Programación Concurrente en Tiempo Real

Practica 4 - Ejercicio 1-2-3

Análisis

Para la realización de esta práctica y la comprobación de su correcto funcionamiento, se ha declarado una variable entera estática n inicializada a 0 en un principio. Además, cada hilo ejecuta un proceso:

- El proceso 1, suma 1 a la variable n por cada vez que se ejecute.

- El proceso 2, resta 1 a la variable n por cada vez que se ejecute.

Para comprobar su correcto funcionamiento la variable n debería valer 0 al final de la ejecución del programa.

1. Primer ejercicio

1.1. Tercer intento

Tras ejecutarlo varias veces, podemos apreciar que existe la posibilidad de que se produzca interbloqueo, producido como consecuencia de que ambos quieren acceder a sus respectivas secciones críticas simultáneamente.

Aunque no sabemos con exactitud en que momento se va a producir, podemos afirmar que este código(intento 3) no preserva la exclusión mutua y consecuentemente no sería un intento válido.

1.2. Cuarto intento

Aunque este ejercicio soluciona el problema de interbloqueo del intento anterior, donde ambos procesos quieren entrar en su secciones críticas. La solución sería forzando al otro proceso para que no intente entrar en su sección crítica.

Sin embargo, todo apunta a que esta solución no presenta ningún problema, puede darse el caso de que ambos procesos estén esperando a un evento que nunca ocurra y aunque es un escenario que se puede producir con poca probabilidad, consecuentemente tendríamos que considerar inválida esta solución.

2. Segundo ejercicio

Este algoritmo es parecido al cuarto intento, aunque con la clara diferencia de que en vez de derechos a entrada, se sustituye por el paso explícito a los procesos. Ofreciéndonos una solución al intento anterior.

Tras numerosas pruebas, apreciamos que no existe ninguna posibilidad de que ocurra un interbloqueo.

Podemos afirmar que este algoritmo es correcto, ya que no solo no produce interbloqueo, si no que además, mantiene la propiedad de exclusión mutua.

3. Tercer ejercicio

El algoritmo se basa en el algoritmo de Dekker, aunque en este, la diferencia fundamental, es decir, la espera de los procesos se combinan en una condición compuesta. Generando una solución aún mejor que la anterior.