```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <semaphore.h>

#define N 5
#define THINKING 2
#define HUNGRY 1
#define EATING 0
#define LEFT (num + 4) % N
#define RIGHT (num + 1) % N

void *philosopher(void *arg);
void take_fork(int num);
void put_fork(int num);
void test(int num);

int phil[N]={0,1,2,3,4};
sem_t *state[N];
pthread_mutex_t mutex;

int main() {

  int i,res;
  pthread_t *thread[N];
  char ch[7]="/state";

  for(i=0;i<N;i++) {
    ch[6]=i+'0';
    thread[i]=(pthread_t*)malloc(sizeof(pthread_t)*N);
    state[i]=sem_open(ch, O_CREAT, 0644, 1);
  }
  pthread_mutex_init(&mutex,NULL);

  //Function Calls
  for(i=0;i<N;i++) {
    res=pthread_create(thread[i],NULL,philosopher,&phil[i]);
    printf("\nPhilosopher %d is Thinking\n",i+1);
  }
  for(i=0;i<N;i++) {
    res=pthread_join(*thread[i],NULL);
  }

  //destroying the semaphores
  for(i=0;i<N;i++) {
    sem_close(state[i]);
  }
  pthread_mutex_destroy(&mutex);

  exit(EXIT_SUCCESS);
}

void test(int num) {

  if(state[num]==HUNGRY && state[LEFT]!=EATING &&
state[RIGHT]!=EATING) {
    state[num]=EATING;
  }
  sleep(2);
  sem_post(&state[num]);
}

void take_fork(int num) {

  pthread_mutex_lock(&mutex);

  state[num]=HUNGRY;
  printf("\nPhilosopher %d is Hungry\n",num+1);
  test(num);

  pthread_mutex_unlock(&mutex);

  sem_wait(&state[num]);
  sleep(1);
}

void put_fork(int num) {

  pthread_mutex_lock(&mutex);

  state[num]=THINKING;

  printf("\nPhilosopher %d putting fork %d and %d
down\n",num+1,LEFT+1,num+1);
  printf("Philosopher %d is Thinking\n",num+1);

  test(LEFT);
  test(RIGHT);

  pthread_mutex_unlock(&mutex);
}

void *philosopher(void *arg) {
  while(1) {

    int *num=arg;
    sleep(1);
    take_fork(*num);
    sleep(0);
    put_fork(*num);
  }
  pthread_exit(NULL);
}
```

Output:

```
Someshwars-MacBook-Pro:Assignment someshwargaikwad$ ./a.out

Philosopher 1 is Thinking

Philosopher 2 is Thinking

Philosopher 3 is Thinking

Philosopher 4 is Thinking

Philosopher 5 is Thinking

Philosopher 1 is Hungry

Philosopher 2 is Hungry

Philosopher 4 is Hungry

Philosopher 3 is Hungry

Philosopher 5 is Hungry

Philosopher 1 putting fork 5 and 1 down
Philosopher 1 is Thinking

Philosopher 2 putting fork 1 and 2 down
Philosopher 2 is Thinking

Philosopher 4 putting fork 3 and 4 down
Philosopher 4 is Thinking

Philosopher 3 putting fork 2 and 3 down
Philosopher 3 is Thinking

Philosopher 5 putting fork 4 and 5 down
Philosopher 5 is Thinking

Philosopher 1 is Hungry

Philosopher 2 is Hungry

Philosopher 4 is Hungry

Philosopher 3 is Hungry

Philosopher 5 is Hungry

Philosopher 1 putting fork 5 and 1 down
Philosopher 1 is Thinking
```