Preguntas del Segundo Cuestionario

1.- Sincronización de Procesos
Unir cada item de Lista A con un item de Lista B

Lista A
a) El problema del productor-consumidor
b) En el problema de los filósofos
c) Un semáforo
d) En el problema del barbero
e) Cuando se hace un notify
f) Cuando se hace un release

Lista B
a) no causa perdidas de tiempo por espera activa
b) es sincronizacion por competencia
c) se requiere sincronización tipo rendezvous
d) la señal no se pierde
e) se resuelve solo con un semáforo binario
f) es sincronización por cooperación
g) la señal puede perderse

- El problema del productor-consumidor es "sincronización por cooperación", dado que ambos tipos de procesos/hilos deben cooperar para poder progresar, es decir el productor solo podrá avanzar si el consumidor coopera consumiendo, y el consumidor sólo podrá avanzar si el productor coopera produciendo.
- El problema de los filósofos es "sincronización por competencia", dado que dos filósofos "compiten" por un tenedor para poder comer.
- Un sémaforo no causa perdidad de tiempo por espera activa, dado que el proceso que ejecuta la acción de adquisición del semáforo, si no consigue el permiso se bloquea y libera la CPU
- En el problema del barbero se requiere una sincronización tipo rendezvous entre el barbero y el cliente que se sienta en su sillón para un corte de pelo, dado que ambos tienen que "encontrarse" para poder progresar, el barbero necesita que haya un cliente en su sillón para poder realizar un corte y el cliente que se sienta en el sillón necesita que el barbero este disponible para realizar el corte.
- Cuando se hace un notify la señal puede perderse. Esto se da cuando un hilo hace un "notify" sobre un objeto cuyo conjunto de espera esta vacio. Se luego un hilo hace "wait" sobre el objeto y espera por el notify (que ya se hizo) nunca será despertado.
- Cuando se hace un release la señal no se pierde. esto es porque el resultado de un release es incrementar la cantidad de permisos disponibles en el semáforo, y esto se produce aún cuando ningun hilo este bloqueado por un acquire (o solicitud de permiso).

2.- Un semáforo tiene actualmente el valor 2. Si se ejecuta una operación "acquire" sobre él, ¿qué sucederá?

Seleccione la opción correcta:

a. Tras hacer la operación, el proceso continuará adelante sin bloquearse

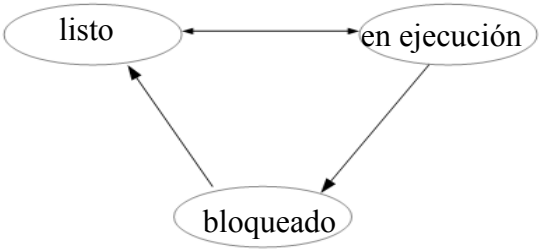
b. el proceso que ejecuta la operación se bloquea hasta que otro ejecuta una operación "release"

c. el proceso continuará sin bloquearse y, si previamente existían procesos bloqueados a causa del semáforo se desbloqueará uno de ellos

d. un semáforo jamás puede tener valor 2, si su valor inicial era 0 y se ha operado correctamente con él

Respuesta correcta: a) Tras la operación, el proceso continuará adelante sin bloquearse.
Esto es porque cuando se ejecuta un acquire sobre un semáforo, lo que se quiere es conseguir un permiso. Si el semáforo tiene valor 2, significa que tiene 2 permisos disponibles, y por lo tanto el proceso consigue el permiso y continua con su ejecución.

3.- En un escenario concurrente, con sincronización de procesos por el uso de espera pasiva, un hilo puede encontrarse en estado "en ejecución", "bloqueado" o "listo/runnable".
Ubique los estados de forma correcta en el diagrama.



Recuerde que un hilo que esta en estado "bloqueado" NO puede pasar directamente a estado "en ejecución", debe pasar previamente por estado "listo".

4.- Ubicar las opciones en los espacios.

En un ambiente concurrente con el modelo de *memoria compartida* se puede dar la *espera activa*, cuando los procesos/hilos están ejecutando instrucciones incluso cuando tienen que esperar para poder continuar.

En un ambiente de *multiprogramación* los procesos que están esperando están malgastando el procesador mientras otros procesos podrían utilizarlo para hacer trabajo útil. Este problema se resuelve con la *espera pasiva*: un proceso no puede continuar y se *bloquea* , liberando la CPU hasta que sea desbloqueado y pueda continuar con su ejecución.

En el modelo de memoria compartida se puede utilizar *monitores* para lograr una buena *sincronización* entre los procesos.

Opciones: multiprogramación, espera pasiva, bloquea, monitores, sincronización, memoria compartida, espera activa.

