Tema 2. Procesos

Competencias:

- ✓ Comprender con claridad qué es un proceso: estados básicos de un proceso y sucesos que pueden provocar las transiciones entre los estados, estructuras de datos utilizadas para implementar el concepto de proceso, operaciones básicas sobre los procesos.
- ✓ Asimilar los objetivos de los algoritmos de planificación y analizar las ventajas y desventajas de estos y de otros posibles algoritmos que se puedan encontrar.
- ✓ Comprender los motivos por los que los procesos necesitan comunicarse entre sí.
- ✓ Detectar cuando se produce interbloqueo y las condiciones que se deben presentar en un sistema para que pueda aparecer dicho problema.
- ✓ Ubicar el núcleo del sistema operativo dentro de éste, identificando sus funciones y sus componentes.
- ✓ Ser capaz de aplicar un planificador simple de procesos.
- ✓ Tomar un papel más activo en el aprendizaje.
- ✓ Adquirir destrezas para resolver problemas aplicando los métodos de la ciencia y la ingeniería.
- ✓ Conseguir capacidad de abstracción.
- ✓ Ser capaz de enfrentarse a problemas nuevos recurriendo conscientemente a estrategias que han sido útiles en problemas resueltos anteriormente.

Tema 2. Procesos

1. Concepto de proceso.
2. Principios de la Programación concurrente.
    1. Concurrencia. Programación concurrente.
    2. Áreas de comunicación entre procesos.
        1. Exclusión mutua.
        2. Sincronización.
    3. Interbloqueos (Deadlocks).
        1. Concepto de interbloqueo. Inanición de procesos.
        2. Condiciones necesarias para que se produzca interbloqueo.
    4. Propiedades de corrección de los sistemas concurrentes.
3. Estados de un proceso.
4. Representación de los procesos.
5. Operaciones básicas sobre procesos.
6. Planificación de procesos. **(Actividad 2)**
    1. Concepto de planificación. Objetivos.
    2. Planificación apropiativa frente a no apropiativa.
    3. Algoritmos de planificación.
7. El núcleo del Sistema Operativo.
    1. Características del núcleo.
    2. Componentes del núcleo.
        1. El dispatcher.
        2. El controlador de interrupciones.

1. Concepto de proceso (I)

Definiciones:

1. Multiprogramación: permite que más de un programa resida en memoria principal al mismo tiempo.

   Al número de programas almacenados simultáneamente en memoria principal en un momento dado se denomina grado de multiprogramación.

2. Un programa es una secuencia de instrucciones o acciones definidas mediante un lenguaje de programación, y que pueden ser ejecutadas por parte de un procesador.
3. Un proceso es una secuencia de acciones derivadas de la ejecución de una serie de instrucciones definidas a priori. Esto implica que:
   - un proceso puede requerir la ejecución de uno o varios programas; y
   - que un programa puede formar parte de más de un proceso.

1. Concepto de proceso (II)

Definiciones:

4. Un programa (entidad estática, pasiva) siempre es el mismo, pero su ejecución (proceso) (entidad dinámica, activa) puede variar bastante según este el sistema en cada momento.

5. Un proceso se lleva a cabo por la acción de un agente (procesador) que ejecuta el programa asociado. Es decir, un procesador ejecuta un proceso, o un proceso se ejecuta sobre un procesador.

6. El programa o programas asociados a un proceso no tienen porqué estar implementados en software.

2. Principios de la Programación Concurrente (I)

2.1. Concurrencia. Programación Concurrente. (I)

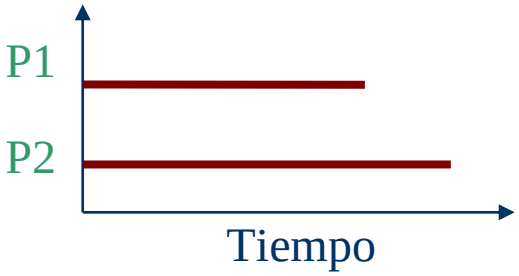
Definiciones de concurrencia:

- ✓ La concurrencia es el solapamiento en el tiempo de la ejecución de varias actividades.
- ✓ Dos procesos, P1 y P2, se ejecutan concurrentemente si la primera instrucción de P1 se ejecuta entre la primera y la última instrucción de P2.

