Write Up

my_semaphore:

```
typedef struct{
   int value;
   pthread_mutex_t mutex;
   pthread_cond_t  condition;
} my_semaphore;
```

value: value of semaphore
mutex: provides access to value
condition: threads wait if they wait on the semaphore

- wait(): Decrease value, if value is less than 0, semaphore is used up. Hence, wait.
- signal(): Remove one process from the waiting list.

Dining Philosopher Problem

Philosopher tries to get both forks. If either of the forks is not available, the philosopher waits. Then, he tries to occupy sauce bowls. If they are not available, he waits. Then, he returns the forks and bowls.

Mutex is used so that no two philosophers pick up or put down forks at the same time.

Working of Code:

1. Create pthreads and semaphores
2. feedPhilosopher()
3. startEating(): First pick forks, then pick bowls.

4. pickFork()
    a. eat(): wait till either of the fork is unavailable
5. pickBowls(): wait till either of the bowl is unavailable
6. stopEating(): put down forks then put down bowls
    a. putFork()
    b. putBowl