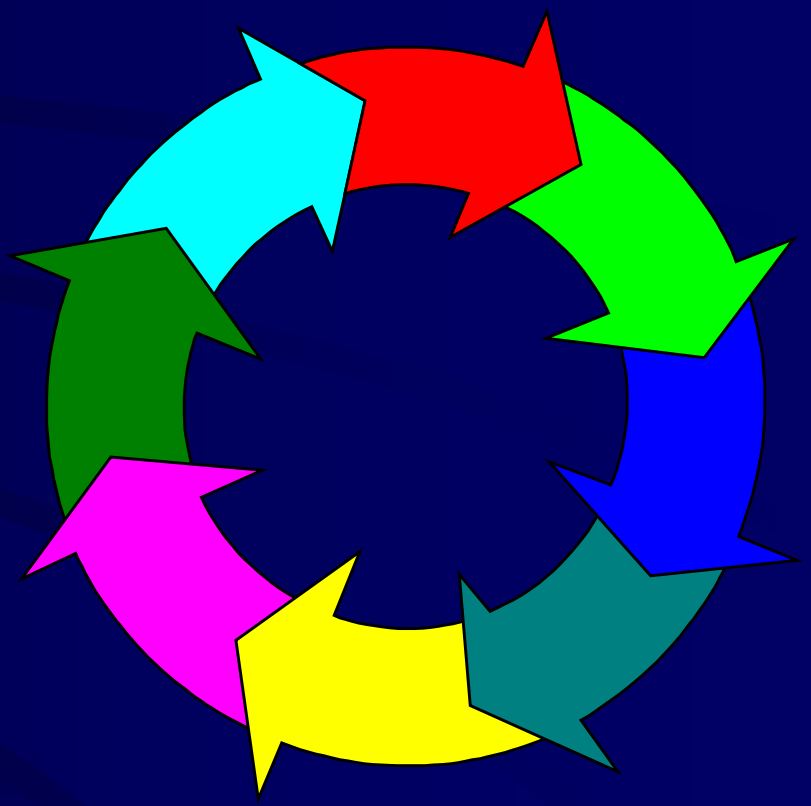
SISTEMAS OPERATIVOS

- **DEADLOCK**
- **ABRAZO MORTAL**
- **BLOQUEO MUTUO**



RECURSOS

Un sistema se compone de un número finito de recursos que se distribuyen entre varios procesos que compiten por ellos: Ciclos de CPU, Espacio de Memoria, Archivos, Dispositivos.

Un proceso debe solicitar un recurso antes de usarlo y liberarlo al terminar su uso.

En el modo de operación normal, un proceso sólo puede utilizar un recurso en la secuencia siguiente:

- Solicitud.

- Utilización.

- Liberación.

DEADLOCK
(BLOQUEO MUTUO, ABRAZO MORTAL)

Un conjunto de procesos se encuentra en estado de bloqueo mutuo, cuando cada uno de ellos espera un suceso que sólo puede originar otro proceso del mismo conjunto.

Los sucesos tienen que ver con la adquisición y liberación de los recursos. Estos recursos pueden ser físicos (Impresoras,unidades de cinta, espacios en memoria, ciclos de CPU, etc.) o lógicos (archivos, semáforos, monitores, etc).

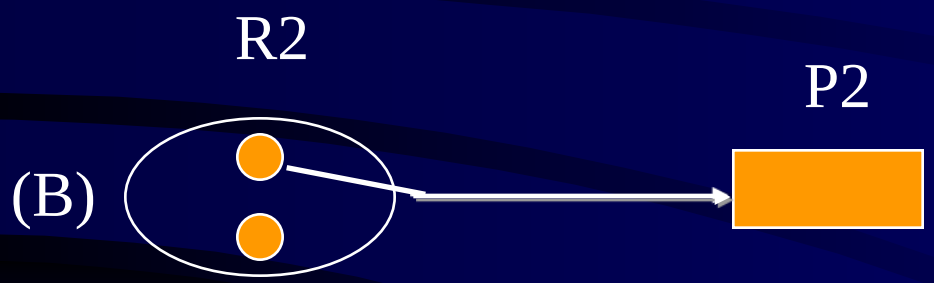
En los bloqueos mutuos, pueden involucrarse uno o varios tipos de recursos.

