Documentación Práctica 2

Por: Arturo Cortés Sánchez

Seminario:

Actividades relativas al monitor Barrera1 (SC)

- La hebra que entra la última al método cita (la hebra señaladora) es siempre la primera en salir de dicho método.
  - Porque es la que ejecuta `notify_one()` y por tanto es la primera en salir
- El orden en el que las hebras señaladas logran entrar de nuevo al monitor no siempre coincide con el orden de salida de wait (se observa porque los números de orden de entrada no aparecen ordenados a la salida).
  - Una vez se desbloquean las hebras, todas intentan acceder al mutex, por tanto el orden es imprevisible
- El constructor de la clase no necesita ejecutarse en exclusión mutua.
  - El constructor solo se ejecuta una vez, luego no tiene sentido que se ejecute en exclusión mutua
- Prueba a usar `notify_all()` en lugar de `notify_one()`. Describe razonadamente en tu portafolio si se observa o no algún cambio importante en la traza del programa.
  - No hay cambios apreciables en la salida del programa

Actividad sobre prodcons1_sc.cpp:

- Modifica la implementación para usar la opción FIFO. Verifica que funciona correctamente. Describe razonadamente en tu portafolio los cambios que esto supone.
  - Para hacer la versión FIFO he añadido dos variables a la clase monitor: `primera_ocupada` y `n` las cuales se inicializan a 0 en el constructor. En el método `leer()` la condición para que el consumidor espere es cambiada a que `n` sea igual a 0. El indice del buffer pasa a ser `primera_ocupada` modulo `num_celdas_total`, ya que primera ocupada puede ser superior al tamaño del buffer. Además, en lugar de decrementar `primera_libre` paso a decrementar `n` e incrementar `primera_ocupada`. En `escribir()` la condición para que el productor espere es cambiada a que `n` sea igual a `num_celdas_total`. El indice del buffer pasa a ser `primera_libre` modulo `num_celdas_total`, ya que primera libre puede ser superior al tamaño del buffer. Además se incrementa `n`.

Semántica SC en múltiples prod/cons: implicaciones

- Describe razonadamente en tu portafolio a que se debe que (con semántica SC), la versión para múltiples productores y consumidores deba usar while, mientras que la versión para un único productor y consumidor puede usar simplemente if.
  - Se usa un while porque así se evita que haya un error si una hebra sale del wait y el buffer está lleno

Actividad sobre la barrera parcial con semántica SU

Describe razondamente en tu portafolio a que se debe que ahora, con la semántica SU, se cumplan las dos propiedades descritas.

- El orden de salida de la cita coincide siempre con el orden de entrada
  - Cuando la hebra señaladora libera a las hebras de la variable de condición estas salen en orden porque hay una política fifo
- Hasta que todas las hebras de un grupo han salido de la cita, ninguna otra hebra que no sea del grupo logra entrar.
  - Como la hebra señaladora tiene la exclusión mutua del monitor, el resto de hebras tienen que esperar a que termine

Actividad sobre el prod/cons con semántica SU

- Verifica si en el caso de la semántica SU es también necesario poner las operaciones wait dentro de un bucle while, o bien podemos sustituir dichos bucles por sentencias if. Describe razonadamente en tu portafolio a que se debe el resultado que has obtenido en el punto anterior.
  - El bucle while ya no es necesario ya que cuando la hebra se desbloquea, continua su ejecución sin que otra hebra interfiera en la condición

Práctica:

El problema de los fumadores

- Variable o variables permanentes: para cada una describe el tipo, nombre, valores posibles y significado de la variable.
  - Ingrediente: Representa el ingrediente sobre la mesa.Sus posibles valores son -1, 0, 2, 3
- Cola o colas condición: para cada una, escribe el nombre y la condición de espera asociada (una expresión lógica de las variables permenentes).
  - estanquero: ingrediente != -1
  - fumadores[3]: ingrediente != fumador
- Pseudo-código de los tres procedimientos del monitor.

```
ponerIngrediente(int ing) {
  ingrediente = ing;
  fumadores[ing].signal();
}

obtenerIngrediente(int fumador) {
  if (ingrediente != fumador)
    fumadores[fumador].wait();
  ingrediente = -1;
  estanquero.signal();
}

esperarRecogidaIngrediente() {
  if (ingrediente != -1)
    estanquero.wait();
}
```

El problema del barbero durmiente

- Decide que variables condición y variables permanentes son necesarias. Ten en cuenta que a veces las condiciones que esperan las hebras en una cola pueden incluir condiciones sobre el estado de

las otras colas.

  - Variables condición: barbero, silla, clientes

- Para cada variable condición, describe la condición asociada, las hebras que la usan y los puntos donde se hace wait/signal.
  - barbero: clientes.empty()
  - silla: clientes.empty()
  - clientes: (clientes.empty() && !barbero.empty() || !clientes.empty()
  - Todas las hebras hacen uso de todas las variables condición
- Escribe el pseudo-código del monitor.

```
class Mon_Barbero {
    CondVar silla, barbero, clientes;

    cortarPelo() {
      if (clientes.empty()) {
        if (!barbero.empty()) {
          barbero.signal();
        }
        else {
          clientes.wait();
        }
      }
      else {
        clientes.wait();
      }
      silla.wait();
    }

    siguienteCliente(){
      if(clientes.empty()){
        barbero.wait();
      }else{
        clientes.signal();
      }
    }

    finCliente(){
      silla.signal();
    }
}
```