PRUEBAS DE LA PRÁCTICA 4 DE FSO

Alejandro Vialard Santana

Grupo 1-41

Búfer Finito Sin Herramientas De Sincronización:

Primero, haremos pruebas con el ejercicio del búfer no implementado.

```
[efso@localhost bufer_sin_implementar]$ ./test_hilos 5 3 2
»»» Comienza la prueba del búfer...
>>> inserción: item=0, pos=0
>>> inserción: item=2, pos=1
<<< extracción: item=0, pos=0
Consumidor extrae: 0
<<< extracción: item=2, pos=1
Consumidor extrae: 2
>>> inserción: item=1, pos=2
»»» Fin de la prueba del búfer
[efso@localhost bufer_sin_implementar]$
```

En esta prueba, hemos probado a poner tres productores y dos consumidores, vemos que insertan dos productores extraen dos consumidores e inserta el último. Aquí no parece que haya nada raro, pero vemos que el último elemento no lo inserta en la posición 0, si no en la 2, haciendo ver como que sigue con objetos el búfer. Además, cada productor solo produce uno y cada consumidor solo consume uno. Además, pese a poner que entren de manera aleatoria, como solo pueden insertar y extraer un elemento no parezca que cambie nunca la llegada de hilos.

```
[efso@localhost bufer_sin_implementar]$ ./test_hilos 10 2 8
»»» Comienza la prueba del búfer...
>>> inserción: item=0, pos=0
<<< extracción: item=0, pos=0
Consumidor extrae: 0
>>> inserción: item=1, pos=1
<<< extracción: item=0, pos=1
Consumidor extrae: 0
```

Aquí probamos con una capacidad de 10, y dos productores y 8 consumidores. Vemos como el productor introduce, luego sacan el elemento, se introduce otro y se saca también, pero el consumidor extrae el item 0 cuando ya no estaba, lo cual es raro. Además, se queda en inanición porque al no haber productores los hilos consumidores se quedan esperando para extraer provocando que se quede en bucle.

```
[efso@localhost bufer_sin_implementar]$ ./test_hilos 4 8 2
»»» Comienza la prueba del búfer...
>>> inserción: item=0, pos=0
>>> inserción: item=4, pos=1
<<< extracción: item=0, pos=0
Consumidor extrae: 0
<<< extracción: item=4, pos=1
Consumidor extrae: 4
>>> inserción: item=2, pos=2
>>> inserción: item=6, pos=3
>>> inserción: item=5, pos=0
>>> inserción: item=7, pos=1
>>> inserción: item=3, pos=2
>>> inserción: item=1, pos=3
»»» Fin de la prueba del búfer
```

Ahora, hemos probado con 8 productores y 2 consumidores. Observamos lo siguiente y es que se insertan 4 objetos, pero se extraen solo 2. Sin embargo, habiendo solo dos huecos en el búfer, se insertan los otros 4 superponiéndose en las posiciones 2 y 3.

Búfer Finito Con Herramientas De Sincronización:

```
[efso@localhost bufer_completo]$ ./test_hilos 5 3 2
»»» Comienza la prueba del búfer...
>>> inserción: item=2, pos=0
>>> inserción: item=2, pos=1
>>> inserción: item=0, pos=2
>>> inserción: item=0, pos=3
<<< extracción: item=2, pos=0
<<< extracción: item=2, pos=1
<<< extracción: item=0, pos=2
Consumidor extrae: 0
<<< extracción: item=0, pos=3
Consumidor extrae: 0
>>> inserción: item=1, pos=4
>>> inserción: item=1, pos=0
»»» Fin de la prueba del búfer
```

Ahora, probamos con el búfer correcto, y para la primera prueba vemos como el mismo productor puede insertar más de un elemento (en este caso he puesto que puedan insertar como máximo 2), al igual que un consumidor puede extraer varios (he puesto 3). Observamos que se insertan hasta llenar el búfer, el consumidor extrae 3, pero el siguiente solo extrae 1, y luego el productor restante agrega otros dos.

```
[efso@localhost bufer_completo]$ ./test_hilos 5 3 2
»»» Comienza la prueba del búfer...
>>> inserción: item=0, pos=0
>>> inserción: item=0, pos=1
<<< extracción: item=0, pos=0
<<< extracción: item=0, pos=1
Consumidor extrae: 0
>>> inserción: item=2, pos=2
>>> inserción: item=2, pos=3
<<< extracción: item=2, pos=2
<<< extracción: item=2, pos=3
Consumidor extrae: 2
>>> inserción: item=1, pos=4
>>> inserción: item=1, pos=0
»»» Fin de la prueba del búfer
```