CONDICIONES NECESARIAS PARA QUE OCURRA UN DEADLOCK

- EXCLUSIÓN MUTUA: Los procesos exigen un control exclusivo de los recursos que necesitan.
- RETENCIÓN Y ESPERA: Los procesos mantienen la posesión de los recursos ya asignados a ellos mientras esperan por recursos adicionales retenidos por otros procesos.
- NO APROPIACIÓN: Un recurso sólo puede ser liberado voluntariamente por el proceso que lo retiene, después que haya cumplido su tarea.
- ESPERA CIRCULAR: Debe existir un conjunto de procesos (p0,p1,...,pn) en espera, tales que p0 espera un recurso retenido por p1, p1 espera un recurso retenido por p2 y así sucesivamente hasta que pn espera un recurso retenido por p0.

GRAFICA DE ASIGNACIÓN Y PETICIÓN DE RECURSOS

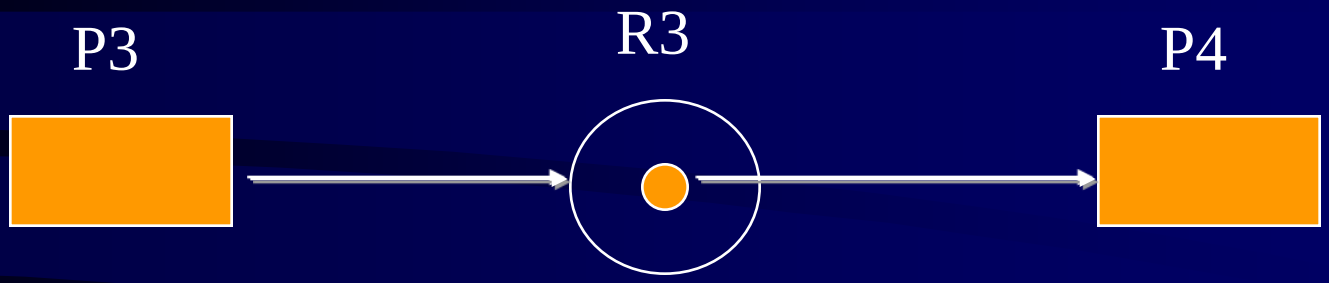
P1 pide un recurso de tipo R1.



Un recurso de tipo R2 a sido asignado al proceso p2.

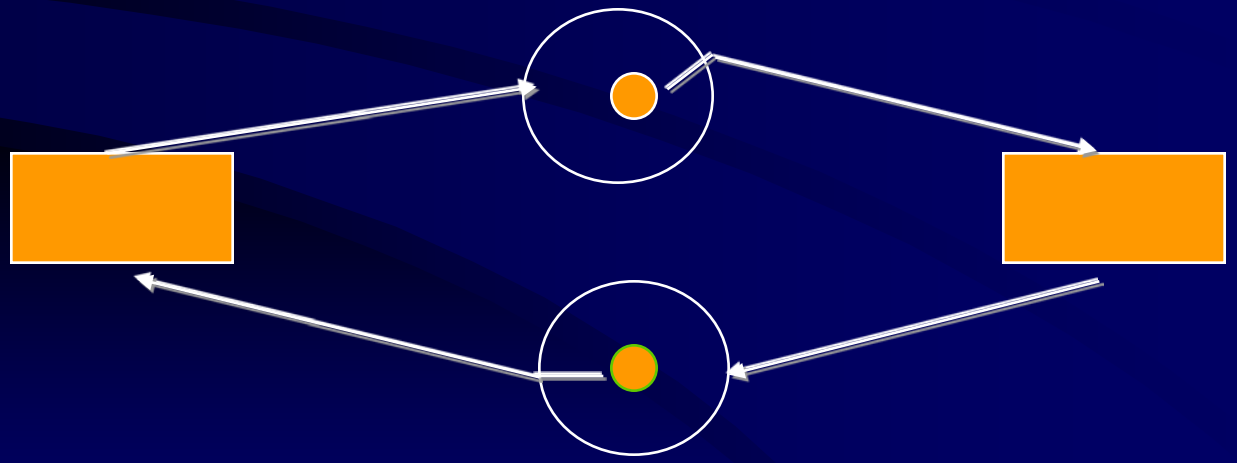
GRAFICA DE ASIGNACION Y PETICION DE RECURSOS

(C)

