Asignación de Prácticas Número 7

Jesús Gómez

Noviembre 2022

Resumen

En este documento se reflejarán los análisis realizados en el código desarrollado, además de su debida interpretación.

1. Análisis I. *Buffer* de una posición, donde trabajan únicamente un productor y un consumidor.

Con un *buffer* finito de una sola posición, en el que actúan solo un productor y un consumidor podemos observar una alternancia estricta, donde el productor produce una unidad y tiene que esperar al que consumidor consuma, y viceversa.
Esto se asemeja a un comportamiento secuencial, empeorado por la creación de hilos y la complejidad de código, lo cual no resulta eficiente.

```
Productor: Thread-0 produciendo el dato: 52 ...
Consumidor: Thread-1 consumiendo el dato: 52 ...
Productor: Thread-0 produciendo el dato: 931 ...
Consumidor: Thread-1 consumiendo el dato: 931 ...
Productor: Thread-0 produciendo el dato: 560 ...
Consumidor: Thread-1 consumiendo el dato: 560 ...
Productor: Thread-0 produciendo el dato: 807 ...
Consumidor: Thread-1 consumiendo el dato: 807 ...
Productor: Thread-0 produciendo el dato: 765 ...
Consumidor: Thread-1 consumiendo el dato: 765 ...
Productor: Thread-0 produciendo el dato: 587 ...
Consumidor: Thread-1 consumiendo el dato: 587 ...
```

Figura 1: Resultado obtenido, *buffer* 1 posición.

2. Análisis II. Un productor y varios consumidores

En este caso, al existir un único productor que abastece a varios consumidores, es muy probable que la condición de concurso induzca una alternancia entre productor-consumidor, ya que el *buffer* casi siempre tenderá a estar vacío.

```
Productor: Thread-0 produciendo el dato: 239 ...
Consumidor: Thread-5 consumiendo el dato: 239 ...
Productor: Thread-0 produciendo el dato: 874 ...
Consumidor: Thread-4 consumiendo el dato: 874 ...
Productor: Thread-0 produciendo el dato: 242 ...
Consumidor: Thread-5 consumiendo el dato: 242 ...
Productor: Thread-0 produciendo el dato: 465 ...
Consumidor: Thread-3 consumiendo el dato: 465 ...
Productor: Thread-0 produciendo el dato: 450 ...
Consumidor: Thread-3 consumiendo el dato: 450 ...
Productor: Thread-0 produciendo el dato: 312 ...
Consumidor: Thread-4 consumiendo el dato: 312 ...
```

Figura 2: Un productor, varios consumidores.

3. Análisis III. Varios productores y un consumidor

En este caso, y al contrario de lo que sucede en el anterior análisis, al existir un único consumidor que es abastecido por varios consumidores, es muy probable que la condición de concurso induzca una alternancia entre productor-consumidor, ya que el *buffer* casi siempre tenderá a estar lleno.

```
Productor: Thread-0 produciendo el dato: 887 ...
Productor: Thread-4 produciendo el dato: 568 ...
Productor: Thread-3 produciendo el dato: 427 ...
Productor: Thread-2 produciendo el dato: 647 ...
Productor: Thread-1 produciendo el dato: 674 ...
Consumidor: Thread-5 consumiendo el dato: 887 ...
Productor: Thread-2 produciendo el dato: 238 ...
Consumidor: Thread-5 consumiendo el dato: 568 ...
Productor: Thread-4 produciendo el dato: 616 ...
Consumidor: Thread-5 consumiendo el dato: 427 ...
Productor: Thread-1 produciendo el dato: 761 ...
Consumidor: Thread-5 consumiendo el dato: 647 ...
```

Figura 3: Un consumidor, varios productores.