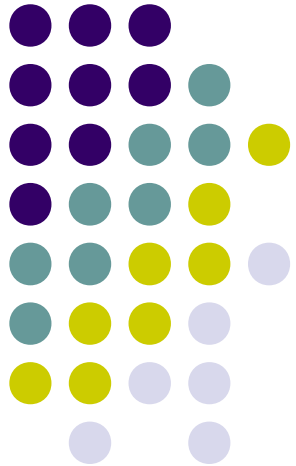
Tema 2

Gestión de Procesos



Índice

1. Introducción
2. Concepto de proceso
3. Estados de un proceso
4. Transiciones de estado de los procesos
5. El Bloque de Control de Procesos (PCB)
6. Operaciones sobre procesos
7. Procesos e hilos
8. Interrupciones
9. Planificación del procesador
10. Comunicación y sincronización entre procesos

1. Introducción

1.1 Registros del procesador

- La arquitectura de registros depende de la máquina.
- Ofrecen el nivel de memoria más rápido y pequeño.
- Se dividen en dos grandes grupos:
  - **Registros visibles por el usuario**: están disponibles para el programador (el compilador los puede usar para optimizar).
  - **Registros de control y estado**: para la ejecución de instrucciones son esenciales.

1. Introducción

1.1 Registros del procesador

- Registros **visibles por el usuario**. Se clasifican en:
  - **Registros de datos** (AX, BX,...): son de propósito general con restricciones (p. ej. sólo para operaciones en coma flotante).
  - **Registros de dirección**: contienen direcciones de memoria de datos e instrucciones. :
    - Registro Índice (SI,DI): para direccionamiento indexado (base + desplazamiento).
    - Puntero de segmento (CS,DS): para direccionamiento segmentado. La memoria está dividida en segmentos.
    - Puntero de pila (SP): contiene la dirección de la cima de la pila.
  - **Registro de código de condición** (FLAGS): bits activados como resultado de determinadas operaciones. Se agrupan en uno o más registros.

1. Introducción

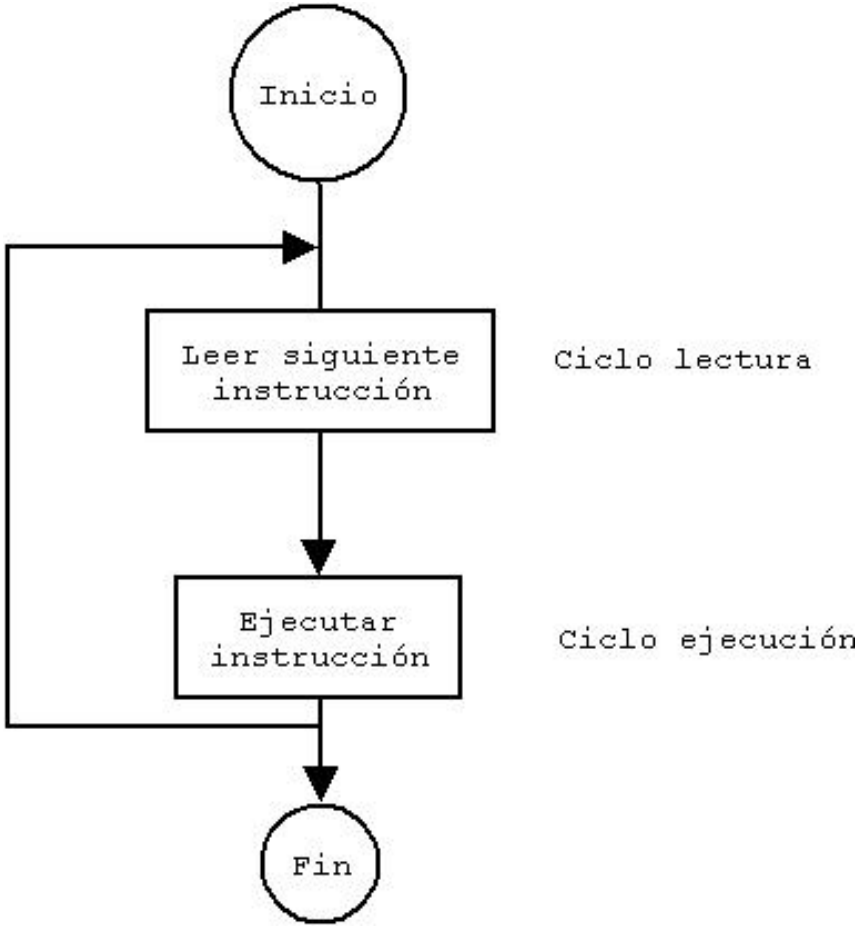
1.1 Registros del procesador

- Registros **de control y estado**
  - **Contador de programa** (PC): la dirección de la próxima instrucción a leer.
  - **Registro de Instrucción** (IR): la última instrucción leída.
  - **Palabra de estado de los elementos del sistema** (PSW): es un conjunto de registros que contiene información de estado (códigos de condición y otra información). Entre otros: bits de signo, bits de cero, bits de acarreo, igualdad, desbordamiento, habilitar/deshabilitar interrupción, supervisor, ...
  - **Otros registros de estado**: registros con punteros a rutinas de tratamiento de interrupción, dirección de memoria con más información de estado, etc.

1. Introducción

1.2 Ejecución de instrucciones

- Vamos a considerar (de forma muy sencilla) que el procesamiento de instrucciones consta de dos pasos:
  1. Trae la instrucción desde memoria, en IR se almacena el contenido indicado por PC.
  2. Ejecución de la instrucción.
- Estos pasos se repiten hasta acabar el programa:



CICLO DE INSTRUCCIÓN

1. Introducción

1.2 Ejecución de instrucciones

- El procesador interpreta la instrucción y realiza la acción, que puede ser de las siguientes **categorías**:
  - **Procesador / memoria**: transfiere datos de procesador a memoria y viceversa.
  - **Procesador- E/S**: transfiere datos de o desde procesador a módulos de E/S (dispositivos).
  - **Tratamiento de datos**: el procesador realiza alguna operación aritmética o lógica de los datos.
  - **Control**: se altera la secuencia de ejecución (saltos, saltos condicionales ...). Esto se realiza ajustando el valor de PC.
  - Otras instrucciones.

1. Introducción

1.3 Niveles de ejecución

- Existen dos **niveles de ejecución** del procesador:
  - **Nivel de usuario**: en este modo se ejecutan los programas de usuario, tiene ciertas restricciones (subconjunto de instrucciones y registros, prohibidas E/S directamente, etc.).
  - **Nivel del núcleo** (modo **supervisor**): es el nivel en el que se ejecuta el núcleo y no tiene restricciones.
- Por tanto la computadora presenta dos modelos de programación. Uno más restrictivo y otro más permisivo.
- El cambio del nivel de ejecución se realiza modificando uno o varios bits del registro de estado.

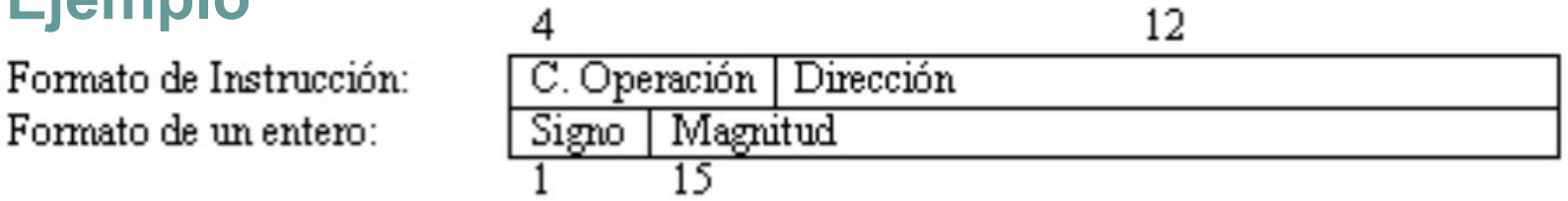
1. Introducción

1.4 Interrupción de reloj

- Es una **interrupción que se produce cada cierto intervalo de tiempo** (20 ms) y provoca que el S.O tome el control mediante la R.S.I (Rutina de Servicio de Interrupción) del reloj.
- Hay que diferenciar entre el reloj hardware y la interrupción de reloj
  - Microprocesador 100 Mhz → 100 Millones de pasos por segundo
  - Si suponemos que una instrucción consume 100 pasos
  - Si se produce interrupción cada 0,02 seg. → 2 millones de pasos entre cada interrupción de reloj → 20 mil instrucciones
- Las R.S.I. forman parte del S.O.

1. Introducción

1.5 Ejemplo



Registros de la C.P.U:

- Contador de programa: PC
- Registro de Instrucción: IR
- Acumulador: AC

Lista parcial de códigos de operación:

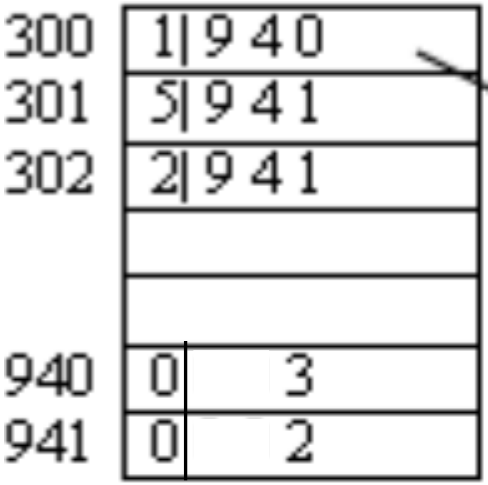
- 0001 --> Cargar de la memoria al AC
- 0010 --> Almacenar de AC a memoria
- 0101 --> Sumar al AC el contenido de la memoria

Los datos e instrucciones son de 16 bits.

