Sincronización

**Definición:** Capacidad de que varios procesos se ejecuten al mismo tiempo pudiendo llegar a interactuar entre ellos.

Puede ser:

* **Real**: Tengo varios núcleos/procesadores y por cada uno un proceso en ejecución.
* **Simulada**: Tengo un solo núcleo encargado de ejecutar procesos concurrentes. Cada uno de ellos se realiza de forma alternada. También se puede dar cuando tengo más procesos en ejecución que procesadores o núcleos.

**Tipos de procesos desde el punto de vista de la sincronización:**

* **Independientes**: Durante su ejecución no necesitan nada de otro proceso. **Cooperativos:** Necesitan trabajar con otros procesos para llevar a cabo su tarea.

Los procesos pueden percibirse de distintas maneras.

* **No** percibirse (independientes) No percibe otros procesos, pero compite por los recursos. En la **competencia** (se encarga SO) existe
  + la **Exclusión mutua**. Hay recursos que pueden ser utilizados solo por un proceso a la vez. Puede suceder que no pueda restaurar un proceso (no puedo volver al estado en el que lo dejo).
  + **Interbloqueo:** Bloqueo mutuo entre dos procesos. A usa recurso 1, necesita recurso 2. B necesita recurso 1 y utiliza recurso 2. Puede llevar a
  + **Inanición:** Quedan a la espera de un recurso que no se liberará.
* Se perciben **indirectamente** (al compartir recursos) Hay cooperación en el uso del mismo. En este caso existe la **Cooperación** (se encarga el programador de organizar los recursos no compartibles) A los recursos no compartibles se los denomina **recursos críticos.** La sección de código en cual utilizo a este recuso crítico es la **sección crítica.**

Esta se debe realizar atómicamente (una atrás de la otra, como si fueran una sola instrucción, para que no lo afecten las interrupciones).Debe estar encerrada entre **sección de entrada** (aviso que voy a usar recursos críticos y doy las instrucciones para su uso) y la **sección de salida** (le quito el uso del recurso crítico). El resto del código se llama sección restante.

|  |
| --- |
| Sección restante |
| Sección de entrada |
| Sección crítica |
| Sección de salida |
| Sección restante |

Se sigue el protocolo de la sección critica, debe haber:

* + Exclusión mutua
  + Progreso – A qué proceso de los que están en sección de entrada le asigno el uso del recurso.
  + Espera limitada – Tengo varios procesos en la sección de entrada pidiendo por uso del recurso crítico. Se lo doy al de mayor prioridad, pero el que queda esperando no puede tener una espera indefinida, para evitar inanición.

Para encarar la solución de la sección crítica es:

1. Inhibir interrupciones (cuando estoy en mono núcleo).
2. Sentencias específicas (tengo tantos assemblers como cantidad de procesadores, diferente para cada procesador)
3. Algoritmo de Dekker: Es difícil de aplicarlo, no se suele utilizar
4. Algoritmo de Peterson: Solo permite sincronizar un par de procesos, pero su ventaja es su simplicidad, trabaja con 2 variables de tipo global, una de ellas es un vector (variable señal) y otra una variable sencilla (turno).

Turno: indica a que proceso le corresponde acceder a su sección critica

Señal: Indica cual está en condiciones de entrar a su sección critica, osea que está en la sección de entrada. 0 está en sección de entrada, 1 no está. El primer espacio es para proceso A y el otro para B.

Al finalizar el proceso seleccionado se invierten los valores en ambas señales.

1. Uso de **semáforos**:
   1. Un semáforo es una variable global de tipo entero con valores positivos o negativos dependiendo de cómo haya sido diseñada. Se define con la cantidad de instancias que tenga el recurso crítico. (semáforo contador). Los valores negativos representan la cantidad de proceso bloqueados.
   2. Binario o mutex: Se coloca un 0 cuando el recurso está libre y se pone en 1 al estar ocupado . Si la selección la hace el planificador de corto plazo, es semáforo blando, si elijo yo qué proceso pasa al recurso crítico es duro.
2. **Monitor** - Herramientas de software para sincronizar procesos.

* Se perciben **directamente** y cooperan en el uso de los recursos.

La **comunicación** entre procesos puede ser:

* **Directa** - cuando se envían mensajes entre un proceso y otro. Puedo mandar/recibir identificación de proceso y mensaje, o número de mensaje y mensaje. Si es igual de los dos lados me determinará simetría o asimetría de la comunicación.
* **Indirecta** – Con buzón, un lugar de la memoria al cual se le asigna un nombre. Un proceso puede tener un buzón propio, en cuyo caso el proceso es el propietario, si el proceso termina, el buzón deja de existir o lo hereda un proceso hijo. Solo un proceso puede recibir mensajes en el buzón. Tanto envío como recepción pueden ser:
  + **Bloqueantes**:
    - B deja mensaje y se bloquea hasta que A confirma recepción. (envío)
    - A queda bloqueado hasta recibir mensaje en buzón. (recepción)
  + **No bloqueantes**:
    - Proceso A pasa por buzón, si hay mensaje lee, sino sigue (recepción)
    - B deja mensaje y sigue, no espera confirmación (envío)

El buzón puede tener capacidad:

* + Nula – B se tiene que bloquear si deja un mensaje – envío con bloqueo.
  + Limitada – Puedo dejar determinada cantidad de mensajes, los primeros se hacen sin bloqueo y los siguientes seguirán con bloqueo porque el buzón no admite más mensajes.
  + Ilimitada – Hasta que se llena la memoria le puedo mandar mecha. Los procesos no se bloquean nunca.

**Indirecto se puede abrir en sincrónico o asincrónico**

**Sincrónica** – Se bloquea

**Asincrónica** – No se bloquea

**Deadlocks – Abrazos mortales**

Es un bloqueo perpetuo de un conjunto de procesos donde cada uno de ellos está bloqueado esperando por la ocurrencia de un evento que debe producir uno de los procesos del conjunto.

Las condiciones que deben darse para la ocurrencia de este interbloqueo son:

* Exclusión mutua:
* Espera y retención: Los recursos que son recibido por un proceso los retiene mientras espera otros recursos que necesita
* Sin desalojo
* Espera circular: Cadena de procesos cerrada

Se puede hacer mediante grafos o mediante matrices.

En grafos puedo haber nodos, de proceso o recurso

Y aristas, que van de un nodo proceso o recurso, indicando cuantas instancias del recurso puedo

Las asignaciones van del recurso al proceso

En el caso de matrices voy a tener una para asignaciones, otra para solicitudes, un vector para indicar total de recursos y otro vector para indicar cuantos de esos recursos están disponibles.