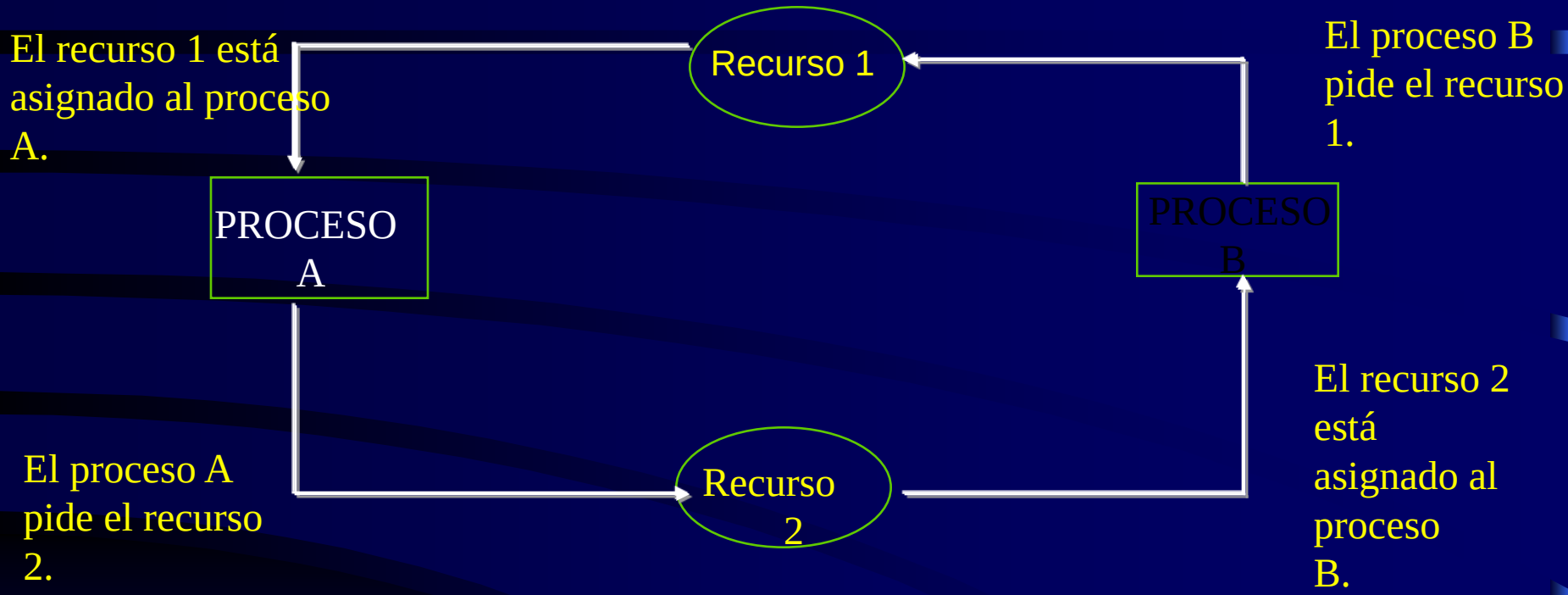


El proceso P3 pide el recurso R3, el cual ha sido asignado al proceso P4.

(D)



UN BLOQUEO MUTUO



Este sistema está bloqueado porque cada proceso tiene un recurso solicitado por el otro proceso y ninguno de ellos está dispuesto a liberar el recurso que tiene.