¿Cuántos códigos hay?

¿Cuántas posiciones de memoria?

Inicialmente el PC contiene 300 y la memoria está de la siguiente forma:



2. Concepto de proceso

- Surge con la necesidad de que el computador esté ejecutando varios programas al “mismo tiempo”(**concurrentemente**).
- **¿Qué es un proceso?**
  - Un proceso es un **programa en ejecución** que comprende el valor actual del contador de programa, de los registros y de las variables, siendo además la unidad de propiedad de los recursos y la unidad de expedición.
  - También podemos decir que un proceso es una **instancia** de un programa en ejecución, por tanto es una entidad dinámica, al contrario del concepto de programa que es una entidad estática.
- Normalmente el procesador está conmutando de un proceso a otro, pero **el sistema es mucho más fácil de entender si lo consideramos como un conjunto de procesos que se ejecutan quasi en paralelo**, olvidándose de cómo el procesador pasa de un proceso a otro.

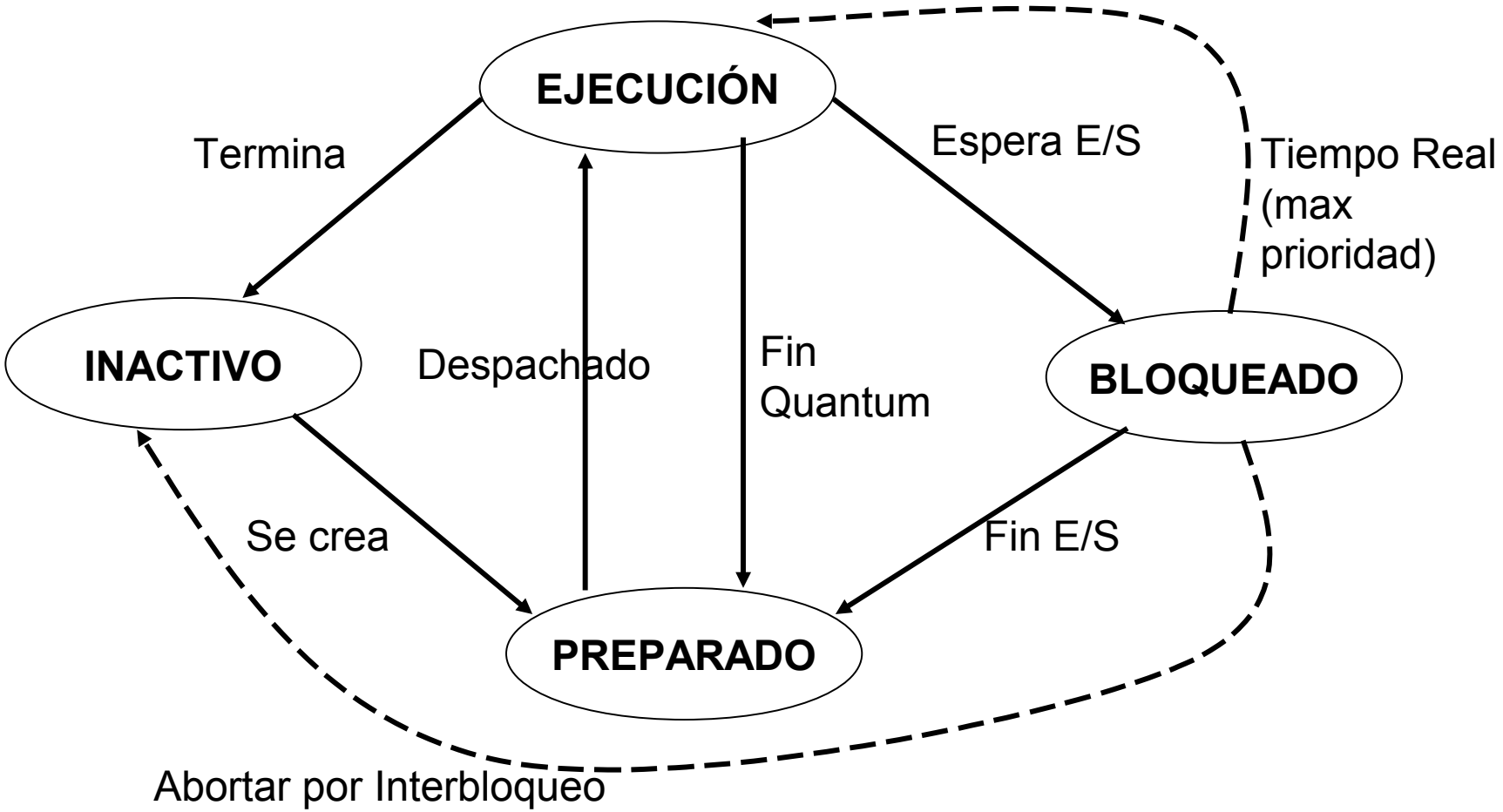
3. Estados de un proceso

- Nosotros vamos a considerar que los procesos se ejecutan en un S.O **multiusuario y multitarea**.
- La **multitarea** se basa en tres **características** diferentes:
  - Paralelismo real entre E/S y procesador
  - Alternancia en los procesos de fases de E/S y de procesamiento
  - Memoria principal capaz de almacenar varios procesos
- Para poder realizar pues una multitarea con esas características necesitamos definir los posibles **estados** en los que puede estar un proceso

3. Estados de un proceso

- Un proceso puede pasar por los siguientes **estados**:
  - **Inactivo**. Consideraremos así a los programas, es decir, al proceso que aún no ha sido ejecutado, y por tanto no ha sido creado
  - **Preparado**. Proceso activo que posee todos los recursos necesarios para ejecutarse, pero al que le falta, precisamente, el procesador.
  - **Ejecución**. Proceso que está siendo ejecutado por el procesador. Con un solo procesador sólo puede haber un proceso en ejecución.
  - **Bloqueado**. Proceso que además de no contar con el procesador requiere algún otro recurso del sistema.
- El conjunto de procesos activos está cambiando de estado.
- Los estados de todos los procesos activos y los recursos (tanto ocupados como libres) del sistema se considera el **ESTADO GLOBAL DEL SISTEMA.**

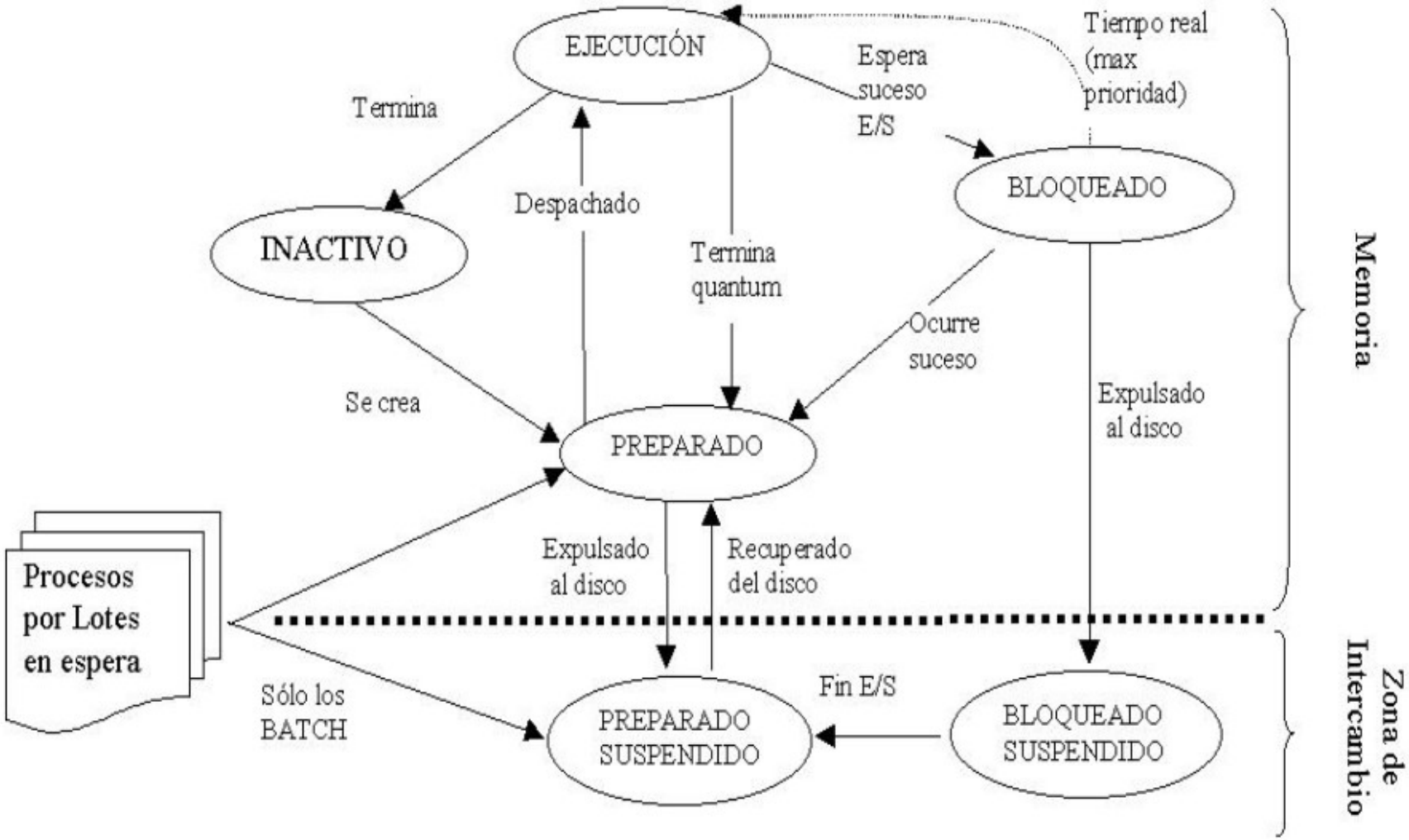
4. Transiciones de estado de los procesos



