Cuestionario guía - Clases Teóricas 5

1- Describa el funcionamiento de los monitores como herramienta de sincronización. Indicar como se realiza la comunicación, sincronización x EM y Sincronización x Condición entre los procesos.

Los monitores son módulos de programa con más estructura, y que pueden ser implementados tan eficientemente como los semáforos. Son un mecanismo de abstracción de datos:

- Encapsulan las representaciones de recursos.
- Brindan un conjunto de operaciones que son los únicos medios para manipula r esos recursos.

Contiene variables que almacenan el estado del recurso y procedimientos que implementan las operaciones sobre él.

La exclusión mutua está dada por los monitores. Es implícita. Un monitor va a estar siendo utilizado en un momento solo por UN proceso. Los demás se encolan y cuando el proceso termine de ser usado, se va a elegir uno encolado de forma no determinística.

En los programas concurrentes con monitores hay procesos activos y monitores pasivos. Dos procesos interactúan invocando procedures de un monitor.
Ventajas:

- Un proceso que invoca un procedure puede ignorar cómo está implementado.
- El programador del monitor puede ignorar cómo o dónde se usan los procedures.

Los monitores tienen interfaz y cuerpo. Sólo los nombres de los procedures (la interfaz) son visibles desde afuera. Los procedures pueden acceder sólo a variables permanentes, sus variables locales, y parámetros que le sean pasados en la invocación. La comunicación entre procesos será haciendo uso del monitor que va a contener los datos compartidos, que son variables o estructuras de datos accesibles por varios procesos. Dentro del monitor van a haber variables condicionales que permitirán a los procesos esperar o notificar eventos específicos.

La sincronización por condición es con variables condición: cond cv;
cv es una cola de procesos demorados, no visible directamente al programador.

- wait(cv): el proceso se demora al final de la cola de cv y deja el acceso exclusivo al monitor.
- signal(cv): despierta al proceso que está al frente de la cola (si hay alguno) y lo saca de ella. El proceso despertado recién podrá ejecutar cuando readquiera el acceso exclusivo al monitor.
- signal_all(cv): despierta todos los procesos demorados en cv, quedando vacía la cola asociada a cv.

Operaciones adicionales que NO SON USADAS EN LA PRÁCTICA sobre las variables condición:

- empty(cv): retorna true si la cola controlada por cv está vacía.

    - Solución: usar una variable contadora de la cantidad de procesos dormido. Cuando un proceso se va a dormir incrementa una variable: ej: espera ++. El proceso que se encarga de despertarlo la decrementa.
- wait(cv, rank): el proceso se demora en la cola de cv en orden ascendente de acuerdo al parámetro rank y deja el acceso exclusivo al monitor.
    - Solución: utilizo una cola.
- minrank(cv): función que retorna el mínimo ranking de demora.
    - Solución: utilizo una cola ordenada de min a max y siempre que hago signal(cv) despierto al mínimo.

2- ¿Qué diferencias existen entre las disciplinas de señalización "Signal and wait" y "Signal and continue"?

La diferencia principal entre ambas disciplinas recae en quien es el que va a ser encolado nuevamente para esperar el uso del monitor. En el caso de Signal and continued va a ser encolado (en la cola no determinística) el proceso despertado y en Signal and wait va a ser encola el proceso que despierta.

Signal and continued
El proceso que hace el signal continúa usando el monitor, y el proceso despertado pasa a competir por acceder nuevamente al monitor para continuar con su ejecución (en la instrucción que lógicamente le sigue al wait).

Signal and wait.
El proceso que hace el signal pasa a competir por acceder nuevamente al monitor, mientras que el proceso despertado pasa a ejecutar dentro del monitor a partir de instrucción que lógicamente le sigue al wait.

3- ¿En qué consiste la técnica de Passing the Condition y cuál es su utilidad en la resolución de problemas con monitores? ¿Qué relación encuentra entre passing the condition y passing the baton?

Passing The Condition se refiere a la transferencia de la condición que se está esperando de un proceso a otro. La idea central es permitir que un proceso notifique a otro sobre un cambio en una condición específica, lo que generalmente se realiza mediante el uso de variables condicionales.

Si hay un proceso dormido esperando por determinada condición, el que lo estaba bloqueando lo despierta, básicamente le pasa la condición para que pueda seguir ejecutándose. Si no hay ningún proceso bloqueado, se hará verdadera la condición para que el próximo proceso que venga pueda usarla. La idea central es que un proceso notifique a otro sobre un cambio en una condición específica
La utilidad en la resolución de problemas con monitores es que se respete el orden de espera de los procesos.

La relación entre passing the condition y passing the baton es que ambas técnicas se basan en el concepto de transferir el uso o acceso de recursos críticos entre procesos,

aunque passing the condition se centra más en la sincronización haciendo uso de condiciones y passing the baton en la transferencia de control de un proceso a otro.

4- Desarrolle utilizando monitores una solución centralizada al problema de los filósofos, con un administrador único de los tenedores, y posiciones libres para los filósofos (es decir, cada filósofo puede comer en cualquier posición siempre que tenga los dos tenedores correspondientes).

5- Sea la siguiente solución propuesta al problema de alocación SJN:

```
monitor SJN {
        bool libre = true;
         cond turno;

        procedure request(int tiempo) {
                if (not libre) wait(turno, tiempo);
                libre = false;

        }

        procedure release() {

                libre = true

                signal(turno);

        }

}
```

a) Funciona correctamente con disciplina de señalización Signal and Continue?

b) Funciona correctamente con disciplina de señalización Signal and Wait? EXPLIQUE CLARAMENTE SUS RESPUESTAS

6- Modifique la solución anterior para el caso de no contar con una instrucción wait con prioridad.

7- Modifique utilizando monitores las soluciones de Lectores-Escritores de modo de no permitir más de 10 lectores simultáneos en la BD, y además que no se admita el ingreso a más lectores cuando hay escritores esperando.

8- Resuelva con monitores el siguiente problema. Tres clases de procesos comparten el acceso a una lista enlazada: searchers, inserters y deleters. Los searchers sólo examinan la lista, y por lo tanto pueden ejecutar concurrentemente unos con otros. Los inserters agregan nuevos ítems al final de la lista; las inserciones deben ser mutuamente exclusivas para evitar insertar dos ítems casi al mismo tiempo. Sin embargo, un insert puede hacerse en paralelo con uno o más searches. Por último, los deleters remueven ítems de cualquier lugar de la lista. A lo sumo un deleter puede acceder la lista a la vez, y el borrado también debe ser mutuamente exclusivo con searches e inserciones.

9- El problema del "Puente de una sola vía" (One-Lane Bridge): autos que provienen del Norte y del Sur llegan a un puente con una sola vía. Los autos en la misma dirección pueden atravesar el puente al mismo tiempo, pero no puede haber autos en distintas direcciones sobre el puente.

a) Desarrolle una solución al problema, modelizando los autos como procesos y sincronizando con un monitor (no es necesario que la solución sea fair ni dar preferencia a ningún tipo de auto).

b) Modifique la solución para asegurar fairness (Pista: los autos podrían obtener turnos)