METODOS PARA MANEJAR EL BLOQUEO MUTUO

- PREVENIR: Ajustar todo el sistema para ELIMINAR TODA POSIBILIDAD que ocurra un deadlock.
- EVITAR: Se PERMITE la posibilidad del bloqueo mutuo, pero se ESQUIVA cuando está a punto de suceder.
- DETECTAR: DETERMINAR SI HA OCURRIDO un bloqueo mutuo y saber exactamente cuáles son los procesos y los recursos involucrados en él.
- RECUPERAR: ELIMINAR el bloqueo mutuo de un sistema para que pueda seguir trabajando y para que los procesos implicados puedan terminar su ejecución y liberen los recursos utilizados.

PREVENCIÓN DEL BLOQUEO MUTUO

Conjunto de estrategias desarrolladas por Havender:

1- Cada proceso deberá pedir TODOS sus recursos al mismo tiempo y no podrá seguir la ejecucion hasta haberlos recibido por completo.

NEGACIÓN DE LA CONDICIÓN DE RETENCIÓN Y ESPERA.

↗ RETENCIÓN Y ESPERA: Los procesos mantienen la posesión de los recursos ya asignados a ellos mientras esperan por recursos adicionales retenidos por otros procesos.

PREVENCIÓN DEL BLOQUEO MUTUO

2- Si a un proceso que tiene ciertos recursos se le NIEGAN los demás, ese proceso deberá LIBERAR sus recursos, y en caso necesario pedirlos de nuevo conjuntamente con los recursos adicionales.

NEGACIÓN DE LA CONDICIÓN DE NO APROPIACION

↗NO APROPIACIÓN: Un recurso sólo puede ser liberado voluntariamente por el proceso que lo retiene, después que haya cumplido su tarea.

