

Pregunta 1

0 / 5 puntos

Desde hace 3 años todos los procesadores son construidos con tecnología de 7nm

 ☒ Verdadero

 ☐ Falso

Pregunta 2

5 / 5 puntos

Establezca la correspondencia entre las dos columnas . A la izquierda están restricciones de sincronización y a la derecha los nombres de dichas restricciones

- ✓ 1

Un proceso debe acceder al mismo tiempo a los datos que otro

1. encuentro
- ✓ 2

Dos procesos no pueden acceder al mismo tiempo a una zona de datos

2. exclusión mutua
- ✓ 3



Un proceso se debe ejecutar antes que otro

3. señalamiento

Pregunta 3

0 / 5 puntos

¿Qué transición de estado realiza un proceso que ejecuta la instrucción  $P()$  sobre un semáforo que tiene el valor de su contador en 1?

-  ☒ De Listo a Ejecutando
- ☐ De Ejecutando a Dormido
-  ☐ Ninguna, podría seguir en Ejecutando
- ☐ De Ejecutando a Listo

Pregunta 4

5 / 5 puntos

En una máquina mono-procesador el uso de `yield()` tiene sentido si de alguna manera sabemos que el proceso que esperamos está ya en la cola de Listo

- ✓ ☒ Verdadero
- ☐ Falso

Pregunta 5 3 / 5 puntos

¿Cuáles de los siguientes ítems son compartidos entre múltiples threads que pertenecen al mismo proceso?

⇒ ✓ ☒ Datos estáticos

✓ ☐ Registros

⇒ ✓ ☒ Código

✓ ☐ Pila

⇒ ✗ ☐ Datos dinámicos

**Pregunta 6****3 / 25 puntos**

En este ejercicio usted debe escribir las soluciones del caso 1 para almacenar y retirar productos del buzón con espera pasiva, pero en pseudo-código utilizando semáforos. Para ello, primero escriba las soluciones que utilizó en Java y luego si traduzca su solución a semáforos.

Si bien el código en Java no tiene que ser exacto al del caso, si esa parte del caso funcionó bien, debería haber una alta similitud con lo allí entregado.

- a) 5% Código en Java de los dos métodos
- b) 10% Pseudocódigo de recibir
- C) 10% Pseudocódigo de enviar

**Sistema de traducciones en línea**

Tenemos un sistema cliente/servidor que funciona de la siguiente manera: existe un conjunto de  $C$  clientes que solicitan traducciones de palabras a un conjunto de  $S$  servidores que las responden.

Para solicitar una traducción, el cliente comunica la solicitud a través de un buffer, de donde los servidores las retiran y realizan las traducciones.

Tenga en cuenta que:

Un cliente debe esperar (espera pasiva) a que algún servidor traduzca su palabra. Una vez ha sido traducida, se despierta y debe encontrar su palabra traducida.

Un servidor espera (pasivamente) a que hayan solicitudes en el buffer.

Cuando logra tomar una solicitud realiza la traducción y despierta al cliente que la solicitó.

Implemente una solución en java, utilizando exclusivamente (no necesariamente todas) las funcionalidades vistas en clase: `synchronized`, `wait()`, `notify()`, `notifyAll()`, `CyclicBarrier`, `sleep()`, `join()`, `yield()`. Para ello:

a) 20% Escriba una clase `T` que extienda de `Thread` para representar a los clientes y servidores. Defina los atributos, implemente el constructor, el método `run()` y el main del programa principal.

Los clientes deben generar una solicitud y esperar su traducción, mientras que los servidores están en un loop infinito resolviendo solicitudes (y avisando a sus clientes cuando las han realizado).

b) 5% Escriba una clase `Buffer` para representar la comunicación de los clientes hacia los servidores. El buffer tiene una capacidad máxima de  $P$  solicitudes de traducción.

Defina los atributos, implemente el constructor e implemente los métodos c) y d) en esta clase:

c) 15% `almacenarSolicitud()`: recibe una solicitud y la coloca en el buffer. Si el buffer está lleno, espera (pasiva) por un espacio.

Si el buffer tiene espacio, adiciona la solicitud al buffer y espera por su traducción.

Note que si el buffer está lleno, se espera sobre el buffer, mientras que si se adiciona la solicitud, la notificación de la traducción se debe esperar sobre la solicitud.

Este método lo usarán los clientes.

d) 10% `retirarSolicitud()`: retorna una solicitud. Si existe al menos una en el buffer retorna la primera, de lo contrario espera (pasiva).

Este método lo usarán los servidores.

Suponga que ya existe (no hay que hacerla) una clase `Solicitud` para representar la palabra a traducir y su traducción (tiene dos atributos `String`: `palabraOriginal` y `palabraTraducida`) con los métodos de consulta, modificación, `generarSolicitud` (que define la palabra a traducir modificando el atributo `palabraOriginal`) y `generarTraducción()` que realiza la traducción (modificando el atributo `palabraTraducida`).