SCD

Repositorio para la asignatura de Sistemas Concurrentes y Distribuidos durante el curso 2023/2024 por Ricardo Ruiz Fernández de Alba.

Contenido

- Seminario 1 - Mutex y Semáforos
- Práctica 1 - Mutex y Semáforos
  - **Problema del Productor-Consumidor (Versión LIFO)**
    * Código
    * Se usan tres semáforos:
      · `Semaphore libres(tam_vec)` indica que hay `n` posiciones libres en el buffer. Se bloquea cuando el buffer está lleno.
      · `Semaphore ocupadas(0)` indica que el buffer está vacío. El consumidor se bloquea cuando el buffer hasta que el productor añada un elemento.
    * Como el buffer es una pila (LIFO), el productor añade elementos al final y el consumidor los saca del final a través del índice `primera_libre`, que se decrementa cuando se saca un elemento y se incrementa cuando se añade.
    * Por ello se usa un semáforo adicional `Semaphore puede_acceder(1)` para garantizar la exclusión mutua en la escritura y lectura del buffer.
    * Código
  - **Problema del Productor-Consumidor (Versión FIFO)**
    * Código
    * Se usan 2 semáforos:
      · `Semaphore libres(tam_vec)` indica que hay `tam_vec` posiciones libres en el buffer. Se bloquea cuando el buffer está lleno.
      · `Semaphore ocupadas(0)` indica que el buffer está vacío. El consumidor se bloquea cuando el buffer hasta que el productor añada un elemento.
    * Como el buffer es una cola FIFO se usan dos índices `primera_libre` y `primera_ocupada` para indicar la primera posición libre y la primera ocupada respectivamente. Ambos índices se incrementan módulo `tam_vec` para que el buffer sea circular. La extracción de elementos se puede hacer de forma concurrente

usando cada variable, por lo que no es necesario usar un semáforo adicional para garantizar la exclusión mutua.

- **Problema de Múltiples Productores y Múltiples Consumidores (Versión LIFO)**
- Código
  * Disponemos de 40 ítems, 8 productores y 5 consumidores (basta con que sean divisores)
  * Cada productor produce p=5 ítems y cada consumidor consume c=8 ítems.
  * Para producir dato se usa el índice del productor i y se llama a producir_dato(i, j) con j=0,...p. Producir dato genera el valor i*p+j.
  * También se almacena en un array producidos de tamaño 8 cuantos datos ha producido cada productor. No hace falta un semáforo para garantizar la exclusión mutua porque cada productor escribe en una posición distinta.
  * Se usan 3 semáforos:
    · `Semaphore libres(tam_vec)` indica que hay `tam_vec` posiciones libres en el buffer. Se bloquea cuando el buffer está lleno.
    · `Semaphore ocupadas(0)` indica que el buffer está vacío. El consumidor se bloquea cuando el buffer está vacío hasta que el productor añada un elemento.
    · `Semaphore puede_acceder(1)` para garantizar la exclusión mutua en la escritura y lectura del buffer. Debe ser el mismo por ser un buffer LIFO.
- **Problema de Múltiples Productores y Múltiples Consumidores (Versión FIFO)**
  * Código
  * Se aplica lo anterior.
  * Se usan 4 semáforos:
  * `Semaphore libres(tam_vec)`: indica que hay `tam_vec` posiciones libres en el buffer. Se bloquea cuando el buffer está lleno.
  * `Semaphore ocupadas(0)` indica que el buffer está vacío. El consumidor se bloquea cuando el buffer está vacío hasta que el productor añada un elemento.
  * `Semaphore puede_leer(1)`: para garantizar la exclusión mutua entre lecturas (consumidores). Puede ejecutarse paralelamente a las escrituras
  * `Sempahore puede_escribir(1)`: para garantizar la exclusión mutua entre escrituras (productores). Puede ejecutarse paralelamante a las lecturas.
- **Problema de los fumadores**
  * Código

    * Se consta de `num_fumadores` fumadores, 1 estanquero
    * Cada fumador tiene un ingrediente distinto, que se representa con un número 0,1,..num_fumadores-1
    * El estanquero produce un ingrediente aleatorio y lo pone en el mostrador.
    * El fumador que tiene ese ingrediente lo recoge, lia su cigarrillo y fuma.
    * Si un fumador no tiene el ingrediente que se ha producido, se bloquea hasta que el estanquero produzca su ingrediente.
    * Si el estanquero ha producido un ingrediente pero el mostrador no está vacío, se bloquea hasta que el fumador recoja el ingrediente.
    * Se necesitan 1 semáforo y un array de 3 semáforos:
        · `Semaphore mostr_vacio(1)`: indica que el mostrador está vacío. Se bloquea cuando el mostrador está lleno.
        · `Semaphore ingr_disp[num_fumadores] = {0, 0, 0}`: Cada Semaphore ingr_disp[i] indica que el ingrediente i está disponible en el mostrador. Se bloquea cuando el ingrediente i no está disponible.
    * La hebra fumadora i se bloquea en ingr_disp[i] y se desbloquea cuando el estanquero produce el ingrediente i. Indica que el mostrador está vacío. Luego fuma (espera aleatoria) repite eso indefinidamente
    * La hebra estanquero produce el ingrediente, espera a que el mostrador esté vacío y pone el ingrediente en el mostrador. Luego desbloquea a la hebra fumadora que espera ese ingrediente.

– **Simulacro de Examen**

– Código

    * Se amplia el problema de los fumadores con una hebra sanitaria.
    * La hebra sanitaria se ejecuta indefinidamente esperando a que un fumador haya fumado 5 veces y haya sido desbloqueado para fumar por 6ta vez.
    * Entonces la hebra imprime: “FUMAR MATA: ya lo sabes fumador `num_fumador`” y se bloquea.
    * Entonces, el fumador que ha fumado 5 veces se desbloquea e imprime: “Soy el fumador `num_fumador` y me han llamado vicioso”. Acto seguido retira el ingrediente, avisa que el mostrador está vacío y fuma.

