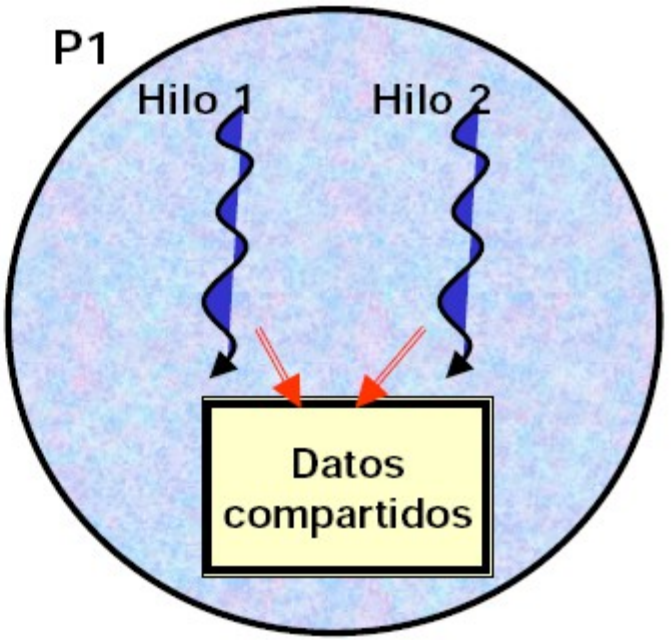
Sincronizacion de Procesos

Sincronizacion de Procesos

- Existe la necesidad de comunicación entre procesos.
- Los procesos requieren con frecuencia comunicación entre ellos (ejemplo: tubos).
- La comunicación entre procesos puede seguir dos esquemas básicos:
  - Comunicación por memoria común
  - Comunicación por mensajes

Sincronizacion de Procesos

- **Comunicación por memoria común**
  - La comunicación por memoria común se puede dar en los siguientes casos:
    - Espacio de direcciones único: es el caso de los hilos de ejecución
    - El s.o. crea una zona de memoria accesible a un grupo de procesos
  - **Problema de la sección crítica**: en un sistema con procesos concurrentes que se comunican compartiendo datos comunes es necesario sincronizar el acceso (lectura, escritura) a los datos compartidos para asegurar la consistencia de los mismos.



Sincronizacion de Procesos

Comunicación por mensajes

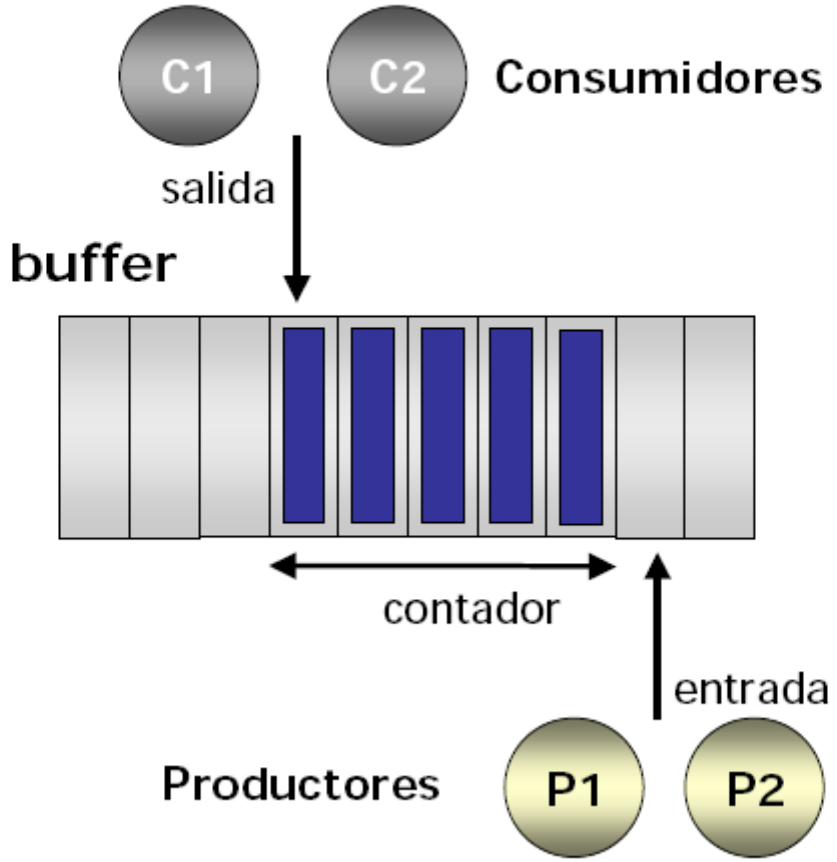
- La comunicación por mensajes se da "normalmente" en el siguiente caso:
  - Espacio de direcciones independentes
- Sincronización en la comunicación por mensajes: Cuando dos procesos se comunican vía mensajes se necesitan mecanismos para que el proceso receptor espere (se suspenda hasta) a que el proceso emisor envíe el mensaje y éste esté disponible.
- No aparece el problema de la sección crítica.