2. Principios de la Programación Concurrente (I)

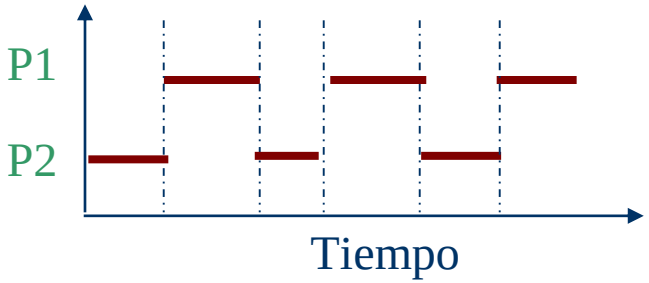
2.1. Concurrencia. Programación Concurrente. (II)

Tipos de concurrencia:



Real o paralelismo

Nº procesadores >= Nº procesos



Aparente, simulada o seudoparalelismo

Nº procesadores < Nº procesos

2. Principios de la Programación Concurrente (I)

2.1. Concurrencia. Programación Concurrente. (III)

Contexto o entorno volátil: es el estado actual del proceso (valor de los registros de la CPU, memoria asignada al proceso, etc.).

Cambio de proceso o conmutación de la CPU: cuando un proceso pierde el control de la CPU y ésta se le asigna a otro proceso. Esta operación requiere guardar el contexto del proceso que pierde el control de la CPU y restaurar el contexto del proceso al que se le asigna la CPU.

Overhead, sobrecarga o carga adicional: tiempo de CPU empleado en la ejecución de los procesos del sistema. Por ejemplo, el tiempo dedicado al cambio de proceso. Durante este tiempo el sistema no realiza ningún trabajo útil (ejecución de programas de usuario).

2. Principios de la Programación Concurrente (II)

2.1. Concurrencia. Programación Concurrente. (IV)

Definición: Procesamiento Concurrente es la situación que se obtiene al hacer una instantánea del sistema, es decir; varios procesos se encuentran en un estado intermedio entre su estado inicial y final.

Definición: Programación Concurrente es el conjunto de *notaciones* que se usan para expresar paralelismo y *técnicas* que se usan para resolver posibles conflictos entre los procesos.

2. Principios de la Programación Concurrente (III)

2.2. Áreas de comunicación entre procesos.

Los procesos de un sistema no actúan de forma aislada. A veces tienen que:

- **Cooperar** para alcanzar un objetivo común ⇒ sincronización
- **Competir** por el uso de recursos ⇒ exclusión mutua

} Tipos de comunicación

2.2.1. Exclusión mutua. (I)

Tipos de recursos:

- **Compartibles**: pueden ser usados por varios procesos de forma concurrente;
- **No compartibles**: su uso está restringido a un solo proceso a la vez, es decir hasta que el proceso que está usando el recurso no compartible no finalice su utilización, éste no puede ser usado por otro proceso. Razones:
  - ✓ por la naturaleza física del recurso;
  - ✓ porque la acción sobre el recurso por parte de un proceso puede interferir en la acción de otro proceso.

2. Principios de la Programación Concurrente (IV)

2.2.1. Exclusión mutua. (II)

Definición: Exclusión Mutua es la comunicación requerida entre dos o más procesos que se están ejecutando en paralelo y que necesitan a la vez el uso de un recurso no compartible.

Consiste en asignar el recurso no compartible a solo uno de los procesos, mientras que los otros deben permanecer a la espera hasta que finalice la utilización de dicho recurso por el proceso al que se le asignó. Cuando este proceso termine, el recurso será asignado a uno de los procesos en espera.

Se asegura el correcto uso del recurso.

Cuando los procesos ejecutan operaciones que no están en conflicto entre sí (uso del mismo recurso no compartible), se les debe permitir que procedan de forma concurrente.

2. Principios de la Programación Concurrente (IV)

2.2.1. Exclusión mutua. (III)

Definición: Sección crítica o región crítica es el trozo de código donde un proceso hace uso de un recurso no compartible, por lo tanto debe ejecutarse en exclusión mutua.

