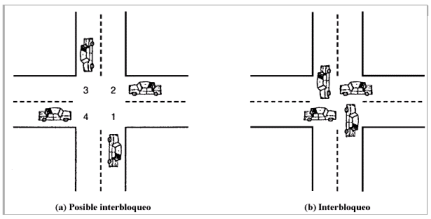
Tema N° 4

SUBTÍTULO DEL INFORME

Nombre | Nombre del curso | Fecha

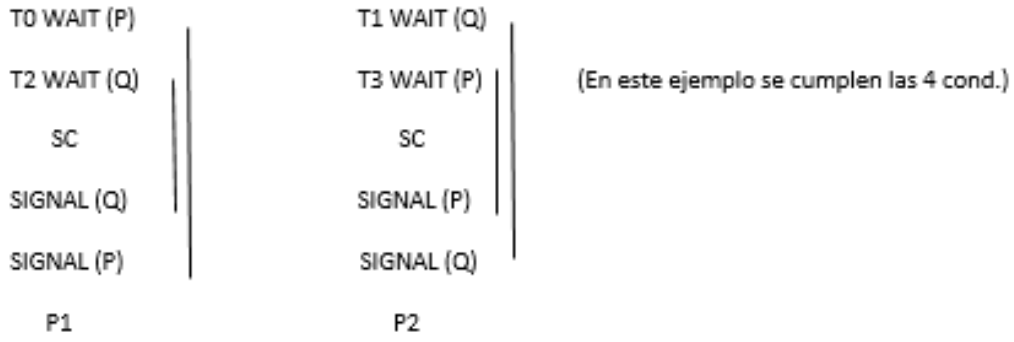
BLOQUEO MUTUO



El Bloqueo es una consecuencia de las secciones críticas. Se presenta porque los recursos dentro del sistema operativo son finitos.

CONDICIONES PARA QUE SE PRESENTE BLOQUEO INDEFINIDO

- ✓ **Exclusión Mutua:** solo uno puede tener el recurso, básicamente esto se da cuando los recursos no son compartibles. Por ejemplo: la impresora, si un proceso tiene la impresora hasta que el no termine de imprimir nadie más podrá imprimir.
- ✓ **Retener y Esperar:** tengo recursos, pero tampoco puede ejecutarme porque estoy esperando otro recurso que necesito. Ejemplo el problema de los Filósofos Comensales.
- ✓ **Sin Desalojo, No Apropiación o No Expulsión:** es decir que lo que tiene nadie se lo puede quitar.
- ✓ **Espera Circular:** cada proceso espera por el recurso del otro, el proceso 1 espera el recurso del proceso 2 y este a su vez espera por el recurso del proceso 1.



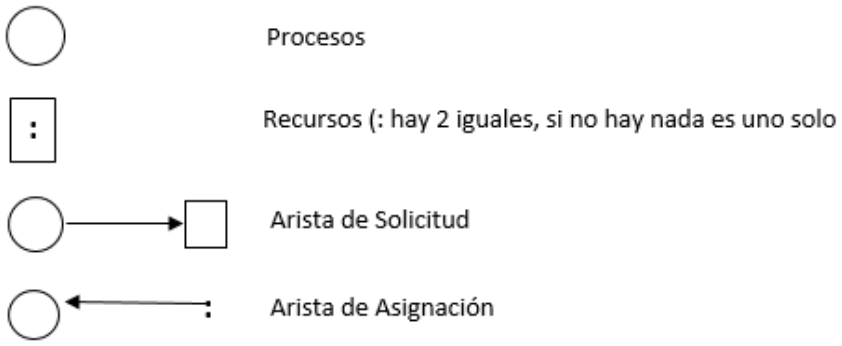
Están bien programadas, pero ¿Qué ocurre?

Cuando transcurren los T0 y T1, cuando P1 quiera ejecutar la siguiente instrucción en T3 queda BLOQUEADO porque esta esperado Q y P2, también queda BLOQUEADO (T3) porque está esperando P. Entonces se cumplen las 4 condiciones, los dos se bloquean mutuamente; los dos están en retención y espera, a ninguno de los dos se le puede quitar el recurso porque están en un semáforo; están esperando y están en una espera circular porque uno espera por el otro.

"Las 4 condiciones son necesarias y son suficientes cuando se tienen un solo recurso del mismo tipo", pero cuando se tienen más recursos del mismo tipo o sea más de uno, no puede darse las 4 condiciones y haber problemas.

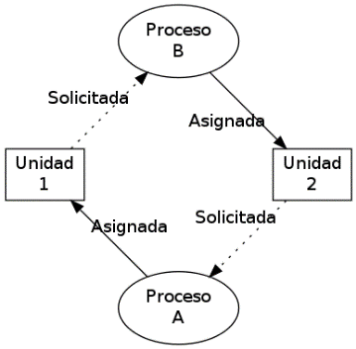
Cuando decimos dos recursos del mismo tipo, significa que desde el punto de vista del proceso le da igual, en uno o en el otro, no hay diferencia.