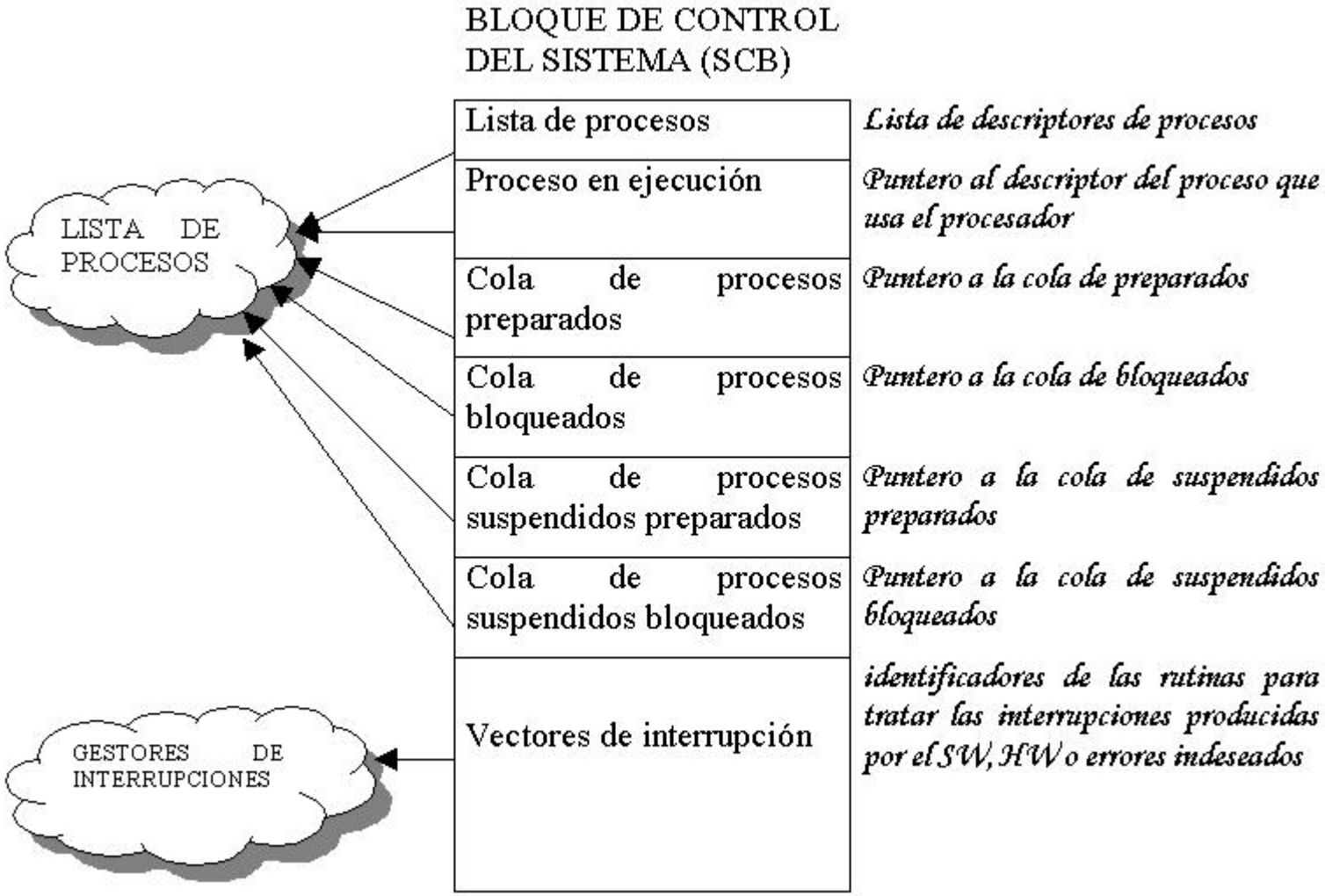
Proceso Nulo

4. Transiciones de estado de los procesos

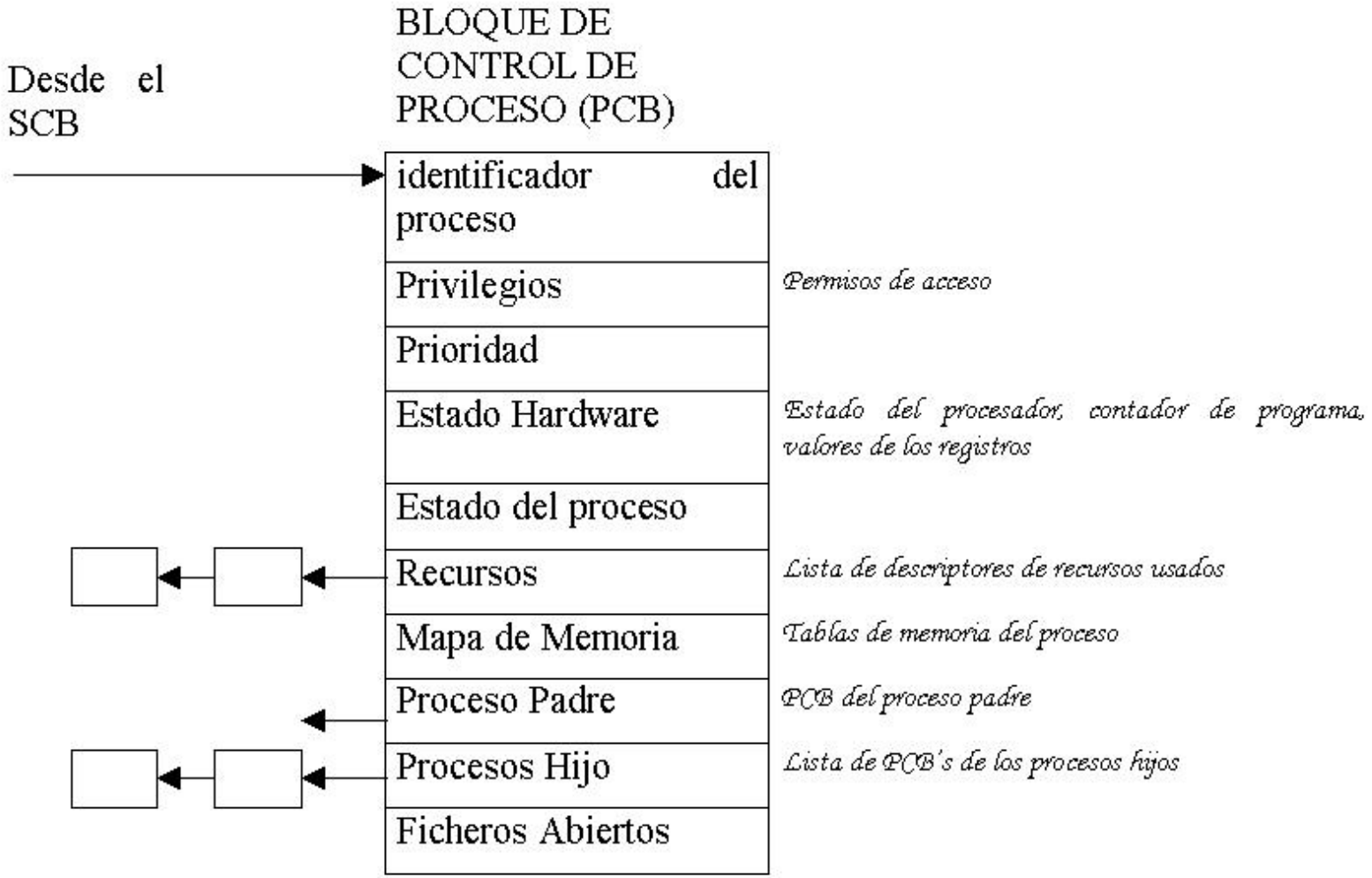
4.1 Estados suspendidos



5. El Bloque de Control de Procesos (PCB)



5. El Bloque de Control de Procesos (PCB)



5. El Bloque de Control de Procesos (PCB)

- La información de cada proceso se encuentra 'parcialmente' en el PCB y parcialmente fuera de él (como vemos en la figura). La decisión de donde colocarla dependerá de:
  - **Eficiencia**. La tabla de PCBs suele ser una estructura estática, donde los PCBs tienen un tamaño fijo. Así pues la información que puede crecer debe estar fuera del PCB.
  - **Compartir información**. Cuando la información es compartida por varios procesos, esta no se incluye en el PCB. Un ejemplo son los punteros de posición de los archivos abiertos.