Cuestiones: Sean RNC1 y RNC2 dos recursos no compartibles, y Pa y Pb dos procesos:

*¿Pueden ejecutarse concurrentemente las regiones críticas de los procesos Pa y Pb relacionadas con el mismo recursos no compartible RNC1?

Pa (RNC1) || Pb (RNC1)

* ¿Y si son sobre distintos recursos no compartibles?

Pa (RNC1) || Pb (RNC2)

Las secciones críticas deben ser ejecutadas lo más rápido posible y además deben ser cuidadosamente codificadas.

2. Principios de la Programación Concurrente (V)

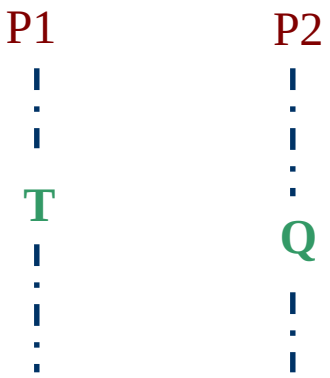
2.2.2. Sincronización.

Definición: Sincronización es la comunicación requerida entre dos o más procesos con el fin de sincronizar sus actividades.

Sincronización simple:

P1 espera por P2

Un proceso, P1, llegado a un punto T, no puede proseguir su ejecución hasta que otro proceso, P2, haya llegado a otro punto, Q, de su ejecución.



Sincronización múltiple:

P1 y P2 se esperan mutuamente

P1 llegado a un punto T de su ejecución, no puede proseguir hasta que P2 haya llegado a otro punto, Q, de su ejecución. Y, P2 llegado a ese punto Q, no puede proseguir hasta que P1 haya llegado a su punto de ejecución T.

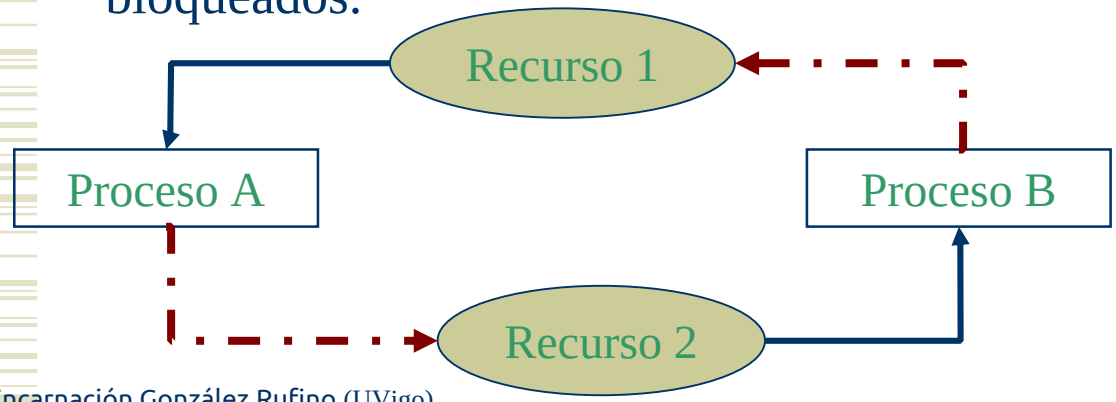
2. Principios de la Programación Concurrente (VI)

2.3. Interbloqueos (Deadlocks).

2.3.1. Concepto de interbloqueo. Inanición de procesos. (I)

Definiciones de interbloqueo:

1. Un proceso esta interbloqueado si está esperando por un evento determinado que nunca va a ocurrir.
2. Un conjunto de procesos se encuentra en estado de interbloqueo, cuando cada uno de ellos está esperando un suceso que sólo puede ser causado por otro proceso del mismo conjunto, y que nunca se producirá porque todos están bloqueados.



Interbloqueo entre dos procesos por el uso de recursos no compartibles (abrazo mortal)

2. Principios de la Programación Concurrente (VI)

2.3.1. Concepto de interbloqueo. Inanición de procesos.(II)

Postergación indefinida, inanición, bloqueo indefinido, espera indefinida, aplazamiento indefinido o Lockout.

Definiciones de postergación indefinida:

1. Un proceso está postergado indefinidamente cuando espera por un evento que puede ocurrir pero no se sabe cuando.
2. Un proceso puede quedar esperando indefinidamente si no puede proseguir su ejecución debido al continuo procesamiento de otros procesos por parte del Sistema Operativo.

