Universidad
Nacional Autónoma de México

Facultad de Ciencias

Practica 4
Productor y consumidor

Integrantes:
Yonathan Berith Jaramillo Ramírez. 419004640
Diego Arturo Velázquez Trejo. 317227257
Marco Antonio Garcia Arce, 421014615

Profesor: Glide Valeria Rodriguez Jimenez
Ayudantes: Hermilio Cortez Gonzalez
Luis Angel Leyva Castillo
Rogelio ALcantar Arenas

22 Octubre, 2023

Computo concurrente

Introducción

Zeratun tuvo un gran éxito con su estacionamiento con puesto de barbacoa, por lo que su siguiente gran negocio es poner una tortillería, de esta manera genera un súper incremento en su billetera. Gracias a su tortellería se volverá el HEROE DE MÉXICO.

PRACTICA

En el primer ejercicio deberan ayudarle a implementar su tortillería, para esto se implementara una versión de la solución del algoritmo del panadero (Convirtiéndose ahora en el algoritmo del tortillero).

Para esto se usaran marcas de tiempo, lo primero será usar la clase StampedSnap y la modificaran para poder generar una marca de tiempo (Ten en cuenta como se genera).

Una vez hecho esto, crea un Snapshot wait free, como el visto en clase con la profesora, para esto implementa la interfaz Snapshot, añade todos los métodos que creas que son necesarios.

Finalmente usando estas dos clases se podra crear el algoritmo, para esto usaremos un arreglo (esto lo haremos como el filtro, es decir, es estático), en este guardaremos las marcas de tiempo. Una vez que esten registradas las marcas, solo quedara ordenarlas (utiliza algun ordenamiento), de esta manera ya tendremos ordenado el arreglo y podremos despachar a los clientes. Para controlar el ingreso de clientes, puedes usar algun lock creado previamente.

Para el segundo ejercicio se realizara una version del productor consumidor, siendo que tenemos n productores y 1 consumidor, para esto se modificara un poco el como se produce y el como se consume.

Para el segundo ejercicio se realizara una version del productor consumidor, siendo que tenemos n productores y 1 consumidor, para esto se modificara un poco el como se produce y el como se consume.

- Se pondra un limite de cuanto se puede producir, de tal manera no se desperdiciarán tortillas.
- El límite anterior lo podemos manejar por casilla (no en total).

- El consumidor lo que hará será recolectar todas las tortillas, es decir, poseerá un contador y este tendrá un total de tortillas.
- Pueden suponer que el consumidor es súper fuerte y puede cargar n*limite de tortillas
- Si no hay tortillas producidas, el consumidor no podra recolectar, es decir se quedará en 0.
- Finalmente, cuando se acabe la ejecucion, para no tener algun deadlock, has que el consumidor se vaya a casa.

TEORÍA

- **Explica porqué tu solución sí funciona, para esto puedes demostrar las propiedades de deadlock-free, libre de hambruna, lock-free, wait-free.**
  La solución utiliza un algoritmo de "Wait-Free Snapshot" para capturar el estado del sistema en un momento dado. Este algoritmo es "wait-free", lo que significa que cada operación se completa en un número finito de pasos, independientemente de la ejecución de otros hilos. Además, al utilizar marcas de tiempo, se garantiza que no hay "deadlocks" ni condiciones de "hambruna", ya que cada cliente (hilo) recibe un servicio basado en su marca de tiempo.
- **¿Qué modificación tendrías que hacer para que tuvieramos m consumidores?**
  Para tener m consumidores, podríamos inicializar el objeto 'WFSnapshot' con una capacidad de m en lugar de 10. También tendríamos que asegurarnos de que el array de marcas de tiempo tenga un tamaño de m.
- **Supón que posees más atributos, de tal manera que tienes un atributo que almacena los kg de masa disponibles, así también de otro que almacena la cantidad de masa requerida para hacer 1 tortilla, ¿Qué modificación tendrías que hacer para que esto funcionara?**
  Podríamos extender la clase 'StampedSnap' para incluir estos nuevos atributos. Además, tendríamos que modificar el método 'update' para actualizar estos nuevos atributos y el método 'scan' para capturarlos en el snapshot.
- **Para el problema de la tortilleria, como podríamos hacer para tener una fila dinámica, es decir que las personas llegaran sin tener que agruparlas o en bloques estáticos. Explica tu solución.**

Para tener una fila dinámica, podríamos utilizar una estructura de datos como una cola de prioridad basada en las marcas de tiempo para insertar y eliminar clientes de manera dinámica. Esto permitiría que los clientes lleguen y sean atendidos en un orden que depende de su marca de tiempo, sin necesidad de agruparlos.

- **Da comentarios de lo que aprendiste en esta práctica.**
  Esta práctica fue muy útil para entender los conceptos de concurrencia como los algoritmos "wait-free" y "lock-free". También proporcionó una buena experiencia práctica en la implementación de algoritmos de concurrencia en un escenario del mundo real, como una tortillería.

ENTREGABLE

Zeratún solicita que las respuestas se hagan a computadora, en el editor de su elección. Deben de poner las referencias bibliográficas en donde consultaron la información, esta debe de ir en formato APA y en formato PDF. Se les dará 0.5 extra si la realizan en LaTeX.

Debe de llevar el siguiente formato: [NOMBRE DEL EQUIPO.[zip] EJEMPLO: The-FunadosTeam.zip