**Resumen teoria PC**

**Seccion critica:**

Propiedades a cumplir

**Exlusion mutua**: A lo sumo un proceso tiene que estar en su SC

**Ausensia de deadlock:** Si varios procesos están esperando par acceder a una SC, siempre al menos uno tiene que podedr entrar par que no se produzca deadlock

**Ausencia de demora innecesaria:** Si **únicamente** un proceso quiere entrar a la SC y el resto de procesoss esta en su SNC, el primer proceso debería entrar a SC.

**Eventual entrada:** Un proceso en algún momento tendrá que poder acceder a su SC.

**Sentencia await:**

**Busy / waiting:**

Es muy ineficiente, por ejemplo en un caso de monoprocesador todo el procesador estara siendo usado unicamente par chekear si esta disponible para su ejecucion, sin dejar ejecutar el proceso que realmente se debe ejecutar.

**Spin locks:**

Es lo mismo que el busy/waiting paro usando una instrucción de grano fino, TS.

Usa el Test and Set (TS(var: booleana)) mediante un whitle TS(var) skip.

Entonces skipea la interaccion esperando que se libere la SC par entrar.

Desventaja: Estos algoritmos cumplen las propiedades de sección critica, pero no aseguran del todo la eventual entrada, ya que un procesos podría ser re elegido constantemente par aacceder a su SC, dependiendo del tipo de scheduling usado.

**Soluciones fair:**

**Tie-Breaker:**

Funciona igual que los algoritmos de entrada pero además usa una variable que marca quien fue el ultimo proceso en ejecutarse, por lo que evita que se ejecute un mismo proceso 2 mismas veces.

Con N procesos es inviable, demasiado costoso y complejo.

**Ticket:**Se le asigna a cada proceso un numero de orden y una variable compartida que va marcando que turno se esta ejecutando. Es decir cada procesos se guarda su numero de turno y se queda interando esperando que la variable compartida sea igual a su turno, y en ese momento entra a su SC.

**Bakery:**

No usa variables compartidas, y los valores son auto asignados, cada proceso cuando llega busca el proceso máximo, lo aumenta en uno y se asigan ese valor.

Para elegir cual procesos se ejecuta se busca si mi numero de turno es el menor de todos.

Resumen:

Estos 3 últimos algoritmos mas o menos distibuyen equitativamente la ejecución de la SC, incluso con el uso de un scheduling débilmente fair.

**Sincronizacion barrera:**La barrera es un punto de demora donde deben llegar todos los procesos para podedr asi continuar todos juntos su ejecucion

**Contador compartido:**  
Cuando un proceso llega a la barrera se incrementa de forma atómica una variable compartida que indica cuantos processos llegaron a la barrera, una vez que el contador llega al numero de procesos, se procede a seguir con su ejecución.

Desventaja: si dicha barrera esta en un while el contador no podría volver a ser puesto en 0 por lo que estaría infinitamente aumentando, pudiendo llegar al punto donde reviente el programa.

**Flags y coordinadores:**

Se hace mediante el uso de 2 variables array de N procesos, donde en un arays “arribo” cada proceso marca en 1 cuando llega. Por otro lado el proceso coordinador espera a que todo el array arribo este en 1, lo resetea todo a 0 y pone el segundo array “continuar” en 1. Cada proceso cuando vuelve su ejecución pone su “continuar” en 0.

Al cada proceso tener su porpio slot en el array, no se deberá checkear por atomicidad al momento de modificar el array.

La desventaja es que se requerirá de un procesos únicamente para coordinar y además será una ejecución secuencial.

**Arboles:**

Es una solución mas compleja pero permite que vayan trabajando en simulateno.

Se basa en que cada nodo le avisa a su superior cuando termina y el superior al recibir la notificación de que sus 2 hijos terminaron, procede a su ejecución y posteriormente avisa a su padre. La desventaja de este modo es que se deberá hacer 3 algoritmos distintos, para las hojas, los internos y la raíz.

**Barrera simétrica:**

En vez de una única barera se hace uso de multiples barreras mas pequeñas, peritendo un mayor paralelismo.

**Butterfly berrier:**

Se separa en barreras por diferentes etapas (log2 N etapas), en cada etapa se sincroniza con un proceso diferente de otra etapa. De esta forma la comunicacion es unicamente con un proceso, pero la dificultad esta en determinar con quien se debe comunicar en cada etapa.

**Semaforo:**

Es un tipo de variable / objeto abstracto, con valores enteros positivos.

Posee 2 operaciones atomicas P y V. V es cuando ocurre un evento y se aumenta. P demora hasta que se cumpla un evento y decrementa.

Con estas 2 operaciones se puede lograr la sincronizacion por exclusion mutua y la sincronizacion por condicion, dependiendo como se implemente

Semaforo para exclusion mutua:

Se inicializa en 1 y se pone P y despues V

Semaforo de senalizacion: (Barrera)

Se inicializa 2 instancias en 0, y en ambos proceso se hace V y luego P.

Semaforo binario dividido:  
Para recursos compartidos con multiples procecsos (ej productores y consumidores), donde va a haber una alternancia entre ellos.

Se hace mediant eel uso de 2 instancias que alterna cuando puede entrar un tipo de proceso o otro