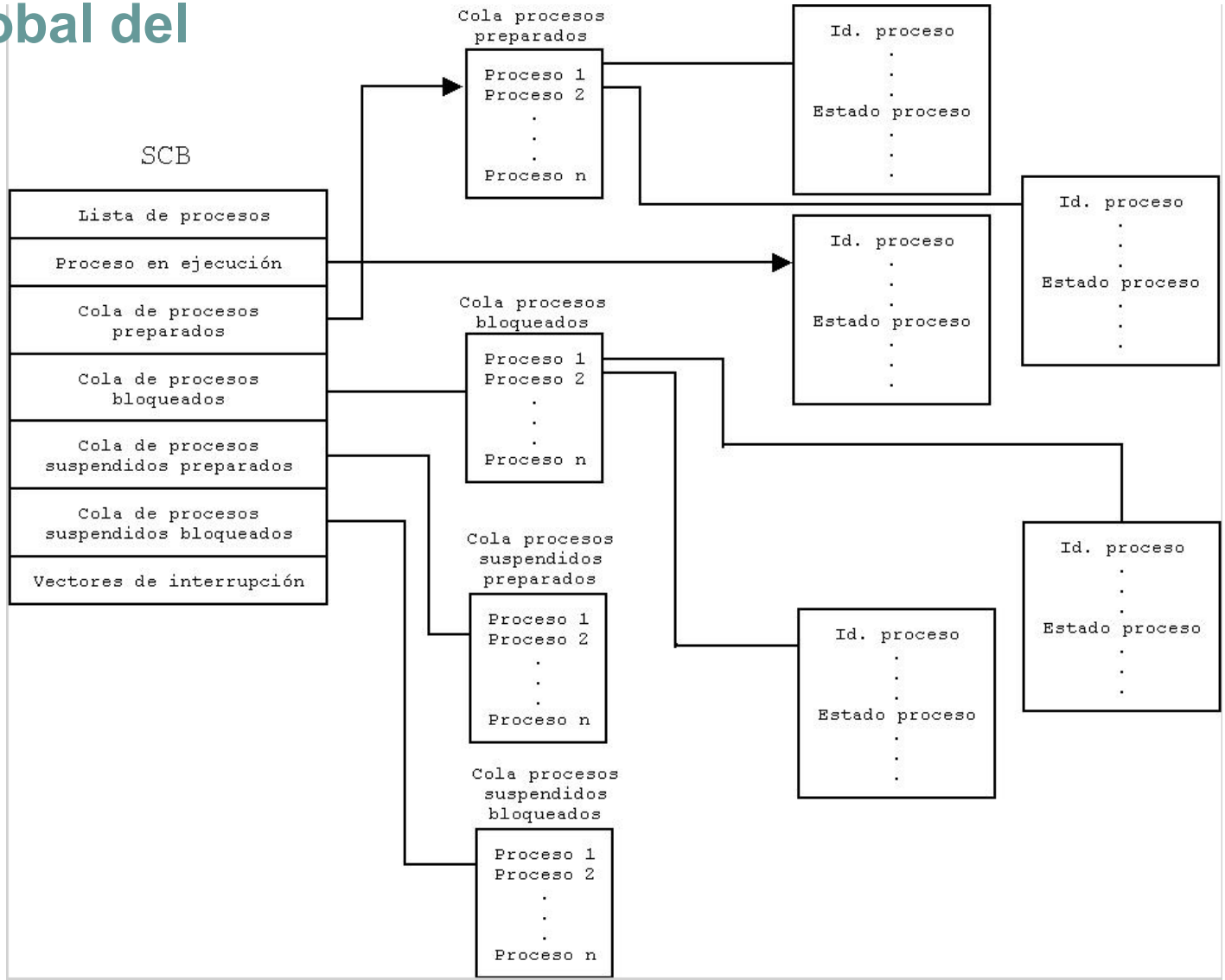
Objetivos

- Que el S.O pueda **localizar fácilmente la información**
- **Preservar** los datos del proceso en caso de que tenga que ser desalojado temporalmente de su ejecución.

Prioridades: Normalmente todos los procesos no tienen las mismas exigencias de tiempo en su ejecución.

5. El Bloque de Control de Procesos (PCB)

Estado Global del Sistema

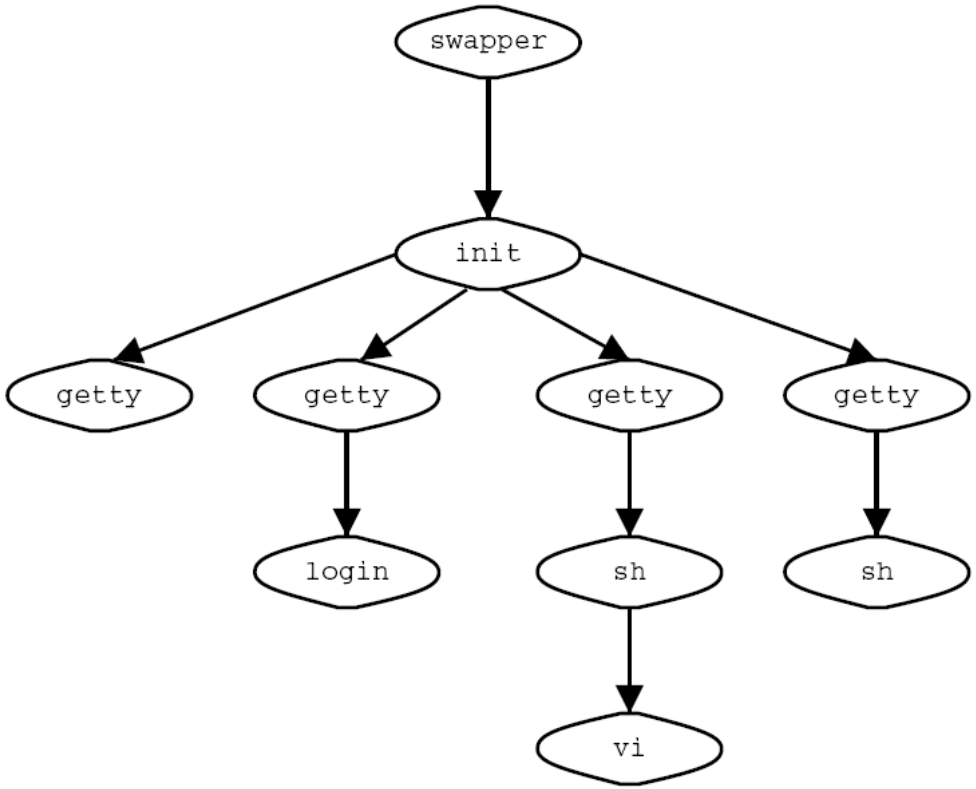


6. Operaciones sobre procesos

- **Crear**: como argumentos requiere argumentos como el nombre y la prioridad.
- ¿Que **acciones** se realizan?
  - Dar un identificador al proceso.
  - Buscar un hueco en la lista de *PCBs* del sistema.
  - Crear el *PCB* del proceso con toda la información necesaria.
  - Asignarle los recursos que necesite.
  - Insertar el proceso en la lista de procesos preparados.