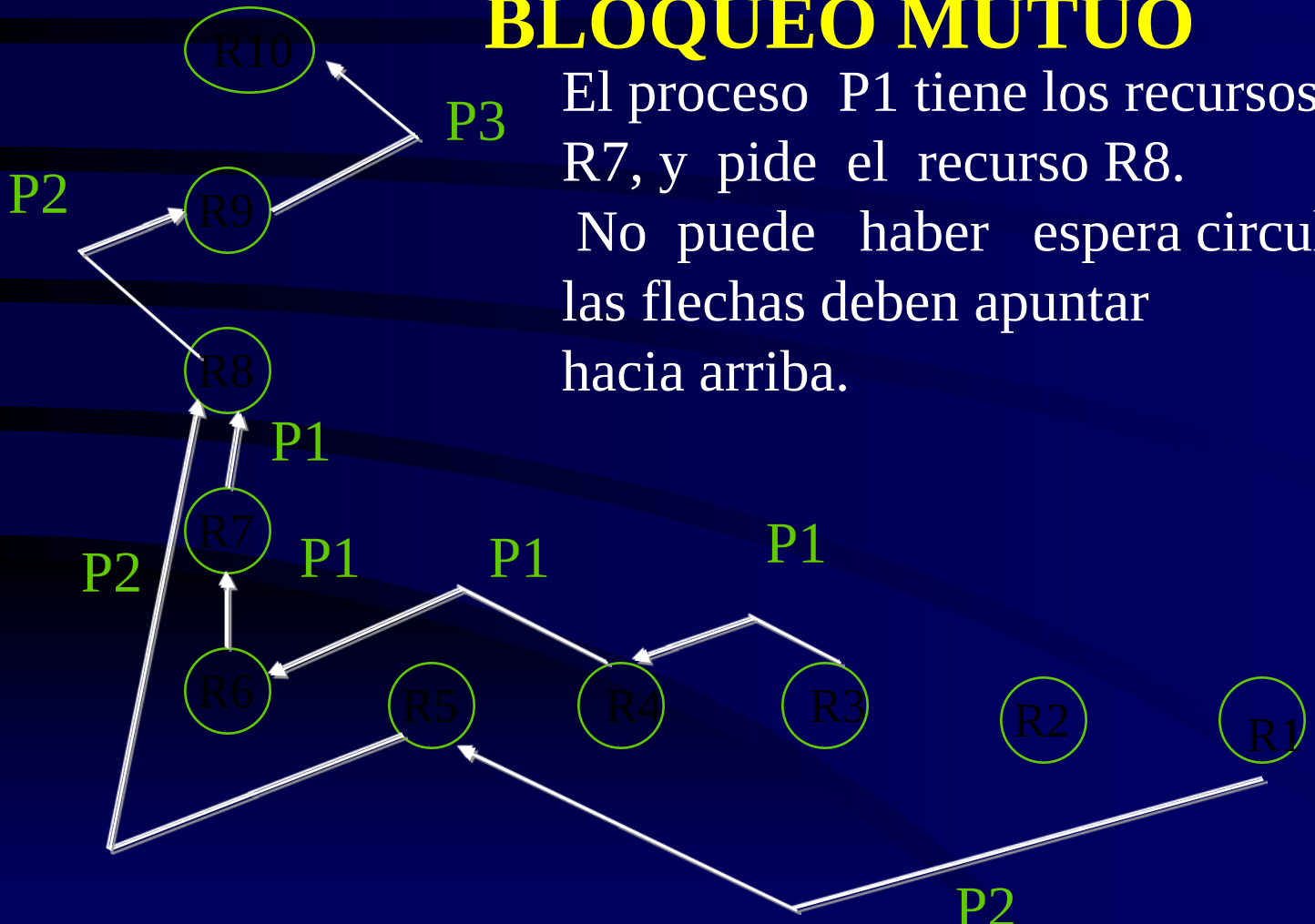
PREVENCIÓN DEL BLOQUEO MUTUO

3- Si a un proceso le han asignado recursos de un tipo especifico, en lo sucesivo sólo podra pedir aquellos recursos que sigan un determinado ORDEN. Este ordenamiento esta realizado en forma LINEAL.

NEGACIÓN DE LA CONDICIÓN DE ESPERA CIRCULAR.

ESPERA CIRCULAR: Debe existir un conjunto de procesos (p0,p1,..., pn)en espera, tales que p0 espera un recurso retenido por p1, p1 espera un recurso retenido por p2 y así sucesivamente hasta que pn espera un recurso retenido por p0.

ORDENAMIENTO LINEAL DE HAVENDER PARA PREVENIR EL BLOQUEO MUTUO



El proceso P1 tiene los recursos R3, R4, R6 y R7, y pide el recurso R8.

No puede haber espera circular porque todas las flechas deben apuntar hacia arriba.

PREVENCIÓN DEL BLOQUEO MUTUO

4- La condición de EXCLUSION MUTUA debe conservarse para aquellos recursos que por naturaleza propia o función no pueden compartirse.

HAVENDER NO ESTABLECE ESTRATEGIAS PARA ESTE CASO.

EVITAR EL BLOQUEO MUTUO

Si se presentan las condiciones necesarias para un bloqueo mutuo, todavía es posible evitarlo mediante una cuidadosa asignación de recursos.

ALGORITMO DEL BANQUERO.

EVITAR EL BLOQUEO MUTUO

ALGORITMO DEL BANQUERO. (Condiciones Iniciales)

1- El sistema operativo administra un número fijo de unidades por recurso entre un número fijo de usuarios.
2- Cada usuario especifica por adelantado el número máximo de unidades de los recursos que necesitará durante la ejecución de los trabajos.
3- El sistema operativo aceptará la petición de un usuario si la necesidad máxima de ese usuario no es mayor al número fijo de unidades del recurso.
4- Un usuario puede obtener o liberar unidades del recurso una a una. Los recursos asignados no podrán ser mayores a las necesidades máximas declaradas por dicho usuario.

EVITAR EL BLOQUEO MUTUO

5- Si el sistema operativo es capaz de satisfacer la necesidad máxima del usuario, entonces este proceso debe garantizar al sistema operativo que las unidades del recurso serán utilizadas y liberadas en un tiempo finito.