Esta sigue siendo la misma prueba pero vemos que el patrón ha cambiado ya que se inserta dos y se extrae dos, ya que entran de manera aleatoria los hilos, pero no hace nada extraño.

```
[efso@localhost bufer_completo]$ ./test_hilos 10 2 8
»»» Comienza la prueba del búfer...
>>> inserción: item=0, pos=0
>>> inserción: item=0, pos=1
<<< extracción: item=0, pos=0
<<< extracción: item=0, pos=1
Consumidor extrae: 0
>>> inserción: item=1, pos=2
>>> inserción: item=1, pos=3
<<< extracción: item=1, pos=2
<<< extracción: item=1, pos=3
Consumidor extrae: 1
El búfer está vacío y no hay productores, no se puede extraer nada
El búfer está vacío y no hay productores, no se puede extraer nada
El búfer está vacío y no hay productores, no se puede extraer nada
El búfer está vacío y no hay productores, no se puede extraer nada
El búfer está vacío y no hay productores, no se puede extraer nada
El búfer está vacío y no hay productores, no se puede extraer nada
```

Para una capacidad de 10, y dos productores y 8 consumidores antes se quedaba en espera siempre, pero ahora observamos que cuando no hay productores se lanza una excepción de que no se puede extraer nada y se salen todos los hilos.

```
[efso@localhost bufer_completo]$ ./test_hilos 4 8 2
»»» Comienza la prueba del búfer...
>>> inserción: item=7, pos=0
>>> inserción: item=7, pos=1
>>> inserción: item=7, pos=2
>>> inserción: item=7, pos=3
<<< extracción: item=7, pos=0
<<< extracción: item=7, pos=1
<<< extracción: item=7, pos=2
<<< extracción: item=7, pos=3
Consumidor extrae: 7
>>> inserción: item=0, pos=0
>>> inserción: item=0, pos=1
>>> inserción: item=0, pos=2
>>> inserción: item=0, pos=3
<<< extracción: item=0, pos=0
<<< extracción: item=0, pos=1
<<< extracción: item=0, pos=2
<<< extracción: item=0, pos=3
Consumidor extrae: 0
>>> inserción: item=6, pos=0
>>> inserción: item=6, pos=1
>>> inserción: item=6, pos=2
>>> inserción: item=6, pos=3
Búfer lleno y no hay consumidores, no se puede insertar nada
Búfer lleno y no hay consumidores, no se puede insertar nada
Búfer lleno y no hay consumidores, no se puede insertar nada
Búfer lleno y no hay consumidores, no se puede insertar nada
Búfer lleno y no hay consumidores, no se puede insertar nada
»»» Fin de la prueba del búfer
```

Y en este ocurría que se insertaban elementos, aunque el búfer estaba lleno, pero ahora vemos que cuando ocurre eso, lanza una excepción y termina el programa.

```
[efso@localhost bufer_completo]$ ./test_hilos 4 8 2
»»» Comienza la prueba del búfer...
>>> inserción: item=0, pos=0
>>> inserción: item=0, pos=1
>>> inserción: item=3, pos=2
>>> inserción: item=3, pos=3
<<< extracción: item=0, pos=0
<<< extracción: item=0, pos=1
<<< extracción: item=3, pos=2
Consumidor extrae: 3
>>> inserción: item=4, pos=0
>>> inserción: item=4, pos=1
>>> inserción: item=6, pos=2
<<< extracción: item=3, pos=3
<<< extracción: item=4, pos=0
<<< extracción: item=4, pos=1
Consumidor extrae: 4
>>> inserción: item=2, pos=3
>>> inserción: item=2, pos=0
>>> inserción: item=7, pos=1
Búfer lleno y no hay consumidores, no se puede insertar nada
Búfer lleno y no hay consumidores, no se puede insertar nada
»»» Fin de la prueba del búfer
```

Es el mismo que el anterior, sin embargo, al ser aleatorio esta vez inserta 4 elementos de primera en vez de dos, probando nuevamente que los hilos entran de manera aleatoria.

Misioneros y Caníbales Sin Herramientas De Sincronización:

```
[efso@localhost misioneros_y_canibales]$ ./Misios_y_Canibales_Incorrecto 3 3
Comienza la prueba del bote...
Sube un misionero: misioneros = 1, canibales = 0
Sube un canibal: misioneros = 1, canibales = 1
Sube un canibal: misioneros = 1, canibales = 2
Zarpa el bote: misioneros = 1, canibales = 2
Baja un canibal: misioneros = 1, canibales = 1
Baja un canibal: misioneros = 1, canibales = 0
Baja un misionero: misioneros = 0, canibales = 0
Bote vacío: misioneros = 0, canibales = 0
Dos misioneros y un canibal no pueden viajar juntos.
Dos misioneros y un canibal no pueden viajar juntos.
Dos misioneros y un canibal no pueden viajar juntos.
Termina la prueba del bote.
```