Sincronizacion de Procesos

- El problema de los productores y consumidores con buffer acotado
  - **Productor**: proceso que produce elementos (a una cierta velocidad) y los deposita en un buffer.
  - **Consumidor**: proceso que toma elementos del buffer y los consume (a una velocidad probablemente diferente a la del productor)
  - **Buffer**: Estructura de datos que sirve para intercambiar información entre los procesos productores y consumidores. Actúa a modo de depósito para absorber la diferencia de velocidad entre productores y consumidores
  - Ejemplo: buffer de impresora.

Sincronizacion de Procesos

- El problema de los productores y consumidores con buffer acotado



Tipos de procesos

Independientes	Cooperantes
Su estado no es compartido	Su estado es compartido
Son deterministas	Su funcionamiento no es determinista
Son reproducibles	Su funcionamiento puede ser irreproducible
Pueden ser detenidos y rearrancados sin ningún efecto negativo	Si son detenidos y posteriormente rearrancados puede que se produzcan efectos negativos
Ejemplo: un programa que calcula 1000 cifras decimales de pi	Ejemplo: un proceso que escribe en el terminal la cadena “abc” y otro la cadena “cba”

Sincronizacion de Procesos

Atomicidad

- Una operación se dice que es atómica (en un sistema uniprocesador) cuando se ejecuta con las interrupciones deshabilitadas
- Las referencias y las asignaciones son atómicas en la mayoría de los sistemas
- Esto no es siempre cierto para matrices, estructuras o números en coma flotante
- Típicamente la arquitectura proporciona operaciones específicas para lograr la atomicidad
- Si el hardware no proporciona operaciones atómicas, éstas no pueden construirse por software

Sincronización

Persona A	Persona B
3:00 Mira en la nevera. No hay leche	
3:05 Va a la tienda	
3:10 Llega a la tienda	Mira en la nevera. No hay leche
3:15 Deja la tienda	Va a la tienda
3:20 Llega a casa y guarda la leche	Llega a la tienda
3:25	Deja la tienda
3:30	Llega a casa y ...

Sincronizacion de Procesos

Cual es el problema planteado?

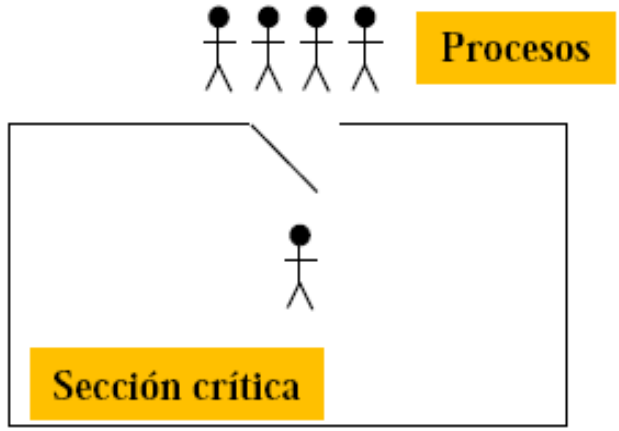
- Alguien necesita leche, pero no tanta
- Definiciones:
  - **Exclusión mutua**: es el mecanismo que asegura que sólo una persona o proceso está haciendo algo en un instante determinado (los otros están excluidos)
  - **Sección crítica**: es la sección de código, o colección de operaciones, en el que se actualizan variables comunes