2. Principios de la Programación Concurrente (VII)

2.3.2. Condiciones necesarias para que se produzca interbloqueo. (I)

Coffman, Elphick y Shoshani establecieron las **cuatro condiciones necesarias** para que se produzca interbloqueo:

1. Condición de *exclusión mutua*: los procesos reclaman control exclusivo de los recursos que piden.
2. Condición de *esperar por*: los procesos mantienen los recursos que ya les han sido asignados mientras esperan por recursos adicionales.
3. Condición de *no apropiatividad*: los recursos no pueden ser extraídos de los procesos que los tienen hasta su completa utilización.
4. Condición de *espera circular*: existe una cadena circular de procesos en la cual cada uno de ellos mantiene a uno o más recursos que son requeridos por el siguiente proceso de la cadena.

2. Principios de la Programación Concurrente (VIII)

2.3.2. Condiciones necesarias para que se produzca interbloqueo. (II)

Puntos de estudio del interbloqueo:

- **Prevención del interbloqueo:** condiciona al sistema para que elimine toda posibilidad de que se produzca interbloqueo. Estrategias:
  - *Negar la condición de esperar por*: cada proceso debe pedir todos los recursos que va a necesitar de golpe. Si el conjunto de todos ellos está disponible, se le asigna todos. Si no está disponible todo el conjunto completo, no se le asigna ninguno al proceso y tendrá que esperar hasta que estén todos disponibles.
  - *Negar la condición de no apropiatividad*: cuando un proceso que tiene recursos le es negada una petición de recursos adicionales, deberá liberar sus recursos y, si es necesario, pedirlos de nuevo junto con los recursos adicionales.
  - *Negar la condición de espera circular*: cuando se instala un recurso se le asigna un número exclusivo, de forma que los procesos deben de solicitar los recursos en orden ascendente de acuerdo a los números asignados a dichos recursos.

2. Principios de la Programación Concurrente (IX)

2.3.2. Condiciones necesarias para que se produzca interbloqueo. (III)

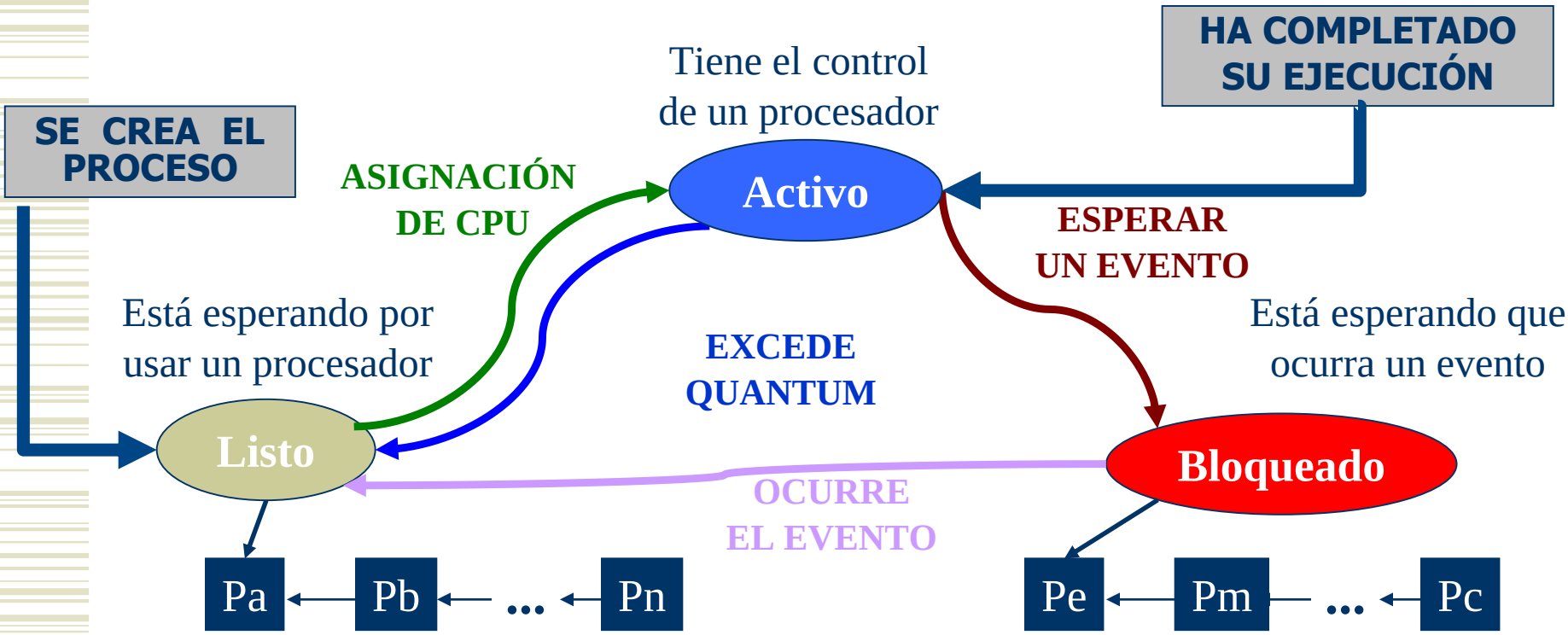
- **Evitación del interbloqueo:** condiciona al sistema para que esquive determinadas situaciones con posibilidad de interbloqueo.
- **Detección del interbloqueo:** determinan si existe o no un interbloqueo, e identifica cuáles son los procesos y recursos implicados en él.
- **Recuperación del interbloqueo:** pretenden romper el interbloqueo retirando una o más de las condiciones necesarias.

