoked whenever a philosopher wished to eat and philosophers will be alternating between thinking and eating.
3. When the philosopher is finished eating, the forks will be returned using the return_forks() method. Conditional variables are used as a locking mechanism for data integrity with the help of Pthreads mutex locks.
4. The mutex locks will be responsible to allow access to the shared data for the threads.

Part-2: with_deadlock

Command lines: gcc with_deadlock.c
./a.exe

Approach:

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])
  }
```

There are three states of the philosopher: **THINKING, HUNGRY, and EATING**. Here there are two semaphores: Mutex and a semaphore array for the philosophers. Mutex is used such that no two philosophers may access the pickup or putdown at the same time. The array is used to control the behavior of each philosopher. But, semaphores can result in deadlock due to programming errors.

Ques2.

Readers_Writers Problem

Command lines: gcc reader_writer.c
./a.exe

READER-WRITER DEFINITIONS

- start_thread: start a reader or writer thread
- reader, writer: subroutines called by pthread_create. They synchronize access to the file. See read_file and write_file.
- read_file, write_file: does the actual reading/writing, called by the reader/writer subroutines to read or write to the file.