Trabajo Practico N° 3

SCM - Sincronización con Monitores

Ejercicios a entregar: 3, 6,12 y 13.

1. Explique en un cuadro las diferencias y similitudes en el uso de las primitivas de semá- foros y monitores.

| **Aspecto** | **Semáforo** | **Monitor** |
| --- | --- | --- |
| **Ubicación** | Funciones independientes (no dentro de estructuras sincronizadas). | Dentro de métodos sincronizados. |
| **Efecto de wait()** | Decrementa el contador; bloquea si es menor que 0. | Libera el monitor y bloquea el hilo en una cola de espera. |
| **Efecto de signal()** | Incrementa el contador; desbloquea un proceso si hay alguno esperando. | Despierta a un hilo esperando (pero no ejecuta hasta que el monitor quede libre). |
| **Control del orden** | Manual (puede haber errores si se usan mal). | Automático (el monitor controla el acceso). |
| **Nivel de abstracción** | Bajo; el programador gestiona todo. | Alto; el sistema/entorno de ejecución gestiona la sincronización. |
| **Terminología en JR** | P(); V();  Wait(); Signal(); | \_wait(); \_signal() / notify() / notifyAll() |

1. Demuestre que semáforos y monitores son funcionalmente equivalentes implementan- do un semáforo general utilizando monitores. Muestre su uso mediante un programa ejemplo.
2. Demuestre que semáforos y monitores son funcionalmente equivalentes implementando una librería que permita simular un monitor valiéndose de semáforos. Muestre su uso mediante un programa ejemplo.
3. Use monitores “de Hoare” para implementar una solución al problema de los lectores y escritores.
4. Use monitores de Hoare para implementar una solución al problema de los lectores y escritores de modo tal que se vean favorecidos los lectores.
5. Use monitores de Hoare para implementar una solución al problema de los lectores y escritores de modo tal que se vean favorecidos escritores.
6. Utilice monitores con notify / broadcast para resolver el problema de lectores y escri- tores.
7. Utilice monitores de Hoare para proporcionar una solución al clásico problema “pro- ductores / consumidores” con las siguientes variantes:
   1. Búffer simple, **1** productor y **1** consumidor.
   2. Búffer ilimitado, **1** productor y **1** consumidor.
   3. Búffer ilimitado, **n** productores y **m** consumidores.
   4. Búffer acotado, **1** productor y **1** consumidor.
   5. Búffer acotado, **n** productores y **m** consumidores.
8. Utilizando monitores de Lampson & Redell, proporcione una solución al problema de productor / consumidor con un buffer limitado.
9. En relación al conocido problema de la Barbería, implemente una solución al mismo utilizando un monitor según la definición de Lampson & Redell en JR con las siguientes restricciones:
   1. Puede haber N clientes esperando en la barbería, además de quien está sentado en el sillón de corte.
   2. Hay un único sillón para corte y un único Barbero que además cobra en la única caja disponible.
10. Se necesita simular el funcionamiento de un pequeño módulo de un sistema que permite asignar N recursos de tipo X a m procesos que los requieren. Cada uno de los procesos necesita 2 recursos del tipo X para poder finalizar. Las experiencias muestran que la cantidad de procesos m es muy superior a la cantidad de recursos totales (N). Este simulador debe ser implementado

Utilizando monitores de Hoare, teniendo en cuenta:

* 1. Siempre N es > 1
  2. Especifique los supuestos que asuma.
  3. la solución debe mostrar también la desasignación de recursos.

1. Implemente una solución para el problema de la pizzería ya enunciado, utilizando mo- nitores de Hoare.
2. El sistema en cuestión es un doble procesador (cpu0 y cpu1), cuya carga de procesos es administrada por un monitor llamado despachador que Ud. debe implementar. Este monitor (despachador de procesos en este caso) se encarga entonces de asignar CPUs libre a los N procesos que la requieren por medio de una llamada al monitor **“despa- chador.adquirirCPU(Pid)”**. Cuando un proceso hace el requerimiento, el monitor verifica si se encuentra disponible alguna de las 2 CPUs y en caso afirmativo, sin mediar ninguna política de planificación en particular (salvo la propia disciplina de colas im- plementadas en el monitor) se le asigna. En caso de no haber ninguna CPU disponible el proceso que intenta la adquisición es bloqueado hasta que otro proceso en ejecución termine y ese procesador pueda ser nuevamente asignado. La solución debe ser tal que en todo momento el monitor debe conocer como mínimo el estado de cada uno de los procesadores (libre / ocupado) y al final de la corrida, debe proporcionar el total de asignaciones realizadas a cada uno de los procesadores.