6. Operaciones sobre procesos

- La creación de procesos puede ser:
  - **Jerárquica**



  - **No Jerárquica**

6. Operaciones sobre procesos

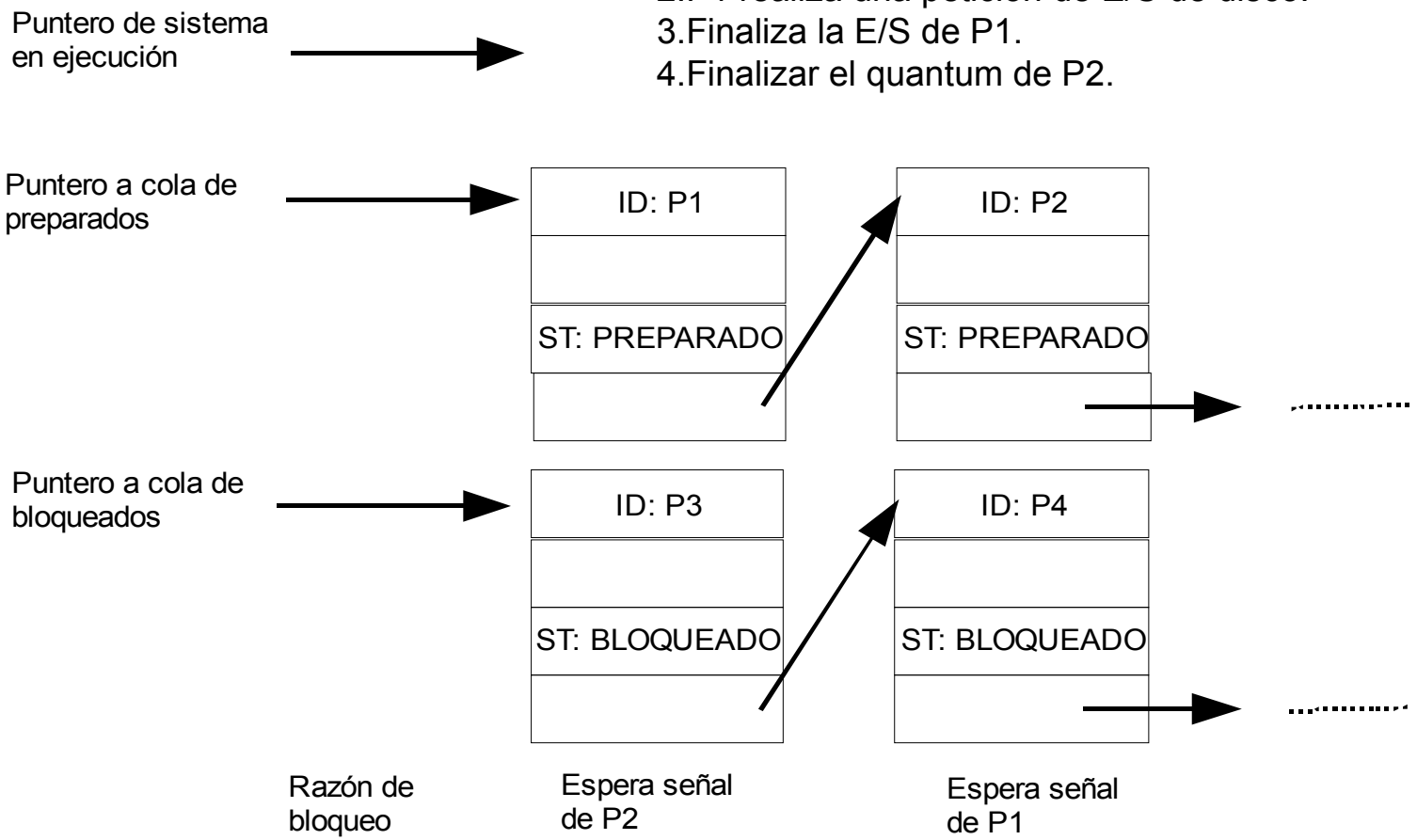
- **Finalizar**: se destruye su PCB y se libera el hueco de la lista de PCB's del sistema. En una jerarquía de proceso hay que decidir si se destruye a los hijos al destruir el proceso o no.
- **Despachar un proceso**: poner en ejecución a uno de los procesos que hay en la cola de preparados.
- **Bloquear un proceso**: Llevar a un proceso de ejecución a la cola de bloqueados.
- **Desbloquear un proceso** : ocurre el evento que esperaba el proceso, éste pasará a la cola de procesos bloqueados a preparados.
- **Suspender un proceso**: se lleva al proceso a la cola de procesos suspendidos preparados o suspendidos bloqueados, según sea su estado.
- **Recuperar un proceso suspendido**. Se lleva el proceso desde la cola de suspendidos a preparados.
- **Cambiar la prioridad**. Modifica la prioridad del proceso.

6. Operaciones sobre procesos

Ejemplo

Reflejar las siguientes operaciones:

1. Despachar el proceso P1.
2. P1 realiza una petición de E/S de disco.
3. Finaliza la E/S de P1.
4. Finalizar el quantum de P2.



7. Procesos e hilos

- Todo proceso se pueden ver como:
  - Unidad de propiedad de los recursos (los recursos se asignan al proceso).
  - Unidad de expedición: el proceso es unidad de ejecución y planificación.
- Estas características se pueden diferenciar para que sean tratadas de forma independiente por el S.O.:
  - Unidad de propiedad de los recursos: proceso o tarea.
  - Unidad de expedición: Hilo.
- A los hilos también se le llaman **procesos ligeros, hebras, thread, subprocesos, …**
- Un hilo es un programa en ejecución (flujo de ejecución), que comparte la imagen de memoria y otras informaciones con otros hilos.

7. Procesos e hilos

- **Utilidad**: puede existir más de un hilo dentro del mismo proceso ⇒ mejora concurrencia y compartición de recursos. (el cambio de contexto entre hebras es mínimo).
- Las hebras de un proceso no son tan independientes (no hay protección) como los procesos distintos ya que:
  - comparten espacio de direcciones ⇒ uso de mecanismos de sincronización.
  - comparten archivos, procesos hijos, recursos E/S, etc.

Elementos por hebra
Contador del programa Pila Registros Hilos hijos Estado

Elementos por proceso
Espacio de direcciones Variables globales Archivos abiertos Procesos hijos Cronómetros Señales Semáforos Estadísticas

7. Procesos e hilos

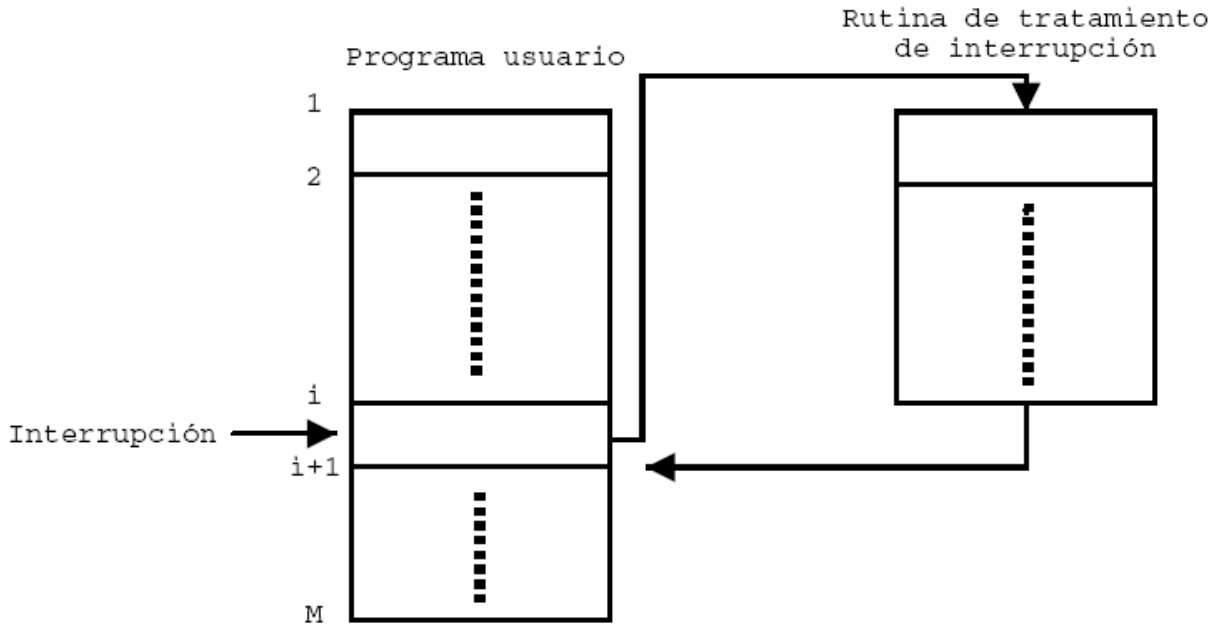
- **Ventajas**:
  - Es más rápido crear un hilo en un proceso existente que crear un proceso nuevo.
  - Es más rápido terminar el hilo que el proceso.
  - Es más rápido cambiar entre dos hilos de un mismo proceso.
- La comunicación entre hebras se hace a través de la **memoria compartida.**
- Cada hilo sólo pertenece a un proceso y no puede existir fuera de él.

7. Procesos e hilos

- Los **estados** de los **hilos** serán:
  - preparados.
  - bloqueados.
  - en ejecución.
- El **estado** del **proceso** será:
  - En ejecución, si hay al menos un hilo en ejecución.
  - Preparado, si hay al menos un hilo preparado y ninguno en ejecución.
  - Bloqueado, si todos los hilos lo están.
- Con los hilos no tiene sentido hablar de estados suspendidos, puesto que si el proceso está suspendido, todos sus hilos lo estarán.
- A partir de Windows NT, los hilos constituyen la unidad ejecutable en los sistemas Windows.