  Cuando un proceso está ejecutando código de su sección crítica, ningún otro proceso puede estar en esa misma sección crítica

Sincronizacion de Procesos

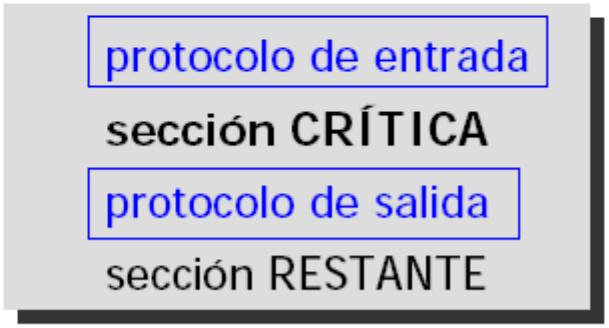
El problema de la sección crítica

- Enunciado del problema (Dijstra, 1976):
  - "n" procesos ejecutan repetidamente una sección no crítica, SNC, seguida de una **sección crítica**, SC
  - La sección crítica de un proceso es un secuencia de acciones que debe ser ejecutadas en **exclusión mutua** con las secciones críticas del resto de procesos.
    - Siempre que un proceso "entra" en la SC, "sale"



Sincronizacion de Procesos

- **Formulación del problema de la sección crítica**
  - Sean n procesos compitiendo para acceder a datos compartidos
  - Cada proceso tiene una zona de código, denominada sección crítica, en la que accede a los datos compartidos.
  - Problema: encontrar un protocolo del tipo:



Que satisfaga las tres condiciones siguientes :

Sincronizacion de Procesos

El problema de la sección crítica

- **Objetivo**: Diseñar los protocolos de entrada y salida (mecanismos de sincronización) a la sección crítica de forma que se satisfagan los siguientes requisitos:
  - **Exclusión mutua**: como máximo un proceso puede estar ejecutando su sección crítica
  - **Ausencia de bloqueos**: si dos o más procesos tratan de acceder a su sección crítica, al menos uno lo logrará
  - **Ausencia de retrasos innecesarios**: si ningún proceso está en la SC y uno solicita entrar, puede entrar sin esperar
  - **Ausencia de situaciones de inanición (individual)**: todo proceso que desee entrar en su SC, tarde o temprano entrará