GRAFOS

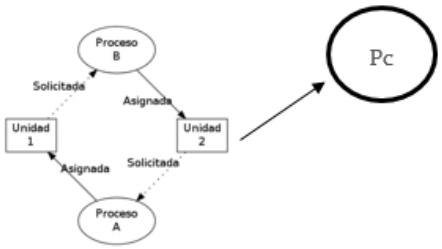


"Todo proceso primero debe solicitar el recurso, después se le asigna y lo debe liberar".

Ejemplo (código llevado a grafo)

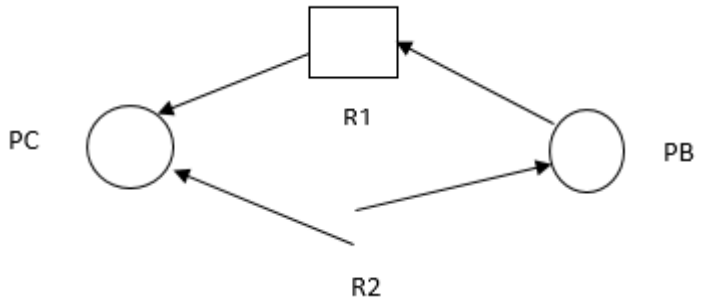


Ahora, supongamos que tenemos un proceso Procesos C y tenemos dos recursos Unidad 2:



Continúa el ciclo, pero tenemos un Proceso C que tiene uno de los recursos de Unidad 2 asignado, PB y PA están en espera. Como proceso C era lo único que necesitaba se puede ejecutar, usar y liberar.

Entonces en un tiempo X, Pc va a liberar el recurso Unidad 2, al desaparecer Pc y liberar el recurso, ese elemento que es del mismo tipo se le puede asignar a PA, entonces se rompe el ciclo.



- Por eso se dice que las 4 condiciones son necesarias y suficientes si hay uno de cada tipo de recurso, o sea un solo ejemplar de cada tipo de recurso.
- Ahora si los recursos tienen más de un ejemplar del mismo tipo, no importa si hay ciclo, lo que hay que ver es si hay una posibilidad de romper el ciclo, y las 4 condiciones ya no son suficientes.

METODOS PARA MANEJO DE BLOQUEOS

- **ALGORITMO DEL AVESTRUZ:** no hacer nada, ignorar el problema. Sistemas monousuarios.
- **PREVENIR EL INTERBLOQUEO:** se basa en la negación de una de las 4 condiciones de bloqueo, es decir prevenir el interbloqueo, es negar una de las condiciones.
  - **NEGAR LA EXCLUSIÓN MUTUA:** es muy difícil negar porque es parte de la sección crítica. La exclusión mutua depende del tipo de recursos que se trate, por ejemplo, no se puede hacerla exclusión mutua cuando tenemos un solo micro, es decir tienen que entrar uno por vez.

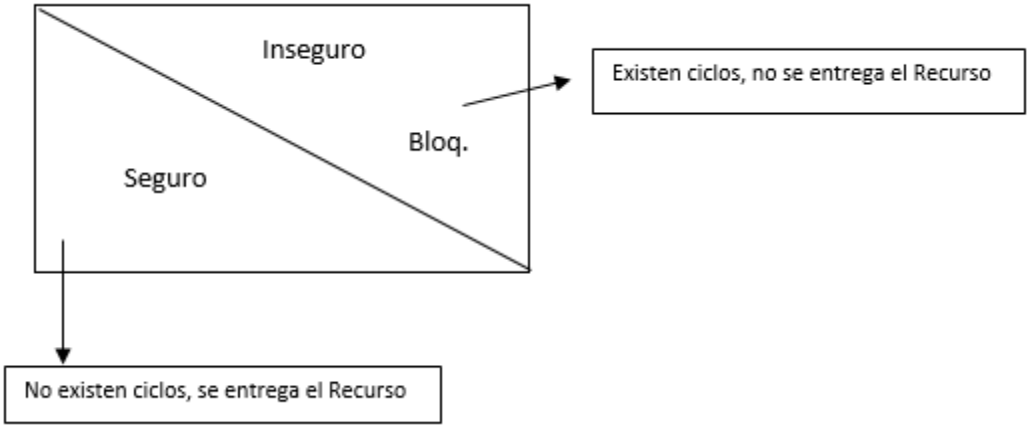
  La única forma que podríamos hacer es cuando manejamos memoria por ejemplo en vez de tener una variable usamos un array de variables, o sea generar una estructura de datos que nos permita controlar los BUFFER, son la forma de tratar de negar la exclusión mutua.