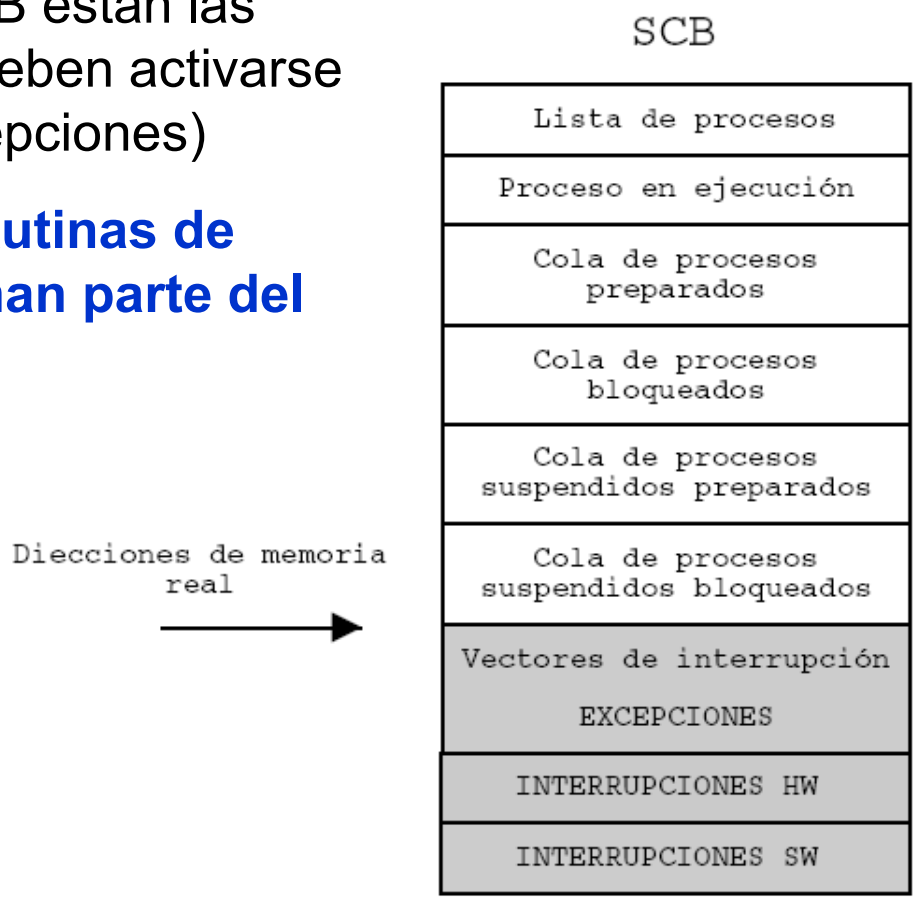
8. Interrupciones

- Una interrupción es la **presencia de un evento o señal que obliga al S.O a tomar el control del procesador para estudiarla y tratarla**. Normalmente esto ocurre por dos tipos de eventos: las interrupciones y las excepciones.
- De cara al **usuario** es una interrupción de la secuencia de ejecución, **no necesita añadir ningún código especial:**



8. Interrupciones

- Para poder atenderlas en el SCB están las direcciones de las rutinas que deben activarse (vector de interrupciones o excepciones)
- Estas rutinas, llamadas **R.S.I (Rutinas de Servicio de Interrupción) forman parte del S.O.**



8. Interrupciones

8.1 Utilidades

- En multiprogramación permiten al **S.O tomar el control** ante un error del HW o del programa en ejecución.
- Permiten **simultanear** el uso de la **E/S** y el **procesador** (el dispositivo avisa con una interrupción).
- Permite reparto de tiempos (**tiempo compartido**).
- Permiten la posibilidad de reconocer **eventos externos** que deba controlar el sistema.

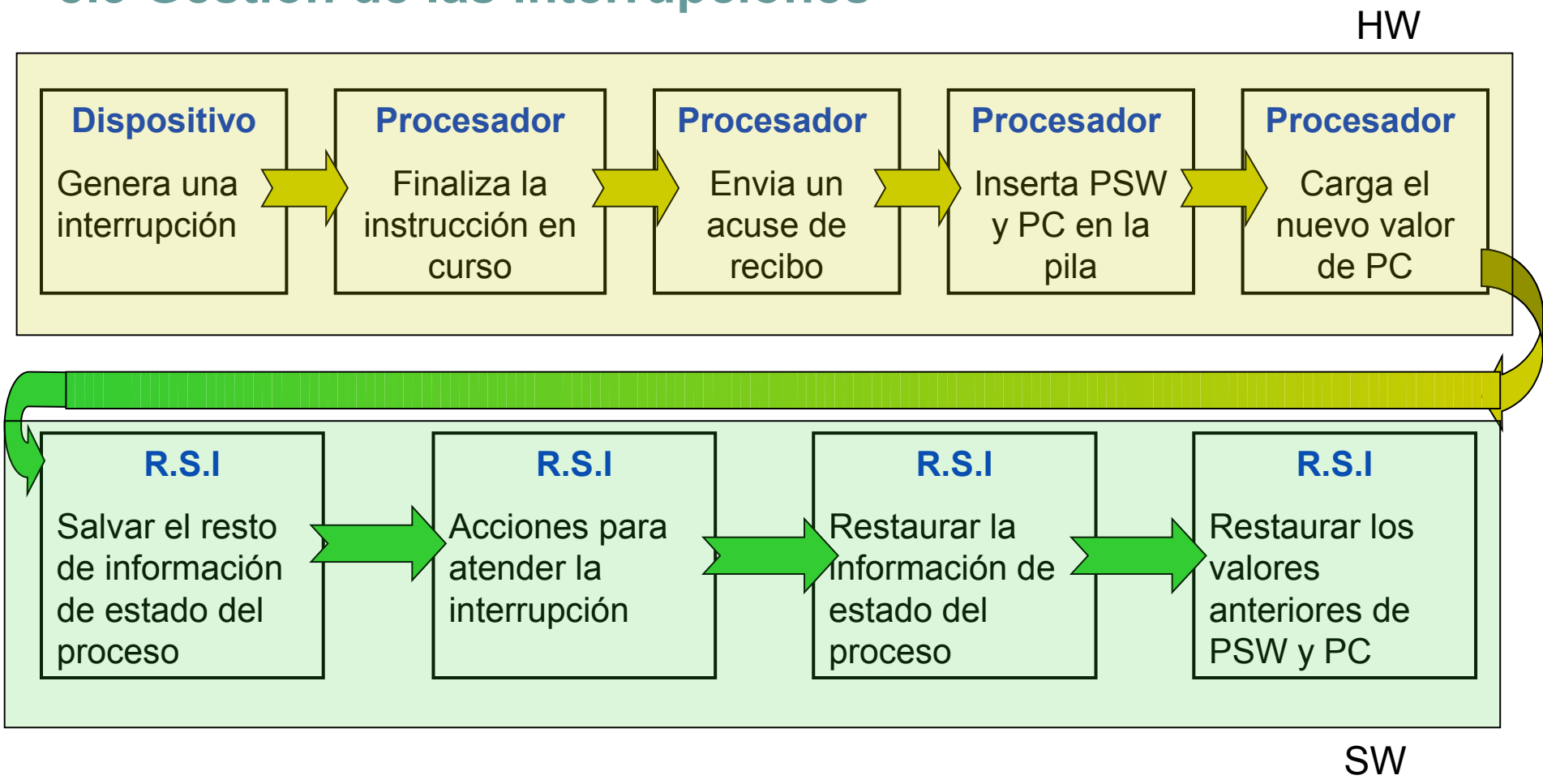
8. Interrupciones

8.2 Tipos de interrupciones

- **Interrupciones HW**. Las interrupciones hardware ocurren cuando un dispositivo necesita atención del procesador y genera una señal eléctrica en la línea IRQ que tiene asignada.
- **Interrupciones SW**: Los procesadores Intel de la gama x86 y compatibles, disponen de una instrucción INT que permite generar por software cualquiera de los tipos de interrupción hardware. El proceso seguido es exactamente el mismo que si se recibe la interrupción hardware correspondiente.
- **Excepciones**: Durante el funcionamiento del procesador pueden ocurrir circunstancias excepcionales; es usual citar como ejemplo el caso de una división por cero. En estos casos, el procesador genera una excepción, que es tratada como si fuese una interrupción, con la diferencia de que el número de interrupción asociado depende del tipo de excepción.

8. Interrupciones

8.3 Gestión de las interrupciones

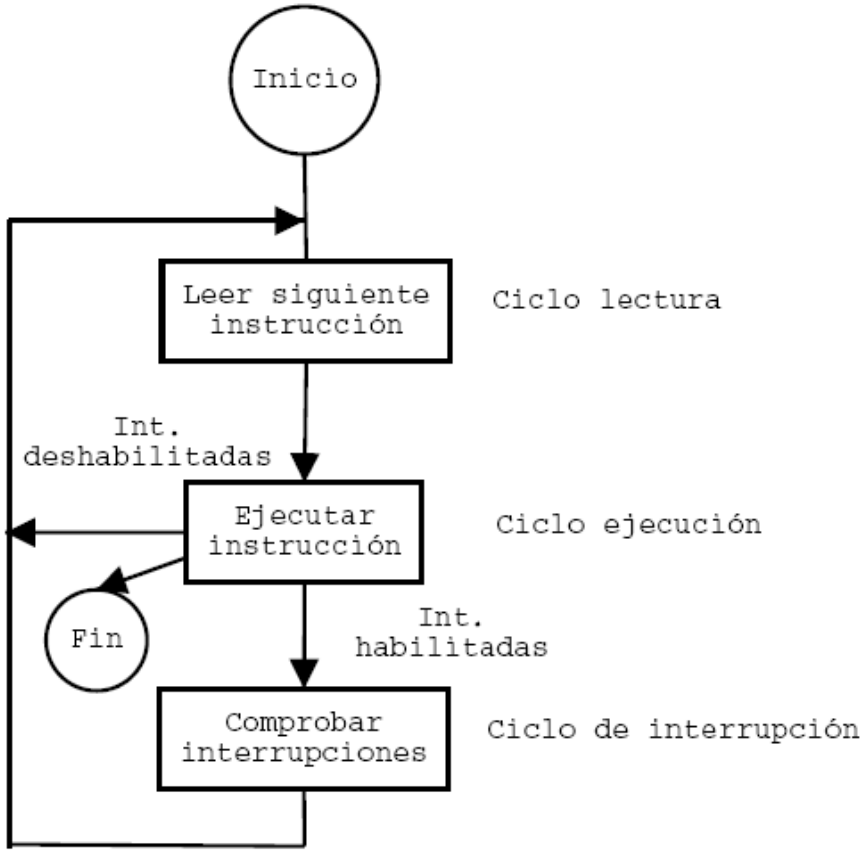


Como vemos el programa de usuario no dispone de ningún código especial

8. Interrupciones

8.3 Gestión de las interrupciones

- Para poder llevar a cabo estas operaciones el ciclo de instrucción debe poder reconocer las interrupciones



El nuevo ciclo añade una pequeña sobrecarga de proceso en comparación con el tiempo que se gana al usar interrupciones.

