Operating System Labs Jan-May 2018

Dining Philosopher Problem

The dining-philosophers problem is considered a classic synchronization problem because it is an example of a large class of concurrency-control problems. It is a simple representation of the need to allocate several resources among several processes in a deadlock-free and starvation-free manner.

Consider five philosophers who spend their lives thinking and eating. The philosophers share a circular table surrounded by five chairs, each belonging to one philosopher. In the center of the table is a bowl of rice, and the table is laid with five single chopsticks. When a philosopher thinks, she does not interact with her colleagues. From time to time, a philosopher gets hungry and tries to pick up the two chopsticks that are closest to her (the chopsticks that are between her and her left and right neighbors). A philosopher may pick up only one chopstick at a time. Obviously, she cannot pick up a chopstick that is already in the hand of a neighbor. When a hungry philosopher has both her chopsticks at the same time, she eats without releasing the chopsticks. When she is finished eating, she puts down both chopsticks and starts thinking again.
Using threads and semaphores, implement a solution that coordinates thinking and eating activities of philosophers.

One possible sample output:

```
Philosopher 0 –> Pick left chopstick
Philosopher 0 –> Pick right chopstick
Philosopher 0 –> Begin eating
Philosopher 2 –> Pick left chopstick
Philosopher 3 –> Pick left chopstick
Philosopher 3 –> Pick right chopstick
Philosopher 3 –> Begin eating
Philosopher 0 –> Finish eating
Philosopher 0 –> Drop left chopstick
Philosopher 0 –> Drop right chopstick
Philosopher 1 –> Pick left chopstick
Philosopher 3 –> Finish eating
Philosopher 3 –> Drop left chopstick
Philosopher 3 –> Drop right chopstick
Philosopher 2 –> Pick right chopstick
Philosopher 2 –> Begin eating
Philosopher 4 –> Pick right chopstick
Philosopher 4 –> Pick left chopstick
Philosopher 2 –> Finish eating
.....
....
```