  - **EL DESALOJO**: tenemos que impedir la apropiación de un recurso, poder desalojar o liberar el recurso. Eso por ejemplo el S.O lo usa en un tiempo compartido. En memoria usamos PMP, liberamos memoria la bajamos a disco y la subimos después.
  - **LA NEGACIÓN DE RETENER Y ESPERAR**: lo que hacer es; en cuanto a programación es obligar a los procesos que pidan por anticipado los recursos que necesiten. Si no se les puede entregar todos los recursos el proceso debe esperar que, es lo que pasa en NUEVO.

  Y lo largo de la ejecución del proceso, para poder pedir más recursos no debe tener ninguno asignado, o sea pido recursos cuando puede me los da. Los usa y los libera para poder pedir recursos después. El problema es que puede haber un desuso de recursos.

- **LA NEGACION DE LA ESPERA CIRCULAR**: se logra dándole valores a todos los recursos y los procesos solo pueden pedir recursos mayores a los que tiene asignado, para esto existe el vector de interrupciones, en este están enumerados todos los recursos y todas las interrupciones que existen. (Los recursos pueden ser lógicos y físicos).

➢ **EVITACION DE BLOQUEO:** se usa la definición de Estado Seguro. Se basa en usar un algoritmo que determina si estamos en un ESTADO SEGURO o sea un ESTADO INSEGURO. Si el sistema con una asignación de recursos está en un estado seguro el proceso se ejecuta. Si el pedido del recurso lo pasaría al sistema a un estado inseguro no se entrega el recurso.

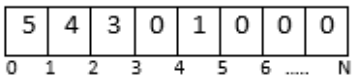


Para hacer esto corre un algoritmo por cada asignación, ese algoritmo se llama:

ALGORITMO DEL BANQUERO

Para generar el algoritmo del banquero se genera un array que se llama DISPONIBLE donde hay n elementos, donde cada uno de las posiciones tiene la cantidad de recursos del mismo tipo.

DISPONIBLES



Luego tenemos una **Matriz de Máximos,** recursos por cada proceso o sea cada línea de la matriz es un proceso.

	R0	R1	R2	...	RN
P1					
P2					
:					
:					
Pn					

→ Cada proceso que existe con su MAX necesidad

Cada vez que se pide una asignación se genera una tupla por proceso.

ASIGNACIÓN P1 (1, 0, 1, 0, 0) →es una matriz

La que se llama **NECESIDAD:**

NECESIDAD = MAXIMO – ASIGNADO

Por cada solicitud que se pida se tiene que preguntar:

Se tienen que cumplir
- Si SOLICITUD <= NECESIDAD (no se puede pedir más de lo que se necesita)
- Si SOLICITUD <= DISPONIBILIDAD (no se puede pedir más de lo que hay)

Simular como quedaría el sistema si se da la Solicitud:

Es un **Estado Seguro**
- DISPONIBLE = DISPONIBILIDAD – SOLICITUD (para ver que queda)
- ASIGANCION = ASIGNACION + SOLICITUD
- NECESIDAD = NECESIDAD – SOLICITUD

Se verifica que a menos uno de los procesos tenga satisfecha su necesidad, si las tiene satisfecha, quiere decir que ese proceso se ejecuta y las libera o tenemos disponible para satisfacer las necesidades de algunos de los procesos que quedan.

Por cada vez que un proceso pide una solicitud se tiene que hacer todo esto.

El problema de este mecanismo es que si bien es un mecanismo que es seguro, hace más lenta la asignación de recursos, o sea disminuye el rendimiento del microprocesador.

La Evitación sobrecarga el SISTEMA, porque en realidad está agregando un proceso más que siempre se tiene que ejecutar.

- Detección y Recuperación: consta de dos partes.
  1- Detectar que existe un interbloqueo, o sea que hay procesos que no responden.
  2- Recuperarse de ese bloqueo.

- <u>Abortar los procesos</u>: hay que contemplar a quien vamos a abortar, que no es nada fácil. Tratar de elegir la victima que menos victimas genere después.

<u>Expropiación</u>:

En realidad, el Sistema Operativos usa estrategias combinadas, para los recursos internos, por ejemplo:

- ✓ el PCB usa una prevención a través del orden de los recursos, esto es fácil porque el PCB tiene el identificador de Proceso entonces lo que hace es COLA FIFO en todos lados y con eso hace prevención.

- ✓ La Memoria Central lo haría por Expropiación, la memoria principal gracias al PMP baja los procesos al disco o sea quita los recursos, niega la retención y espera.
- ✓ Los recursos de trabajo lo hacen a través de la Evitación como siempre tiene lo disponible gracias al vector de interrupciones y a la tabla de dispositivos, utiliza el algoritmo del banquero para entregar o no.
- ✓ Y para los espacios intercambiables, o sea los Discos lo hace con Asignación previa, se pide la capacidad máxima de almacenamiento.