2. Principios de la Programación Concurrente (X)

2.4. Propiedades de corrección de los sistemas concurrentes.

- **Propiedades de seguridad (safety):**
  - ✓solución al problema de la *Exclusión Mutua*;
  - ✓solución al problema de la *Sincronización*.
- **Propiedades de vivacidad (liveness):**
  - ✓ausencia de *Deadlock*;
  - ✓ausencia de *Lockout*.

3. Estados de un proceso



4. Representación de los procesos (I)

Bloque de control de proceso (PCB) (Bloque de contexto o descriptor de proceso): estructura de datos que contiene toda la información de un proceso que el S. O. necesita para su control.

Posible contenido:

- estado actual;
- identificación;
- punteros para localizar la memoria que usa;
- área para preservar registros;
- punteros para asignar recursos.

4. Representación de los procesos (II)

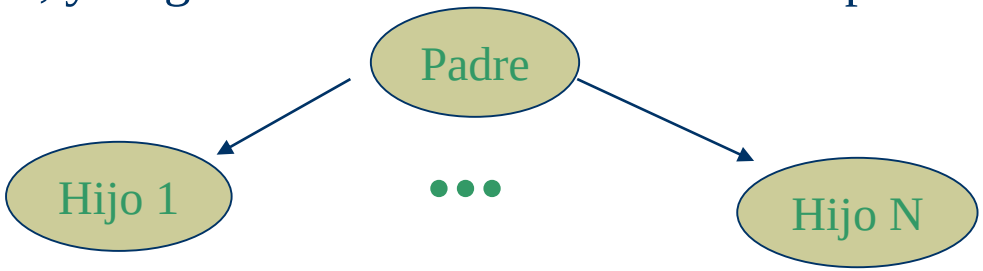
Bloque de Control del Sistema (SCB): estructura de datos que gestiona el Sistema Operativo para controlar todos los PCB's de los procesos. Esta estructura también reside en memoria principal por su alta frecuencia de consulta.

Posible contenido:

- Lista de descriptores (PCB's) de los procesos;
- Puntero al PCB que está activo;
- Puntero a la lista de procesos en estado listo;
- Puntero a la lista de procesos en estado bloqueado.

5. Operaciones básicas sobre procesos

1. **Crear un proceso:** dar nombre al proceso, insertarlo en la lista de procesos en estado listo, determinar su prioridad inicial, crear el PCB del proceso, y asignarle los recursos iniciales del proceso.