6- Uso de los términos: “SISTEMA EN ESTADO SEGURO" y “SISTEMA EN ESTADO INSEGURO".

EVITAR EL BLOQUEO MUTUO

Se dice que el sistema se encuentra en estado SEGURO, si el sistema operativo puede permitir que todos los procesos actuales terminen sus trabajos en un tiempo finito. En otro caso, el estado del sistema es INSEGURO.

Ejemplo:	Préstamo	Necesidad
Procesos	Actual	Máxima
Usuario 1	1	4
Usuario 2	4	6
Usuario 3	5	8

Unidades disponibles:2

La clave para que un sistema sea seguro es que exista al menos una forma adecuada de que terminen todos los procesos.

EVITAR EL BLOQUEO MUTUO

Ejemplo: Procesos	Préstamo Actual	Necesidad Máxima
Usuario 1	8	10
Usuario 2	2	5
Usuario 3	1	3

Unidades disponibles:1

Un sistema inseguro indica que alguna secuencia desafortunada en la asignación de recursos podria llevar al bloqueo mutuo.

ESTRUCTURAS DE DATOS PARA EL ALGORITMO

n: número de procesos, m:tipos de recursos

- Disponible:Un vector de longitud m que indica el número de recursos disponibles de cada tipo. Si Disponible[j]=k, entonces hay k ejemplares disponibles del tipo de recurso R_j.
- Máx. Una matriz de n*m que define la demanda máxima de cada proceso por cada clase de recurso. Si máx[i,j]=k, entonces el proceso Pi puede solicitar como máximo k ejemplares del tipo de recurso R_j.
- Asignación: Una matriz de n*m que define el número de recursos de cada tipo asignados en ese momento a cada proceso. Si asignación[i,j]=k, entonces el proceso p_i tiene actualmente asignados k ejemplares del tipo de recurso R_j.
- Necesidad:Una matriz de n*m que indica los recursos que le hacen falta a cada proceso. Si necesidad[i,j]=k, entonces el proceso p_i puede necesitar k ejemplares más del tipo de recurso R_j para completar su tarea. Observe que Necesidad[i,j]=Máx[i,j]-Asignacion[i,j].

IMPLEMENTACION DEL ALGORITMO DEL BANQUERO

Sea $Solicitud_i$ el vector de solicitudes para el proceso P_i. Si $Solicitud_i[j]=k$, entonces el proceso P_i quiere k ejemplares del tipo de recurso R_j. Cuando el proceso P_i efectúa una solicitud de recursos, se emprenden las siguientes acciones:

1- Si $solicitud_i <= Necesidad_i$, continuar en el paso 2. De lo contrario, presentar una condición de error, ya que el proceso se ha excedido de su demanda máxima.

2- Si $Solicitud_i <= Disponible_i$, continuar en el paso 3. De lo contrario, P_i deberá esperar, pues los recursos no están disponibles.

3- El sistema simula haber asignado todos los recursos solicitados al proceso p_i modificando el estado de la manera siguiente:

$Disponible:Disponible-Solicitud_i$;

$Asignacion_i:=Asignacion+Solicitud_i$;

$Necesidad_i:=Necesidad-Solicitud_i$;

IMPLEMENTACION DEL ALGORITMO DEL BANQUERO

Si el estado de asignación de recursos resultante es seguro, entonces se efectúa la transacción y los recursos se asignan al proceso p_i.

Sin embargo, si el nuevo estado no es seguro, entonces P_i, deberá esperar a que se sirva $Solicitud_i$ y se restablece el anterior estado de asignación de recursos.

ALGORITMO PARA DETERMINAR EL ESTADO DE UN SISTEMA

1- Sean Trabajo y Fin vectores de longitud m y n, respectivamente.
Asígnese Trabajo:=Disponible y Fin[i]:=falso para toda i=1,2,...,n.

2- Encontrar una i tal que se cumplan ambas proposiciones:

a) Fin[i]=falso

b) $Necesidad_i$<=Trabajo.

Si no existe tal i, continuar en el paso 4.

