ITESO

Universidad Jesuita de Guadalajara

Axel Roberto Orozco Hernández
NAVARRO QUINN, LUIS ROBERTO

Fundamentos de Sistemas Operativos
Profesor: Leonardo Sandoval. Gonzalez

Actividad 11

1.- Considere el siguiente programa que es el intento donde dos procesos P(0) y P(1) se sincronizan a través de un algoritmo por software empleando variables en memoria compartida.

P(0)	P(1)	Flag[0]	Flag[1]	Turno
		False	False	0
Flag[0] = true		True	False	0
	Flag[1] = true	True	True	0
	P(1) espera	True	True	0
P(0) entra SC		True	True	0
P(0) sale SC		true	True	0
Flag[0] = false		False	True	0
	P(1) entra SC	False	True	0
	P(1) sale SC	False	True	0
	Flag[0] = true	False	False	0

Explicacion del fallo: este algoritmo falla porque, en algunos casos uno de los procesos puede quedarse esperando indefinidamente debido a la condición de carrera en el manejo de la variable turno. Como ambos procesos pueden cambiar el valor de turno, se puede generar un estado donde ambos esperan por el turno y nunca avanzan, creando un interbloqueo.

2.- Considera el algoritmo de Dekker con la siguiente modificación con la intención de que funcione con 3 procesos. Mostrar que esta solución NO FUNCIONA, haciendo una tabla que muestre la secuencia de los 3 procesos concurrentes y los valores de las variables. El que no funcione correctamente no necesariamente es que dos o más entren a la sección crítica. Considera posibilidades como de que uno de los tres no inicie y los otros dos sí.

P(0)	P(1)	P(2)	Flag[0]	Flag[1]	Flag[2]	turno
			false	false	False	0
Flag[0] = true				false	False	0
	Flag[1] = true		True	True	false	0
		Flag[2] = true	True	True	True	0
	P(1) espera	P(2) espera	True	True	True	0
P(0) entra SC			True	true	True	0

P(0) sale SC			True	True	True	1
Flag[0] = false			False	True	true	1
	P(1) entra SC		False	true	true	1
	P(1) sale SC		False	false	true	2

Explicacion el fallo: Falla ya que debido a la forma en que se maneja la alternancia del turno, un proceso puede quedarse esperando mientras otros dos procesos alternan su entrada y salida de la sección crítica. El algoritmo no tiene mecanismo que garantice que todos los procesos entren a la sección crítica en algún momento.

3.- Considera el algoritmo de Peterson con la siguiente modificación con la intención de que funcione con 3 procesos. Mostrar que esta solución NO FUNCIONA, haciendo una tabla que muestre la secuencia de los 3 procesos concurrentes y los valores de las variables.

P(0)	P(1)	P(2)	Flag[0]	Flag[1]	Flag[2]	turno
			false	False	False	0
Flag[0] = true			true	false	False	1
	Flag[1] = true		True	True	False	1
		Flag[2] = true	True	True	True	1
P(0) espera	P(1) espera	P(2) espera	true	true	true	1

Explicacion del fallo: Éste algoritmo también sufre un problema de interbloqueo o inanición. La razón es que el manejo del turno y las banderas no es suficiente para garantizar que uno de los tres procesos no se quede esperando indefinidamente. Si turno es asignado a un proceso, pero otros dos procesos tienen sus banderas activas, pueden bloquear, impidiendo que todos los procesos accedan a la sección crítica.

¿Qué aprendimos?

1. Sincroniza exclusión mutua: aprendí como la falta de mecanismos de sincronización robustos, como mutexes o semaforos, puedes condiciones de carrera, inter bloqueos o inanición en algoritmos que usan variables compartidas.

2. Limitación de algoritmos clásicos: tanto el algoritmo de Decker como el de Peterson funcionan bien para dos procesos, pero sus versiones modificadas para más de dos procesos fallan debido a la complejidad añadida. Este ejercicio me mostró que es necesario usar avanzadas, como para manejar correctamente más de dos procesos.
3. Interbloque o inanición: vi como los procesos pueden quedar en un estado de inter bloqueo o sufrir de inanición, incluso cuando las variables de control parecen estar bien definidas.