**SINCRONIZACIÓN**

**PRIMER INTENTO:**

Deshabilitar interrupciones

EJERCICIO 1: Escriba como queda el protocolo de entrada y salida para esta solución y sus inconvenientes

while(true){

deshabilitar interupciones

SECCIÓN CRÍTICA

habilitar interrupciones

SECCIÓN NO CRÍTICA

}

Debido a que son operaciones de modo núcleo solo deben ser ejecutadas por el sistema operativo y eso resulta peligroso.

**SEGUNDO INTENTO:**

Compartir una variable de turno que indique quien puede entrar a la sección crítica

EJERCICIO 2: Escriba como queda el protocolo de entrada y salida para esta solución y sus inconvenientes

**PROCESO 0**

while(true){

while (turno != 0)

SECCIÓN CRÍTICA

turno = 1;

SECCIÓN NO CRÍTICA

}

**PROCESO 1**

while(true){

while (turno != 1)

SECCIÓN CRÍTICA

turno = 0;

SECCIÓN NO CRÍTICA

}

Alternancia estricta con lo cual se viola la segunda condición que es que la única forma en la que un proceso entre a su sección crítica que es los demás no estén en sección crítica.

**TERCER INTENTO:**

**En este intento se maneja la información de quienes están interesados que maneja el siguiente algoritmo**

EJERCICIO 3: Escriba el algoritmo para cada proceso y haga un análisis

int interesado[2]

proceso i (i=0,1)

while(true){

interesado[i] = TRUE;

while (interesado[1-i])

SECCIÓN CRÍTICA

interesado[i] = FALSE;

SECCIÓN NO CRÍTICA

}

**PROCESO 0**

while(true){

interesado[0] = TRUE;

while (interesado[1-0])

SECCIÓN CRÍTICA

interesado[0] = FALSE;

SECCIÓN NO CRÍTICA

}

Se recorre el arreglo de procesos y se coloca como interesado al proceso 0, mientras el proceso 1 no esté interesado este entra en sección crítica, después pasa a su salida dejando de estar interesado y sale.

**PROCESO 1**

while(true){

interesado[1] = TRUE;

while (interesado[1-1])

SECCIÓN CRÍTICA

interesado[1] = FALSE;

SECCIÓN NO CRÍTICA

}

Se recorre el arreglo de procesos y se coloca como interesado al proceso 1, mientras el proceso 0 no esté interesado este entra en sección crítica, después pasa a su salida dejando de estar interesado y sale.

EJERCICIO 4: Haga un seguimiento (Prueba de escritorio) al algoritmo de Peterson. Trate de asociar este algoritmo a un caso de la vida real.

Prueba para dos procesos

bandera[0] = 0

bandera[1] = 0

turno = 0

p0: bandera[0] = 1 p1: bandera[1] = 1

turno = 1 turno = 0

while( bandera[1] && turno == 1 ); while( bandera[0] && turno == 0 );

//no hace nada; espera. //no hace nada; espera.

// sección crítica // sección crítica

// fin de la sección crítica // fin de la sección crítica

bandera[0] = 0 bandera[1] = 0

Un caso de la vida real en la cual se puede aplicar el algoritmo es cuando dos personas llegan al mismo tiempo a un cajero y ambas quieren usarlo. Ambos manifiestan la intención de usarlo, pero entonces se establece un turno, podemos decir que el turno = 1 entonces la persona 1 puede usar el cajero, al terminar ya no tiene necesidad de usarlo por ende manifiesta que ya no quiere usarlo aún cuando sigue siendo su turno, y a partir de ahí la persona 2 puede usar el cajero.

EJERCICIO 5: investigue en internet una solución intermedia a la exclusión mutua que se dio con apoyo del hardware la instrucción TSL (TEST AND SET LOCK)

TEST AND SET LOCK (TSL, probar y fijar candado) La instrucción lee el contenido de la palabra de memoria, lo coloca en un registro y luego almacena un valor distinto de cero en esa dirección de memoria. Se garantiza que las operaciones de leer la palabra y guardar el valor en ella son indivisibles; ningún otro procesador puede acceder a la palabra de memoria en tanto la instrucción no haya terminado.

La CPU que ejecuta la instrucción TSL pone un candado al bus de memoria para que ninguna otra CPU pueda acceder a la memoria en tanto no termine.

* Verifica si la variable está ocupada
* La modifica y cambia el valor para bloquear su acceso
* Luego la desbloquea

**Ejemplo**

boolean lock = false

function Critical(){

while TestAndSet(lock)

skip //Hace un bucle hasta el bloqueo se quita

criticalSection //Solo un proceso puede estar en esta zona

lock = false //levantar bloqueo cuando termine con la zona critica

}