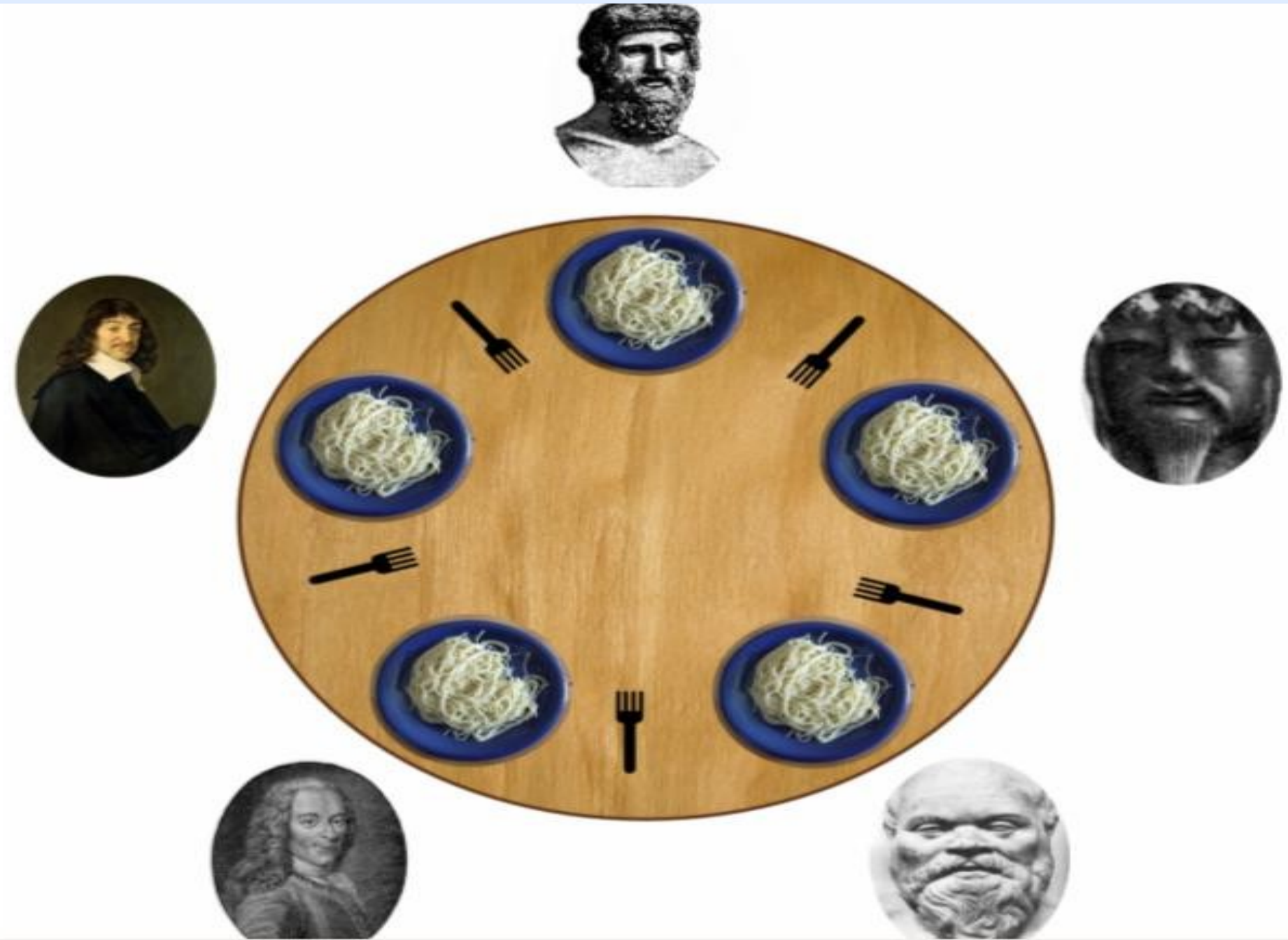
Sincronizacion de Procesos

- Solución al problema de la sección crítica
  - Supuestos:
    - Los procesos se ejecutan a velocidad no nula.
    - La corrección no ha de depender de hacer suposiciones sobre la velocidad relativa de ejecución de los procesos.

ABRAZO MORTAL - DEADLOCK

Un conjunto de procesos está en estado de "DEADLOCK" cuando cada proceso del conjunto está esperando por un evento que solo puede ser causado por otro proceso que está dentro de ese conjunto.

ABRAZO MORTAL - DEADLOCK



ABRAZO MORTAL - DEADLOCK

CONDICIONES NECESARIAS

- **Exclusión Mutua:** debe haber recursos no compartidos de uso exclusivo.

- **Espera y Retenido (Hold & Wait):** un proceso retiene un recurso y espera por otro

- **Sin Desalojo:** no es posible quitarle el recurso retenido a un proceso. Lo libera voluntariamente

- **Espera circular:** hay un ciclo de espera entre procesos

ABRAZO MORTAL - DEADLOCK

Conceptos.

- Bloqueo permanente de un grupo de procesos que compiten por los recursos del sistema o bien se comunican unos con otros.
- Un sistema consiste en un número finito de recursos que será distribuidos entre los procesos dentro del sistema.
- Ejemplos Recursos: CPU, memoria, disco, etc.

ABRAZO MORTAL - DEADLOCK

Modelo del Sistema.

- Tipos de recursos: R1,R2,….,Rn: ciclos de CPU, espacio de memoria, E/S
- Cada Recurso R_i posee W_i instancias.
- Cada proceso usa los recursos
  - -. solicita o hace un requerimiento
  - -. los usa
  - -. los libera (al final)

ABRAZO MORTAL - DEADLOCK

Modelo del Sistema.