3- Trabajo:=Trabajo+$Asignación_i$

Fin[i]:=verdadero

Continuar en el paso 2.

4- Si Fin[i]=verdadero para toda i, entonces el sistema está en un estado seguro.

DETECCIÓN DE BLOQUEO MUTUO

- Si un sistema no emplea un algoritmo de prevención o evitación de bloqueo mutuo, entonces deberá:
  - Determinar si ha ocurrido un bloqueo mutuo.
  - Recuperarse del bloqueo mutuo.

  Esto requiere tiempo de procesamiento adicional e involucra pérdidas potenciales de procesos y recursos.

ALGORITMO DE DETECCIÓN

- Disponible. Un vector de longitud m que indica el número de recursos disponibles de cada tipo.
- Asignación: Una matriz de n*m que define el número de recursos de cada tipo actualmente asignados a cada proceso.
- Solicitud: Una matriz de n*m que indica la solicitud actual de cada proceso. Si solicitud[i,j] = k, entonces el proceso p_i solicita k ejemplares más del tipo Rj.

ALGORITMO DE DETECCIÓN

ACCIONES:

1.Sean Trabajo y fin vectores de longitud m y n respectivamente.

Asignar Trabajo:=disponible

Para i=1,2,...,n, si asignación$_i$ <> 0, entonces Fin[i]=falso;

de lo contrario, Fin[i]=verdadero.

2.Encontrar un índice i tal que se cumplan ambas proposiciones:

a) Fin[i]=falso

b) Solicitud$_i$<=Trabajo.

Si no existe tal i, continuar en el paso 4.

3.Trabajo:=Trabajo+Asignación$_i$

Fin[i]:=verdadero

continuar en el paso 2

4.Si Fin[i]=Falso, para una i, 1<=i<=n, entonces el sistema está en un estado de bloqueo mutuo. Es más, si Fin[i]=falso, entonces el proceso p_i está en bloqueo mutuo.

RECUPERACIÓN DESPUÉS DE UN DEADLOCK

Permitir al sistema operativo recuperarse después de la ocurrencia de un deadlock, sin intervención directa del operador.

Alternativas:

i) Abortar uno o más procesos.

ii) Arrebatar recursos a uno o más procesos que se encuentran en deadlock.

ABORTAR

MÉTODOS:

i) Abortar todos los procesos en deadlock:

* Costo alto.

* Pérdida total del trabajo realizado.

ii) Abortar un proceso a la vez, hasta que el deadlock haya sido eliminado:

* Overhead (Sobretiempo).

* Algoritmo de detección.

RECUPERACIÓN DESPUÉS DE UN DEADLOCK

FACTORES PARA LA ESCOGENCIA DEL PROCESO.

1- Prioridad de los procesos.

2-Cuánto tiempo se ha ejecutado el proceso y cuánto tiempo necesita para culminar su actividad.

3-Cuántos y qué tipos de recursos ha usado el proceso.

4-Cuántos recursos más necesita el proceso para culminar.

5-Cuántos procesos necesitan ser terminados.

6-Si el proceso es interactivo o en lotes.

ARREBATAR RECURSOS

Quitar recursos a los procesos en forma sucesiva y otorgar éstos recursos a otros procesos.

Aspectos a considerar:

1- SELECCIÓN DE UN PROCESO “VICTIMA".

Determinar a cuáles procesos y cuáles recursos les serán arrebatados.

Parámetros:

* Número. de recursos que tiene asignado un procesos en deadlock.
* Cantidad de tiempo consumida por el proceso en su ejecución.

ARREBATAR RECURSOS

2- ROLLBACK:

Una vez que un proceso le han sido quitados los recursos, se puede regresar el proceso a algún estado seguro y restaurarlo desde ese momento.

Requiere que el sistema mantenga mayor cantidad de información acerca del estado de todos los procesos en ejecución.

3. STARVATION (APLAZAMIENTO INDEFINIDO):

Garantizar que los recursos no siempre sean arrebatados a los mismos procesos “Víctimas“, ya que ésto aplazaría su culminación.