2. **Destruir un proceso:** devolver al sistema los recursos que tiene asignado, eliminarlo de todas las listas del sistema y borrar su PCB.
3. **Cambiar la prioridad de un proceso.**
4. **Bloquear un proceso.**
5. **Despertar un proceso.**
6. **Despachar un proceso.**

7. El núcleo del Sistema Operativo (I)

7.1. Características del núcleo.

El kernel se ejecuta con las interrupciones desactivadas y es la parte del S. O. más cercana al hardware, por ello suele estar codificado en ensamblador o lenguaje máquina, además de estar residente en memoria principal. El resto del S. O. se suele programar en lenguajes de alto nivel.

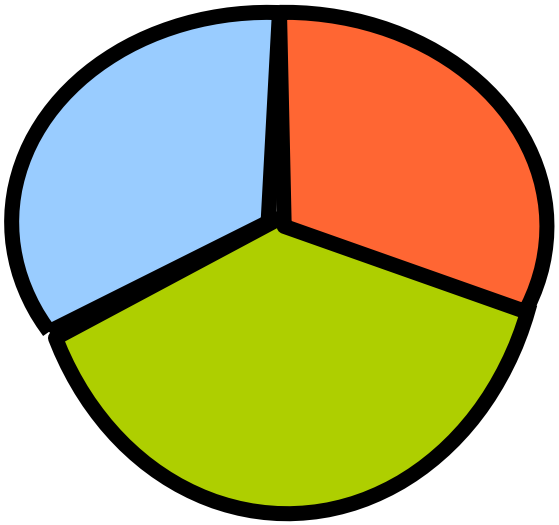
Funciones básicas del kernel:

a) manipulación de interrupciones;
b) inhabilitación y habilitación de interrupciones;
c) creación y destrucción de procesos;
d) cambio de estado de un proceso;
e) despachar un proceso;
f) comunicación entre procesos;
g) manipulación de los PCBs;
h) soporte para servicios de más alto nivel.

7. El núcleo del Sistema Operativo (II)

7.2. Componentes del núcleo.

Dispatcher



Controlador de interrupciones

Manejador de la comunicación entre procesos

7. El núcleo del Sistema Operativo (II)

7.2.1. El dispatcher. (I)

Se encarga de asignar los procesadores a los diferentes procesos, por lo tanto debe actuar cuando se debe comprobar si es necesario cambiar el proceso que está activo, es decir en las siguientes situaciones:

a) Cuando el proceso que está activo se bloquea;