El uso de los recursos solo puede ser hecho por medio de llamadas al sistema (System Calls)

Ejemplo

- Open/close para archivos
- malloc/free para memoria
- request/release para dispositivos (waits y signal)

Existe deadlock cuando cada proceso en el conjunto está esperando por un evento que sólo puede ser causado por otro proceso en el conjunto.

ABRAZO MORTAL - DEADLOCK

Grafo de Recursos.

- G = (V,E) dirigido.

- V dividido en dos tipos
  - -. P : conjunto de procesos en el sistema
  - -. R : conjunto de recursos en el sistema.

- Lados de requerimientos: arcos dirigidos P_i-->R_j

- Lados de asignación: arcos dirigidos R_j-->P_i

ABRAZO MORTAL - DEADLOCK

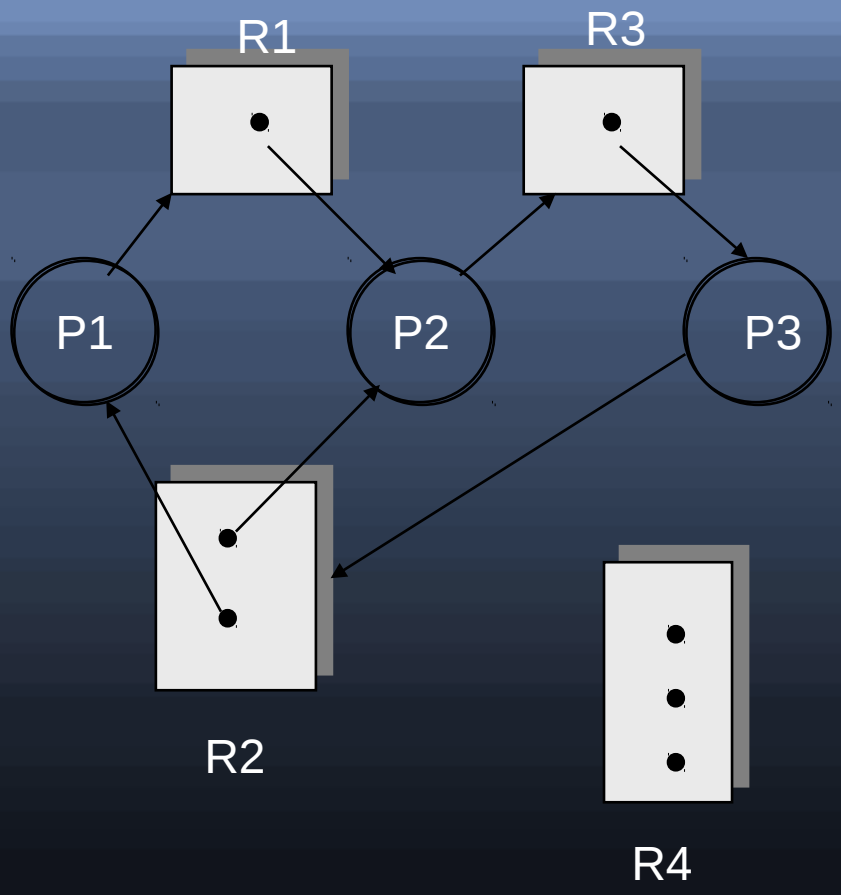
Grafo de Recursos.

- Si el grafo NO contiene ciclos ==> NO deadlock.

- Si el grafo contiene ciclos ==>
  - -. Deadlock si hay una instancia por recurso
  - -. Posibilidad de deadlock si muchas instancias por tipo de recursos

ABRAZO MORTAL - DEADLOCK

Ejemplo.



DEADLOCK

ABRAZO MORTAL - DEADLOCK

Metodos para manejar Deadlock.

- Asegurar que el sistema nunca entra en estado de deadlock.
    -. Prevención
    -. Prediccion

- Permitir que el sistema entre al estado de deadlock y luego recuperarlo.

- Ignorar el problema y pretender que no va a ocurrir. (Sol. De muchos sistemas como UNIX. El sistema se va degradando y tiene que intervenir el administrador. Es un enfoque aceptable.