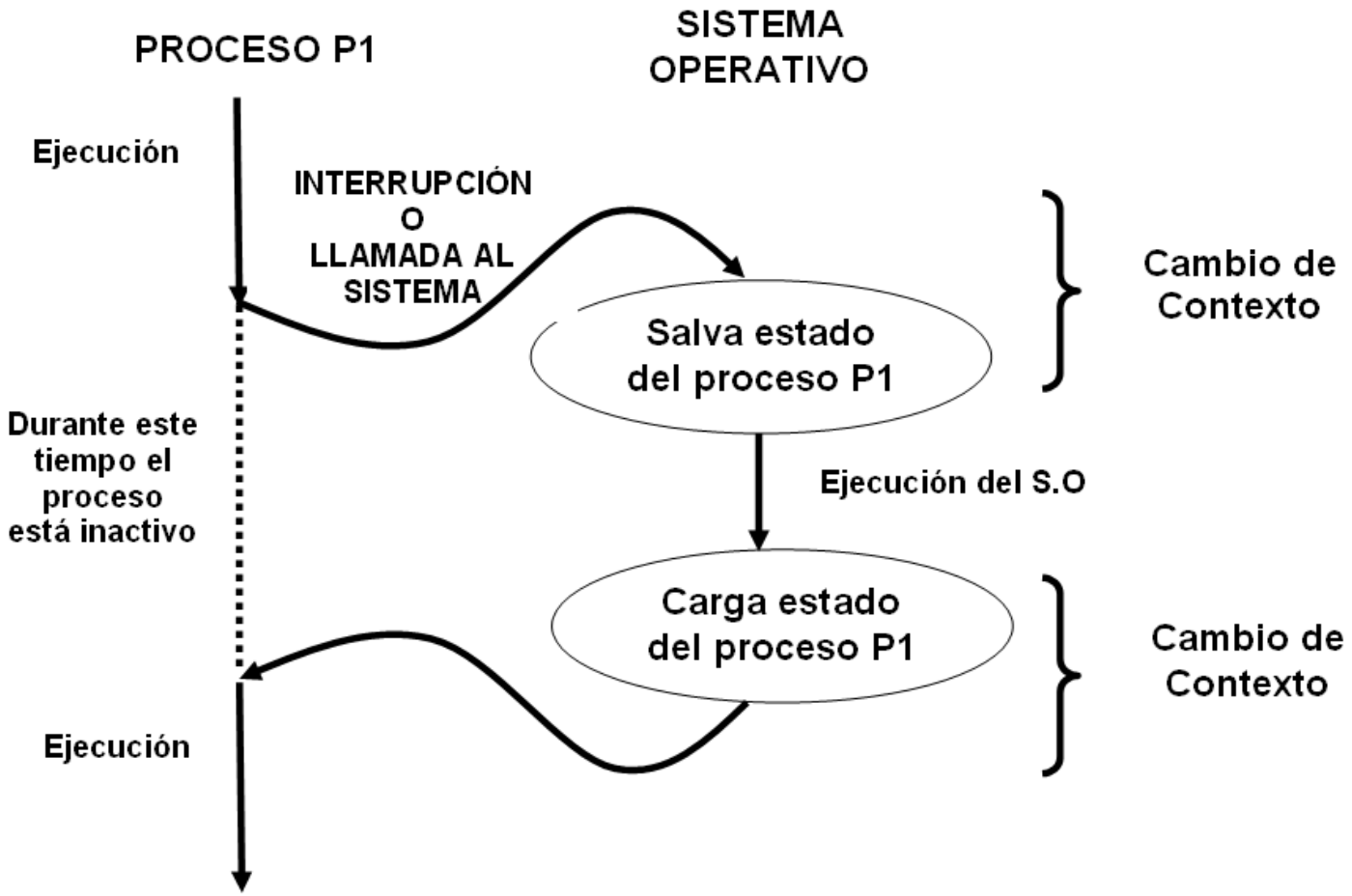
8. Interrupciones

8.4 Cambios de contexto

- **Contexto:** toda aquella información que puede verse alterada, por la ejecución de una rutina de interrupción, y que será necesaria para continuar el proceso que ha sido interrumpido (registros del procesador, zonas de memoria, etc.).
- Un **cambio de contexto** ocurre, por tanto, cuando la información referente al proceso en curso debe ser almacenada, para dar paso a la ejecución de otra rutina o proceso, que utiliza su propia información. Normalmente, esto lleva asociado además un cambio en el modo de trabajo del procesador.
- Los cambios de contexto se deben a dos **causas**:
  1. El proceso que está en ejecución produce una **llamada al sistema operativo**.
  2. Llega una **interrupción**.
- En ambos casos, la información del proceso en curso debe ser almacenada, para dar lugar a la ejecución de una rutina del sistema operativo

