ELVI MIHAI SABAU SABAU - 51254875L - 27/10/2022

PRACTICA 3

Algoritmos con espera ocupada

POSIX - A & B.

A

Esta implementacion aunque asegura la exclusion mutua mediante el uso de la variable `turno` no asegura que el programa funcione correctamente, debido a que cada hilo se pasa el turno entre ellos, y que si en algun momento alguno falla, el otro no podra continuar.

B

En esta implementacion, se soluciona el problema usando un array de booleanos para validar si cada hilo está o no en la sección crítica, aun asi, seguimos teniendo un problema de exclusion mutua, porque cada hilo verifica el estado del otro antes de cambiar su propio estado.

Java - B & C.

B

En esta implementacion seguimos sin asegurar la exclusion mutua.

C

En el tercer intento, se soluciona el problema de exclusion mutua comprobando el estado actual antes de comprobar el estado del otro hilo, impidiendo así que entren a la vez a la sección crítica, aun asi, puede generarse un interbloqueo si los hilos se quedan esperando a que cambie su estado.

NOTA: observa el atributo volatile para la variable turno, ¿qué ocurrre si lo eliminas?

Si eliminamos el atributo volatile, se podría dar la situacion en la cual uno de los hilos tenga asignado el valor antiguo al actualizar una variable del mismo objeto al mismo tiempo.

Al ejecutar el código vemos que siempre se encuentra en el hilo 0.

Python - C & D.

C

En esta implementacion, al igual que en Java, se provoca un interbloqueo.

D

En esta implementacion solucionamos el interbloqueo cambiando temporalmente el estado de `states[i]` a falso, e inmediatamente volver a ponerlo en verdadero.

De esta forma, se creamos un acceso temporal para que uno de los hilos pueda continuar su ejecucion, esto garantiza la exclusión mutua pero debido a ello nos arriesgamos a provocar un livelock.