ABRAZO MORTAL - DEADLOCK

Prevencion.

- Método: trata de garantizar que alguna de las 4 condiciones no ocurra.
- No es posible prevenir por negación de la exclusión mutua, ya que existe recursos intrínsicamente no compartibles. Ej printer
- Se debe garantizar que siempre que un proceso solicite un recurso, el no tiene otro.

ABRAZO MORTAL - DEADLOCK

Prediccion.

- Se le da a los procesos los recursos siempre que sea posible. Con esta información se decide si el proceso debe esperar o no.
- Información requerida
    - -. Recursos disponibles
    - -. Recursos asignados a cada proceso
    - -. Liberaciones y futuras solicitudes de recursos.
- Un algoritmo que evita deadlock examina dinámicamente el estado de los recursos asignados para asegurar que nunca puede haber una condición de espera circular.
- El estado de asignación de recursos es definido por el número de recursos disponibles y asignados mas la demanda máxima de los procesos.

ABRAZO MORTAL - DEADLOCK

Estado Seguro.

- Cuando un proceso solicita recursos, el sistema debe decidir si la inmediata asignación deja al sistema en estado seguro.
- Un sistema está en estado seguro, si existe una secuencia segura de todos los procesos.
- La secuencia <P1,P2,….Pn> es segura si para cada P_i, los recursos que P_i aun necesita pueden ser satisfechos por los recursos disponibles + los que libere P_j para todo j < I.

ABRAZO MORTAL - DEADLOCK

Estado Seguro.

- Si los recursos que P_i necesita no est'an disponibles inmediatamente, entonces P_i puede esperar hasta que P_j haya finalizado.
- Cuando P_j finaliza, P_i puede obtener los recursos, se ejecuta y devuelve los recursos asignados.
- Cuando P_i termina, P_{i+1} puede obtener los recursos que el necesita.

ABRAZO MORTAL - DEADLOCK

Hechos Basicos.

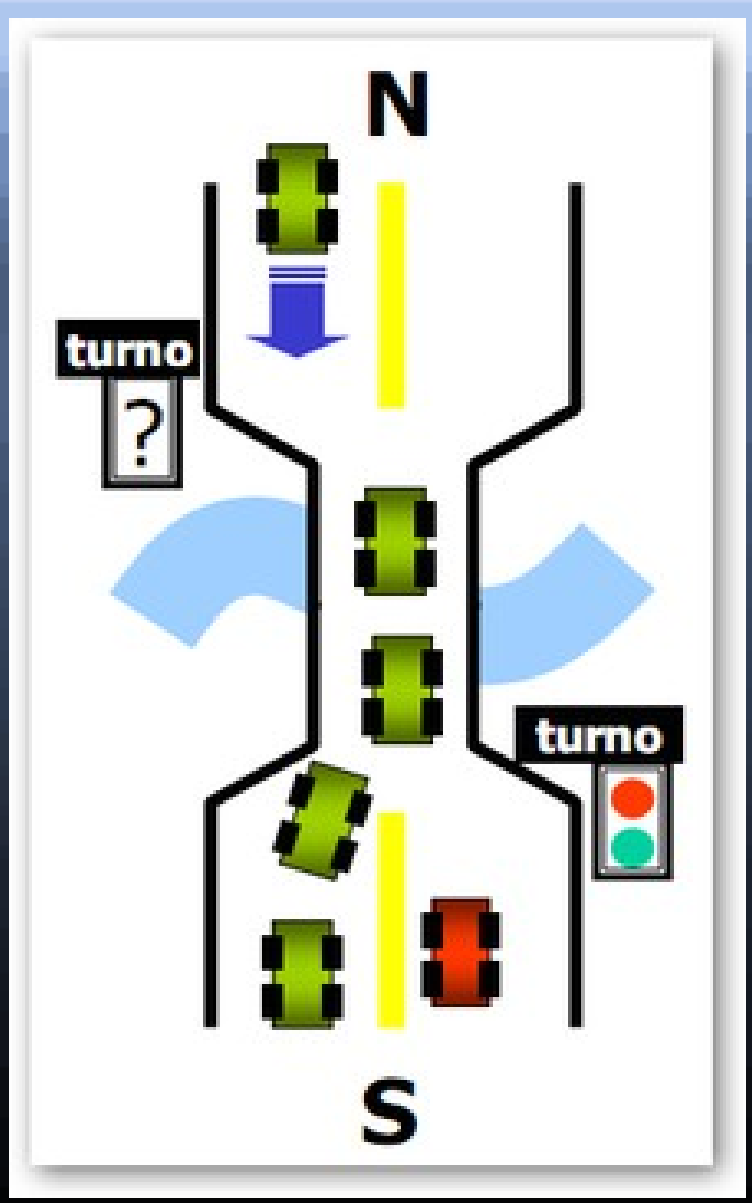
- Si un sistema está en estado seguro ==> NO deadlock
- Si un sistema está en estado inseguro ==> existe posibilidad de deadlock.
- Predicción ==> asegurar que el sistema nunca entre en estado inseguro.

