**Sincronización de procesos:**

Varios procesos de forma concurrente, no se interfieran entre sí en la ejecución.

Región critica: sección del codigo, en la cual los procesos están utilizando los mismos recursos compartidos, y la

Exclusión mutua: solo un proceso por vez pueda acceder a sus recursos compartidos.

Para lograr esto había distintas estrategias entre ellas los semáforos y los monitores (suministradas por el so).

Si utilizamos la exclusión mutua mediante semáforos:

* Ventajas: es confiable y no presenta espera activa
* Problemas: al usar múltiples procesos y semáforos será difícil de implementar y verificar a la hora de evitar deadlock y starvation.

Sin concurrencia, los procesos no interfieren entre si

Con concurrencia completa, un proceso está esperando por un recurso que tiene tomado por otro proceso y el otro proceso está esperando por un recurso que tiene tomado por el primer proceso (deadlock) y el último proceso está esperando un recurso que forma parte del deadlock, pero él no tiene tomado ningún recurso que genere dicho deadlock (starvation).

**Interbloqueo entre procesos:**

Deadlock (conocido como interbloqueo entre procesos):

* Conjunto de procesos en sistema concurrente que compiten por recursos del sistema están bloqueados permanentemente
* Estos están bloqueados por el deadlock esperando por un evento que no va a suceder

Condiciones:

* Mutua exclusión
* Tomar y esperar (toman un recurso y esperan por otro)
* Recursos no apropiativos (proceso que tiene tomado un recurso, será el encargado de liberarlo cuando no lo utiliza mas)

los primeros 3 se los considera necesarias

* Espera circular

Se la considera suficiente

Starvation (conocido como inanición):

* Cuando 1 o más procesos esperan de forma indefinida para acceder a un recurso de forma no activa de una espera circular

**Mecanismos para resolver deadlock y starvation:**

* ¡Estrategia del avestruz -> no hacer nada! (grave error)
* Prevenir el deadlock -> eliminar las condiciones del deadlock:

1. Mutua exclusión -> el so no permita usar mecanismos de sincronización (mala solución).
2. Tomar y esperar -> un proceso pida todos los recursos juntos (mala solución).
3. Recursos no apropiativos -> un proceso libere todos los recursos si debe esperar otro recurso (mala solución).
4. Espera circular -> un proceso pida los recursos en orden creciente, con lógica de id, si pido un recurso de id 4 o superior, no puedo pedir luego una id menor. (mala solución).

Por ende, eliminar las condiciones fue una pésima idea, la prevención se desechó.

* Evitar el deadlock -> asignar los recursos a medida que son solicitados siempre que no produzcan una espera circular. Para implementar esta asignación de procesos nació:

Algoritmo del banquero: cada vez que el so tenga que asignar un recurso cualquiera a un proceso, va a simular si ese proceso puede finalizar por los recursos que ya tiene y lo que pueda necesitar a futuro, se lo da. Si no puede finalizar, no se lo asigna (puede producir espera circular).

* + - Ventajas: nunca se producirá un deadlock.
    - Problemas: costoso para el sistema y se requiere conocer por adelantado todos los recursos que va a solicitar cada proceso.
* Evitar y detectar el deadlock -> se aplica actualmente, el so en forma recurrente analiza si entre los procesos que están en estado bloqueados, se produce una espera circular.

El so simula si estos procesos en algún momento podrán finalizar o no.

* Si no hay, no hace nada
* Si hay, se elimina
  + Liberar recurso/s tomado/s
  + Rollback de proceso/s (estado seguro de los procesos)
  + Matar proceso/s (se libera el recurso, lo toma otro proceso y termina la ejecución)

Si tengo una espera circular de 5 procesos, con matar o rollback de 1 proceso, puede ser suficiente para liberar recursos y terminar la ejecución.

Su coste es similar a la estrategia anterior, pero será ejecutado cada cierto tiempo (1 vez por hora/día).

* Ventajas: es la menos costosa para el sistema.
  + - Problemas: se puede producir deadlock y y se requiere conocer por adelantado todos los recursos que va a solicitar cada proceso.