En este caso ocurrió un buen funcionamiento aun no teniendo las herramientas necesarias, ya que se subieron dos caníbales y un misionero, y luego solo quedaban 2 misioneros y un caníbal y salta el error, pero esto es excepcional porque es por como lo implementé, si le quitara la condición de que no debe de haber dos misioneros y un caníbal ocurriría que se quedaría esperando el hilo constantemente. Puede darse el caso de que entren 3 misioneros y 3 caníbales y que se complete bien.

```
[efso@localhost misioneros_y_canibales]$ ./Misios_y_Canibales_Incorrecto 5 1
Comienza la prueba del bote...
Sube un canibal: misioneros = 0, canibales = 1
Sube un misionero: misioneros = 1, canibales = 1
```

Con esta combinación si entra un caníbal, se queda parado sin terminar el programa.

```
[efso@localhost misioneros_y_canibales]$ ./Misios_y_Canibales_Incorrecto 4 5
Comienza la prueba del bote...
Sube un canibal: misioneros = 0, canibales = 1
Sube un misionero: misioneros = 1, canibales = 1
Sube un canibal: misioneros = 1, canibales = 2
Zarpa el bote: misioneros = 1, canibales = 2
Baja un canibal: misioneros = 1, canibales = 1
Baja un canibal: misioneros = 1, canibales = 0
Baja un misionero: misioneros = 0, canibales = 0
Bote vacío: misioneros = 0, canibales = 0
Sube un canibal: misioneros = 0, canibales = 1
Sube un canibal: misioneros = 0, canibales = 2
Sube un canibal: misioneros = 0, canibales = 3
Zarpa el bote: misioneros = 0, canibales = 3
Baja un canibal: misioneros = 0, canibales = 2
Baja un misionero: misioneros = -1, canibales = 2
Baja un canibal: misioneros = -1, canibales = 1
Bote vacío: misioneros = -1, canibales = 1
Dos misioneros y un canibal no pueden viajar juntos.
Dos misioneros y un canibal no pueden viajar juntos.
```

Y aquí observamos una cosa extraña, y es que, habiéndose subido tres caníbales, luego se extrae un misionero y se queda con valor -1, hecho que no debería de ocurrir.

Misioneros y Caníbales Con Herramientas De Sincronización:

```
[efso@localhost misioneros_y_canibales]$ ./Misios_y_Canibales 3 1
Comienza la prueba del bote...
Sube un misionero: misioneros = 1, canibales = 0
Sube un misionero: misioneros = 2, canibales = 0
Sube un misionero: misioneros = 3, canibales = 0
Zarpa el bote: misioneros = 3, canibales = 0
Baja un misionero: misioneros = 2, canibales = 0
Baja un misionero: misioneros = 1, canibales = 0
Baja un misionero: misioneros = 0, canibales = 0
Bote vacío: misioneros = 0, canibales = 0
El bote no se puede llenar y no puede zarpar.
Termina la prueba del bote.
[efso@localhost misioneros_y_canibales]$ S
```

Si por ejemplo ponemos valores que su total no sea múltiplo de tres, llegará posiblemente a que el bote no se puede llenar.

```
[efso@localhost misioneros_y_canibales]$ ./Misios_y_Canibales 5 1
Comienza la prueba del bote...
Sube un misionero: misioneros = 1, canibales = 0
Sube un misionero: misioneros = 2, canibales = 0
Sube un misionero: misioneros = 3, canibales = 0
Zarpa el bote: misioneros = 3, canibales = 0
Baja un misionero: misioneros = 2, canibales = 0
Baja un misionero: misioneros = 1, canibales = 0
Baja un misionero: misioneros = 0, canibales = 0
Bote vacío: misioneros = 0, canibales = 0
Dos misioneros y un canibal no pueden viajar juntos.
Dos misioneros y un canibal no pueden viajar juntos.
Dos misioneros y un canibal no pueden viajar juntos.
Termina la prueba del bote.
[efso@localhost misioneros_y_canibales]$
```

En caso de que queden dos misioneros esperando junto a un caníbal, termina el programa lanzando un mensaje por pantalla que de lo ocurrido.

*PROBLEMA:

```
[efso@localhost misioneros_y_canibales]$ ./Misios_y_Canibales 5 1
Comienza la prueba del bote...
Sube un misionero: misioneros = 1, canibales = 0
Sube un canibal: misioneros = 1, canibales = 1
```

Con la misma combinación, si entra de primeras un caníbal y un misionero, se queda en inanición debido a que yo sólo conseguí quitarla cuando se quedan 3 esperando al final y no al principio, se podría mirar métodos para hacer que esto no ocurra como por ejemplo un temporizador que cuando haya esperado mucho tiempo, termine.