ABRAZO MORTAL - DEADLOCK

Recuperacion.

- Finalización de Procesos
  - -. Abortar todos los procesos que forman parte del deadlock. => costo elevado
  - -. Abortar un proceso a la vez hasta que el ciclo del deadlock sea eliminado. => overhead (se debe llamar muchas veces el algoritmo)
  - -. El orden de elección del proceso a abortar

ABRAZO MORTAL - DEADLOCK



Semaforo

Inventado por Dijkstra (1965), un semaforo es un contador entero (S), el valor del contador puede ser positivo o cero, pero nunca negativo. Se le puede asignar un valor al crear el semaforo. Se puede realizar dos operaciones en un semaforo, llamadas wait(S) y signal(S).

wait(S): Verifica si el valor del semaforo es mayor que cero, si es asi lo decrementa y retorna, si es cero wait() no finaliza por el momento.

signal(S): Si uno o mas procesos estan bloqueados en el semaforo, sin poder finalizar la llamada a wait(S), el sistema elige uno de ellos y se le permite que finalize la llamada a wait(S). Si no hay procesos bloqueados en el semaforo, incrementa el valor del semaforo.

Las operaciones wait(S) y signal(S), son indivisibles (atomicas), para que no ocurran condiciones de carreras.

Puede haber mas de un proceso bloqueado en un semaforo, cuando otro llama a signal(S), a uno de los procesos bloqueados se le deja finalizar wait(S). El mecanismo para elegir cual debe finalizar puede resolverse utilizando una cola de procesos bloqueados, eligiendo el proceso con la prioridad mas alta, o aleatoriamente, etc. Es dependiente de la implementacion.

Semáforos

- Primitiva de Sincronización propuesta por Dijkstra en 1968
  - Como parte de sistema THE
  - Usados para exclusión mutua y planificación
  - De nivel más alto que locks
- Variable atómica manipulada por dos operaciones
  - Wait(semaforo)
    - Decrementa semáforo
    - Bloquea hebra/proceso si el semáforo es menor que cero, sino entonces permite a hebra/proceso continuar
    - Operacion tambien llamada P(semaforo) o down(semaforo)
  - Signal(semáforo)
    - Incrementa semáforo en uno y si hay algún proceso/hebra esperando lo despierta
    - También llamada V(semaforo) o up(semaforo)
  - Valor de semáforo puede ser mayor que 1
    - Inicializado en 1 es como lock.
      - Usado para exclusión mutua
    - Inicializado en N
      - Usado como contador atómico

Problemas con Semáforos

- A pesar que se pueden usar para resolver cualquier problema de sincronización
  - Son variables globales por lo tanto pueden ser accesadas de cualquier hebra directamente
    - no es buena técnica de ingeniería de software
  - No hay conexión entre el semáforo y el recurso para el cual se quiere controlar acceso
  - Usados como mutex (ingreso a sección crítica) y para coordinación (planificación, elección quien tiene acceso al recurso)
  - No se puede controlar su uso, no hay garantía que el programador los use adecuadamente (fácil de cometer errores)