Multiprogramación: no es un modelo, es un ambiente de ejeción para un programa concurrentes. Se da cuando existen varios procesos ejecutados en un único procesador, con memoria compartida.
Memoria compartida: es un modelo para trabajar con concurrencia. En un ambiente concurrente se puede trabajar con memoria compartida, pasaje de mensajes (si es un ambiente distribuido), etc.
Espera pasiva: es la situación en la que se encuentra un hilo como resultado de hacer un wait sobre un monitor, por ejemplo.

5.- La operación de espera (acquire) de un semáforo y de una variable de condición de un lock se diferencian en:

Seleccione la opción correcta:
a. en el caso de la variable de condición se elimina la espera activa
b. en el caso de la variable de condición siempre se suspende el hilo que le aplica
c. no existe diferencia pues en ambos casos se usa para lograr la exclusión mutua de la sección crítica
d. no existe diferencia pues en ambos casos sirve como mecanismo para lograr la sincronización

La opción correcta es la b): *en el caso de la variable de condición siempre se suspende el hilo que la aplica.* Se refiere a la operación "await", que tiene el mismo efecto que la aplicación de un "wait" sobre un monitor, solo que la variable de condición está asociada a un lock (explicito) previamente adquirido. Una variable de condición NO se usa para lograr la exclusión mutua de la sección critica.

6.- Si se usa un semáforo para lograr la sincronización de procesos,

Seleccione la opción correcta:
a. las operaciones acquire/release se utilizan dentro del mismo hilo
b. las operaciones acquire/release se utilizan en procesos separados
c. se debe inicializar al numero de procesos/hilos que se desean sincronizar
d. se deben incluir variables de condición pues el semáforo únicamente proporciona exclusión mutua

La opción correcta es la b): *las operaciones acquire/release se utilizan en procesos separados.*Es decir el proceso que requiere que se de alguna condición producida por otro pide un permiso, y el proceso que actua y produce un cambio en el estado avisa liberando un permiso.

7.- La sincronización mediante monitores

Seleccione la opción correcta:
a. se consigue porque existe una única cola asociada a todos los procedimientos del monitor
b. se consigue mediante la utilización de variables de condición y esperas guardadas
c. es implícita, basta con invocar a la operación correspondiente del monitor
d. se consigue porque existe una cola asociada a cada procedimiento del monitor

La opción correcta es la b): *se consigue mediante variables de ocndición y esperas guardadas.*
Lo implícito es la exclusion mutua, NO la sincronización. La cola o conjunto de espera asociado al monitor no asegura la sincronización. Las variables de condición y esperas guardades se refiere a las variables (booleanas, enteras, etc) que se utilizan para verificar si se dan las condicinoes necesarias, y en conjunto con la operación "wait" sobre el monitor producen una "espera guardada".

8.- Unir cada item de Lista A con un item de Lista B

Lista A
a) La operación "wait" en Java
b) Un lock
c) El uso de variables de condición en Java
d) Los semáforos

Lista B
a) no sirven para exclusión mutua
b) no pueden tener asociadas varias variables de condición
c) puede aplicarse sobre cualquier objeto
d) debe hacerse asociado a un lock
e) sirven para exclusión mutua
f) puede tener asociadas varias variables de condición

- La operación "wait" en Java puede aplicarse sobre cualquier objeto. Todos los objetos disponen de la operación "wait" que es heredada desde Object, y que junto con la operación "notify" y el conjunto de espera dan a un objeto la semántica de monitor.
- Un lock puede tener asociadas varias variables de condición: Pueden crearse tantas variables de condición como sea necesario sobre un mismo lock.
- El uso de variables de condición en Java debe hacerse asociado a un lock: las variables de condición deben ser creadas antes de ser utilizadas (como cualquier otro objeto), y deben crearse "sobre" un lock creado previamente.
- Los semáforos sirven para exclusión mutua: cuando se los utiliza como semáforos binarios, y el permiso es adquirido y liberado por el mismo hilo antes y después de trabajar sobre la variable compartida, respectivamente.

9.- Los monitores proporcionan exclusión mutua porque

Seleccione la opción correcta:
a. solo un proceso puede estar activo cada vez para ejecutar un método del monitor
b. se diseñan mediante métodos encapsulados dentro de un modulo
c. no proporcionan exclusion mutua
d. para ello utiliza variables de condición

La opción correcta es la a), dado que un monitor por definición proporciona exclusión mutua. Que los métodos sean encapsulados dentro de un modulo no tiene relación con la exclusión mutua. LAs variables de condición no son para la exclusión mutua.