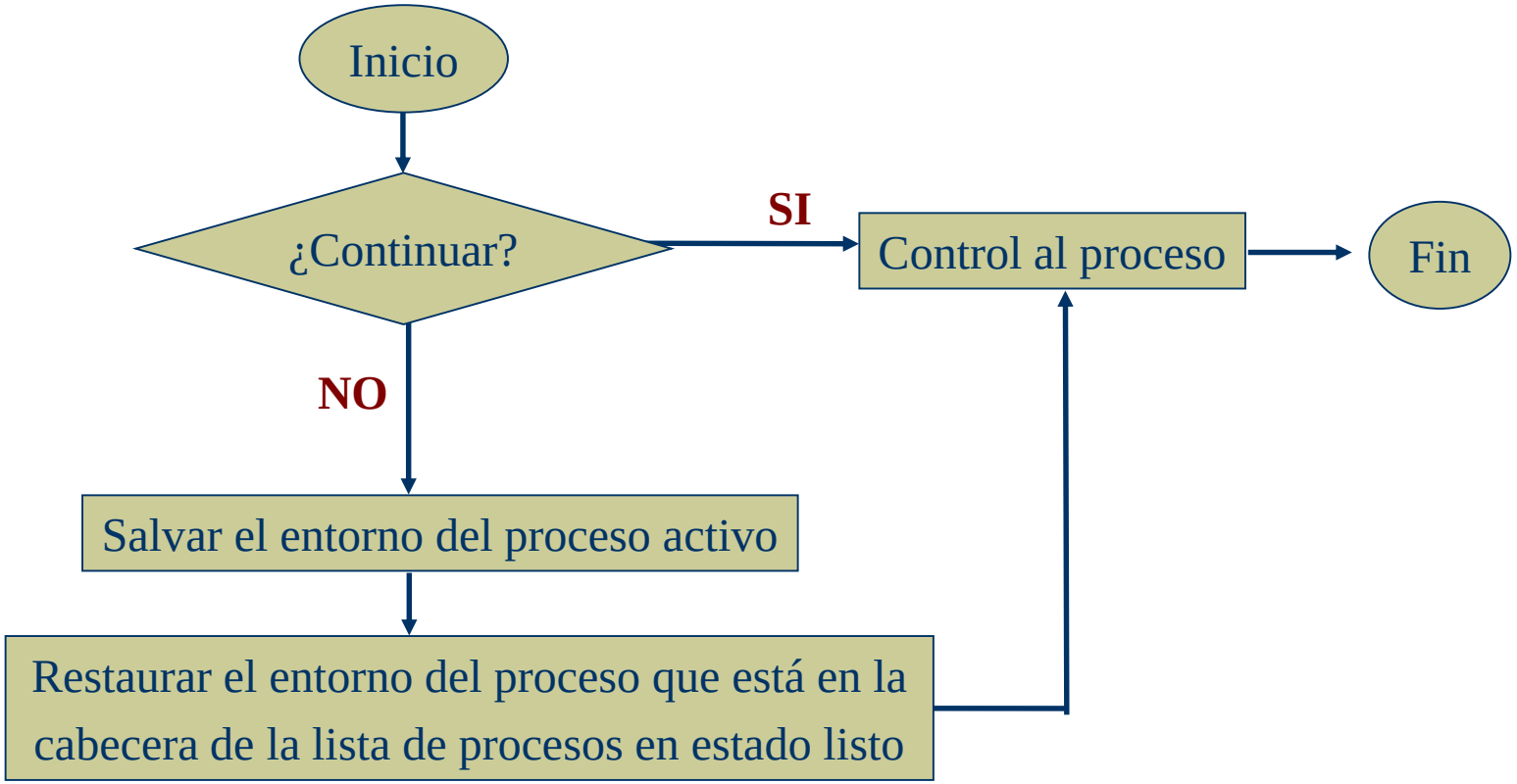
b) Cuando un proceso se desbloquea pasando a estado listo y debido al algoritmo de planificación este puede desplazar al que está activo;

c) Cuando un proceso debe pasar de activo a listo y por lo tanto otro pasara de listo a activo.

7. El núcleo del Sistema Operativo (III)

7.2.1. El dispatcher. (II)

Funcionamiento:



7. El núcleo del Sistema Operativo (IV)

7.2.2. El controlador de interrupciones. (I)

Tipos de interrupciones:

- *Interrupciones internas*: producidas por los propios procesos;
- *Interrupciones externas*: producidas por elementos ajenos a los procesos.

7. El núcleo del Sistema Operativo (IV)

7.2.2. El controlador de interrupciones. (II)

Objetivos del controlador de interrupciones:

1. Determinar el origen de la interrupción.

   Métodos:

   - ejecutando un trozo de código formado por estructuras de decisión anidadas (*cadena de saltos*);
   - que el hardware sea capaz de distinguir las diferentes fuentes de interrupción y transferir el control a su correspondiente rutina de tratamiento;
   - primero el hardware reconoce a qué grupo pertenece la interrupción y luego mediante una pequeña cadena de salto asociada a ese grupo se identifica la interrupción.

7. El núcleo del Sistema Operativo (V)

7.2.2. El controlador de interrupciones. (III)

2.Dar servicio a la interrupción.

- El controlador inicia el servicio de la interrupción invocando a la rutina de tratamiento adecuada. Estas rutinas de tratamiento deben ser lo más cortas posibles, por lo tanto, en general, llevan a cabo acciones mínimas.
- Es posible que una interrupción varíe el estado de un proceso, en cuyo caso será la propia rutina de tratamiento la que cambie dicho estado accediendo al PCB del proceso.

7. El núcleo del Sistema Operativo (V)

7.2.2. El controlador de interrupciones. (IV)

Mientras que actúa el controlador de interrupciones estas están inhabilitadas.

Sin embargo, esto no se puede realizar en sistemas donde determinadas interrupciones requieren una respuesta rápida. Por ello, surge la noción de *prioridad* entre las interrupciones, de forma que una interrupción puede ser interrumpida si llega otra con una prioridad más alta.

En definitiva, solo se inhabilitan las interrupciones de igual o menor prioridad.