# Lab 3 – Dining Philosophers in pthreads

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#define N 5

#define LEFT (i + N - 1)%N

#define RIGHT (i + 1)%N

#define THINKING 0

#define HUNGRY 1

#define EATING 2

int state[N];

int threadID[N];

pthread\_t tid[N];

pthread\_cond\_t s[N];

pthread\_mutex\_t mutex\_lock;

void philosopher(void\* p);

void think();

void eat();

void take\_forks(int i);

void put\_forks(int i);

void test(int i);

void philosopher(void\* p) {

int\* start = (int\*) p;

int current = \*start;

for(int i=0; i<N; i++) {

think(current);

take\_forks(current);

eat(current);

put\_forks(current);

}

}

void think(int i) {

printf("Philosopher %d is thinking...\n", i+1);

sleep(1);

}

void eat(int i) {

printf("Philosopher %d is eating...\n", i+1);

sleep(1);

printf("Philosopher %d has finished eating\n", i+1);

}

//pick up forks

void take\_forks(int i) {

pthread\_mutex\_lock(&mutex\_lock);

state[i] = HUNGRY;

test(i);

while(state[i] != EATING) {

pthread\_cond\_wait(&s[i], &mutex\_lock);

}

pthread\_mutex\_unlock(&mutex\_lock);

}

//put down forks

void put\_forks(int i) {

pthread\_mutex\_lock(&mutex\_lock);

state[i] = THINKING;

test(LEFT);

test(RIGHT);

pthread\_mutex\_unlock(&mutex\_lock);

}

void test(int i) {

if(state[i]==HUNGRY && state[LEFT]!=EATING && state[RIGHT]!=EATING) {

state[i] = EATING;

pthread\_cond\_signal(&s[i]);

}

}

int main(int argc, char const \*argv[]) {

//initialize threads

for(int i=0; i<N; i++) {

state[i] = THINKING;

threadID[i] = i;

pthread\_cond\_init(&s[i], NULL);

}

pthread\_mutex\_init(&mutex\_lock, NULL);

//create philosophers

for (int i=0; i<N; i++) {

pthread\_create(&tid[i], NULL, (void\*) philosopher, (void\*) &threadID[i]);

}

for (int i=0; i<N; i++) {

pthread\_join(tid[i], NULL);

}

pthread\_exit(NULL);

return 0;

}

## Makefile

CFLAGS = -Wall -g

run: lab3.o

cc -o lab3 -pthread lab3.o

clean:

rm lab3 lab3.o

## Output

|  |  |
| --- | --- |
|  |  |