Práctica 3
Computación concurrente

Peto Gutiérrez Emmanuel

9 de noviembre de 2018

Filósofo

```
Begin
    while(true){
        pensar()
        wait(palillo_izquierdo)
        wait(palillo_derecho)
        comer()
        signal(palillo_derecho)
        signal(palillo_izquierdo)
    }
End
```

El problema no se resuelve porque todos los filósofos toman primero el palillo izquierdo. Si todos terminan de pensar al mismo tiempo, todos tomarán el palillo que tienen a la izquierda al mismo tiempo, después al intentar tomar el derecho se dormirán al llamar el wait(palillo_derecho), esto es porque todos los palillos que tienen a su derecha ya están ocupados. Esto ocasionará un deadlock.

Para resolver el problema se utilizó un semáforo general que le permite la entrada a solo 4 hilos a la sección crítica. Es decir, solo 4 filósofos podrán intentar comer a la vez y el quinto filósofo(el que llegó al final) deberá esperar a que uno de ellos termine de comer para intentar tomar los palillos y eso lo hace libre de deadlock. Antes de que intenten tomar los palillos siempre habrá 1 filósofo dormido, entonces, si todos toman su palillo izquierdo podemos asegurar que el filósofo a la izquierda del dormido va a comer.