    * Se requiere un array num_fumados[num_fumadores] que indica cuantas veces ha fumado cada fumador. Se inicializa a 0. Cuando un fumador fuma, se incrementa en 1 su contador. Esto se hace en exclusión mutua sin semáforos porque cada fumador aumenta su contador.
    * Se requiere una variable num_fumador_sanitaria que indica el fumador que ha desbloqueado a la hebra sanitaria.
    * Se requieren 2 semáforos adicionales al semáforo y array de semáforos del problema de los fumadores.
        · `Semaphore libre_sanitaria(0)`: por defecto la hebra sanitaria se bloquea hasta que un fumador haya fumado 5 veces y haya sido desbloqueado para fumar por 6ta vez.
        · `Semaphore ocupada_sanitaria(0)`: una vez un fumado rhaya fumado 5 veces y haya sido desbloqueado para fumar por 6ta vez, se bloquea hasta que la hebra sanitaria imprima el mensaje "Fumar mata ...". Acto seguido desbloquea al fumador para imprimir su mensaje. Se hace uso de la variable num_fumador_sanitaria para saber el numero de fumador.

- **Ejercicios Adicionales**
- **Productor consumidor FIFO con impresora con semáforos**
- Código
- Se creará una nueva variable compartida que contendrá el número de celdas ocupadas en el buffer en cada momento (es decir, elementos producidos y añadidos al vector pero todavía no consumidos). Esta variable se debe actualizar por el productor y por el consumidor, teniendo en cuenta que dicha actualización forma parte del proceso de inserción o extracción de valores del buffer, de forma que el valor de la variable siempre se corresponda con el estado de dicho buffer (es decir, ninguna hebra en ningún momento leerá en esa variable un valor distinto del real según las inserciones y extracciones completadas hasta ese momento).
- (2) Se debe crear una nueva hebra llamada impresora, que ejecuta un bucle finito. En cada iteración debe bloquearse hasta que sea desbloqueada por otra hebra, y después, tras ser desbloqueada, debe imprimir por pantalla el número de celdas ocupadas que hay en el vector en ese momento

(3) Cuando el productor produzca un número múltiplo de 5, inmediatamente después de insertar dicho número al vector, debe desbloquear a la hebra impresora y luego debe bloquearse hasta ser desbloqueado por la hebra impresora. Hay que tener en cuenta que la

hebra impresora debe imprimir el contador de casillas ocupadas resultante de esta última inserción del productor (no un valor posterior distinto, resultado de otras operaciones posteriores distintas a esta última).

(4) La hebra impresora, una vez que haya escrito el mensaje por pantalla, desbloqueará al productor y volverá al principio de su ciclo. El número de iteraciones de la hebra impresora es igual al número de múltiplos de 5 que hay entre 0 y N-1 (ambos incluidos), es decir, será N/5 (división entera), donde N es el número total de valores producidos.

- **Gasolinera con semáforos**
  - Tenemos 10 hebras que ejecutan un bucle infinito y que representan a coches que necesitan entrar a repostar a una gasolinera en cada iteración del bucle. Hay A=4 hebras tipo diesel y B=6 hebras tipo gasolina. 2 os más coches no pueden repostar a la vez en un surtidor de su tipo.
  - Para entrar a la gasolinera, un coche debe esperar a que quede libre algún surtidor del tipo que necesita. La gasolienera tiene C=2 surtidores de diesel y D=3 surtidores de gasolina. Debe cumplirse C < A y D < B.
  - La parte del código de los coches que no está en la gasolinera se simula con retardos aleatorios. El respotaje con la función repostar, que se hace retraso de duración n. En repostar repostar se imprime “Coche número `n` de”disel/gasolina” comienza a respotar” y el mensaje de terminar.
  - Es necesario contabilizar en una variable compartida el número total de surtidores en uso en cada momento. (surtidores_ocupados inicialmente a 0)
  - Cada coche de gasolina y gasoil la incrementará en exclusión mutua.
- Se requieren 3 semáforos:
  - `Semaphore surtidor_diesel_libre(C)`**:** cuenta el número de surtidores de diesel libres. Se bloquea cuando no hay surtidores de diesel libres.
  - `Semaphore surtidor_gasolina_libres(D)`**:** cuenta el número de surtidores de gasolina libres. Se bloquea cuando no hay surtidores de gasolina libres.
  - `Semaphore uso_surtidor(1)`: para garantizar la exclusión mutua en la actualización de la variable compartida surtidores_ocupados. Como la variable indice el número total hay que utilizarlo tanto para los surtidores de gasolina como para los de diesel.
- Se hace uso de 1 semáforo adicional `Semaphore msg(1)` para garantizar la exclusión mutua en la impresión de mensajes.
- **Fumadores y contrabandista (con semáforos)**

- Seminario 2 - Monitores

- Práctica 2 - Problemas con Monitores
  - **Monitor SU: Problema del Productor consumidor (Versión LIFO)**
  - **Monitor SU: Problema del Productor consumidor (Versión FIFO)**
  - **Monitor SU: Problema de Múltiples Productores y Consumidores (Versión LIFO)**
  - **Monitor SU: Problema de Múltiples Productores y Consumidores (Versión FIFO)**
  - **Monitor SU: Problema de los fumadores**
  - **Monitor SU: Problema de los lectores y escritores**
  - **Productores consumidores múltiples FIFO con impresores con Monitor SU**
  - **Gasolinera con Monitor SU**
  - **Fumadores y contrabandista con 2 Monitores SU**

Licencia