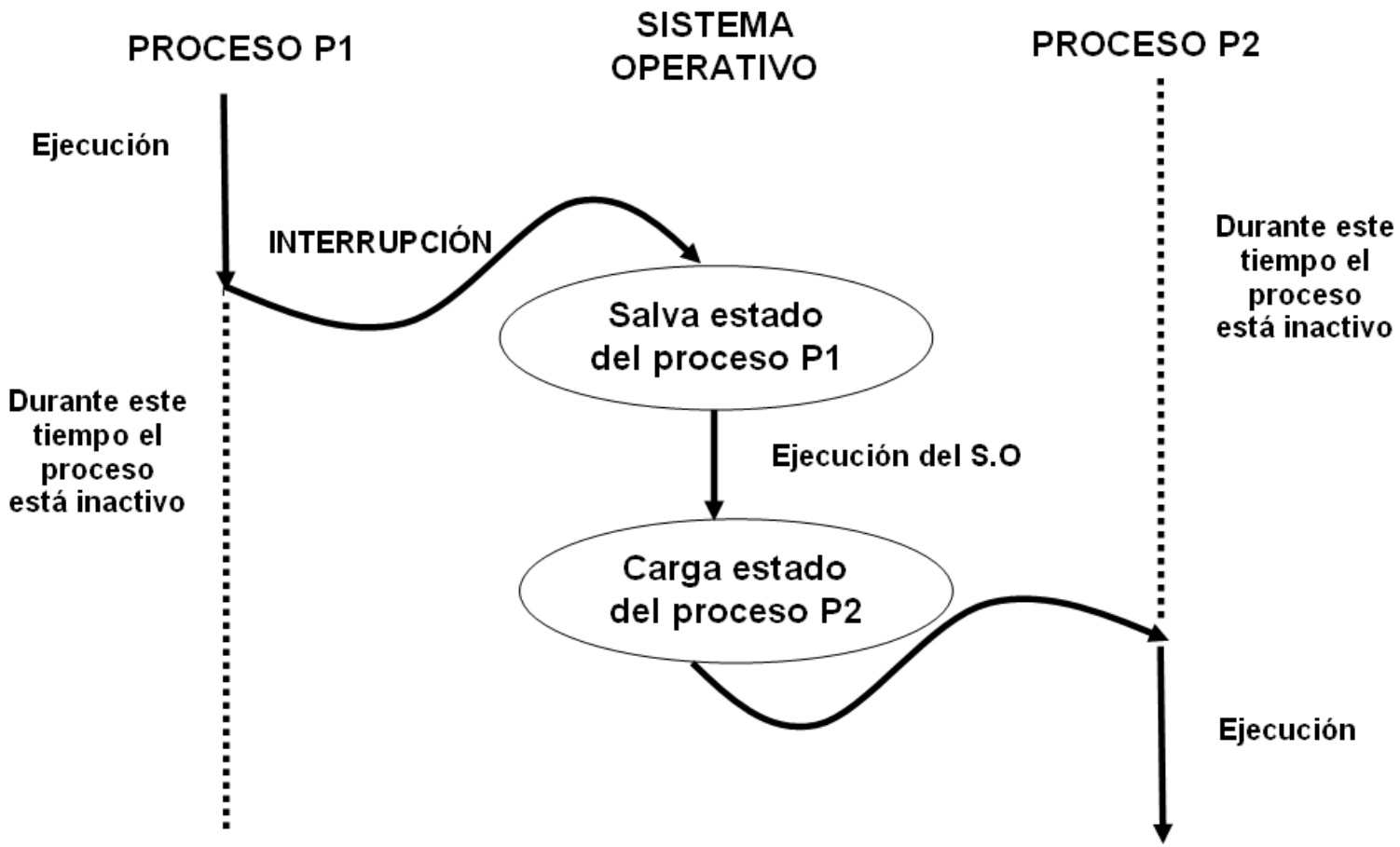
8. Interrupciones

8.4 Cambios de contexto



8. Interrupciones

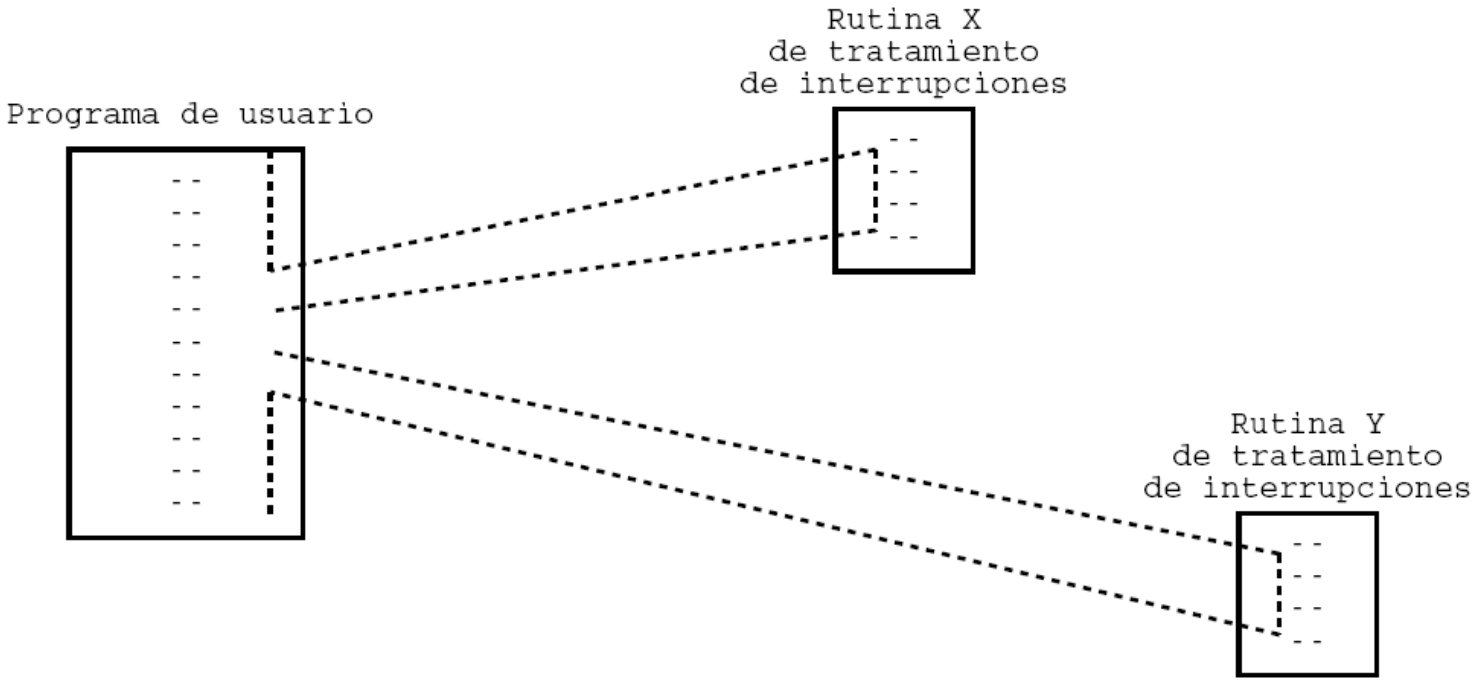
8.4 Cambios de contexto



8. Interrupciones

8.5 Interrupciones Múltiples

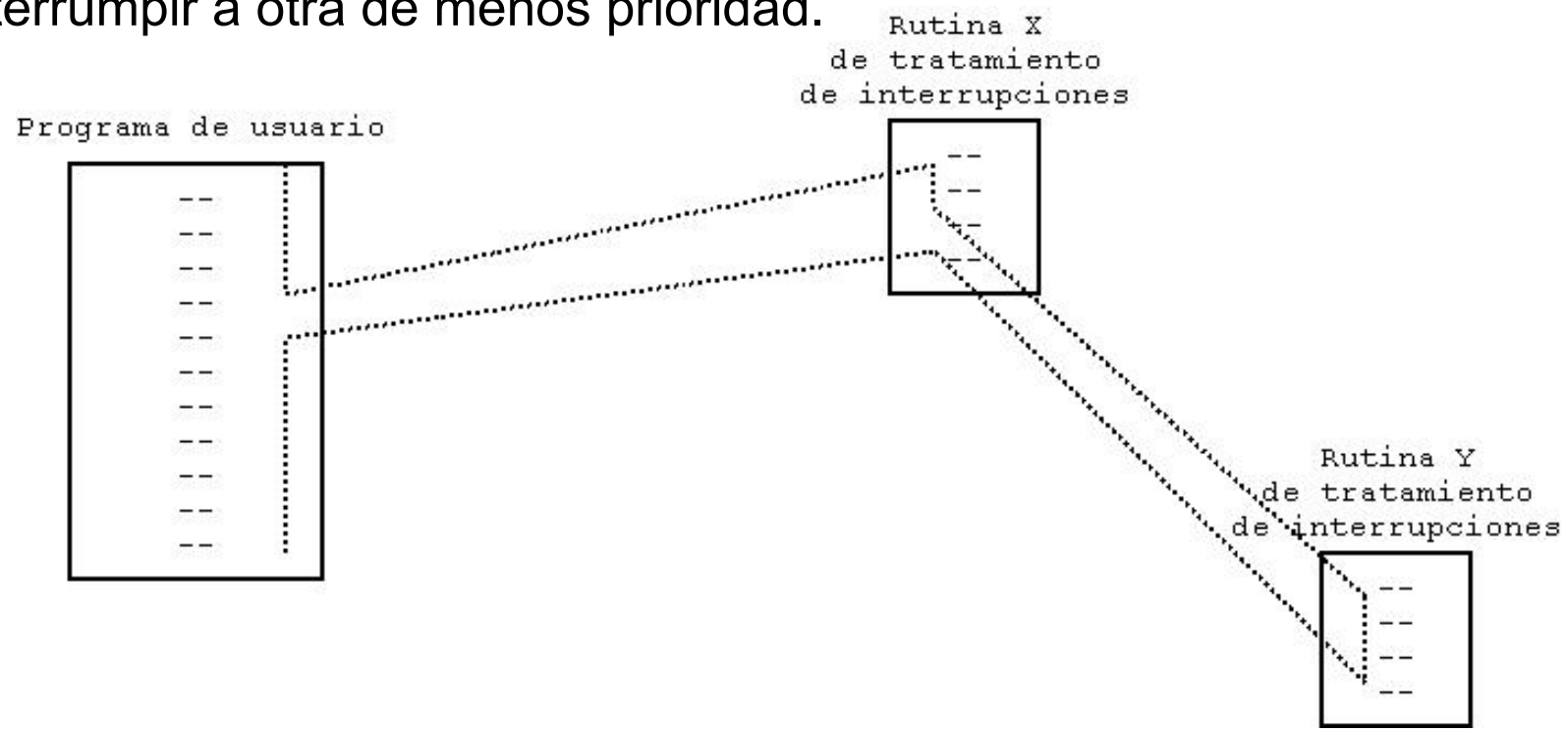
- Es posibles que lleguen varias interrupciones a la vez.
- Dos enfoques para resolverlo:
  - **Inhabilitar interrupciones** mientras se trata una, se ignorar las señales de interrupción que quedaran pendientes.



8. Interrupciones

8.5 Interrupciones Múltiples

- Es posibles que lleguen varias interrupciones a la vez.
- Dos enfoques para resolverlo:
  - **Definir prioridades**, una interrupción de alta prioridad puede interrumpir a otra de menos prioridad.



9. Planificación del procesador

- En un sistema multiprogramado tenemos **varios procesos** que se alternan en el uso del procesador y en esperar a realizar operaciones E/S u otro suceso.
- La clave de la eficiencia de la **multitarea** es la planificación.
- La planificación consiste en **asignar procesos al procesador** para que sean ejecutados a lo largo del tiempo cumpliendo una serie de objetivos.

9. Planificación del procesador

9.1 Niveles de planificación

- La actividad de planificación se divide en tres funciones o niveles independientes:
- **El planificador a largo plazo** (planificador de trabajos)
  - Determina cuáles son los programas **admitidos** al sistema, **controla el grado de multiprogramación.**
  - Crea los procesos y los coloca en cola, sacándolas de la misma cuando puedan ser **cargados en memoria**
  - Sólo existe cuando el sistema permite **procesos por lotes**
  - Decide quien se ejecuta basándose en criterios de **disponibilidad** y necesidad de recursos.
  - Su frecuencia de actividad es de **varios minutos**.
  - Su tarea es pasar los procesos por lotes de **inactivos a preparados**, y deciden el **grado de multiprogramación**

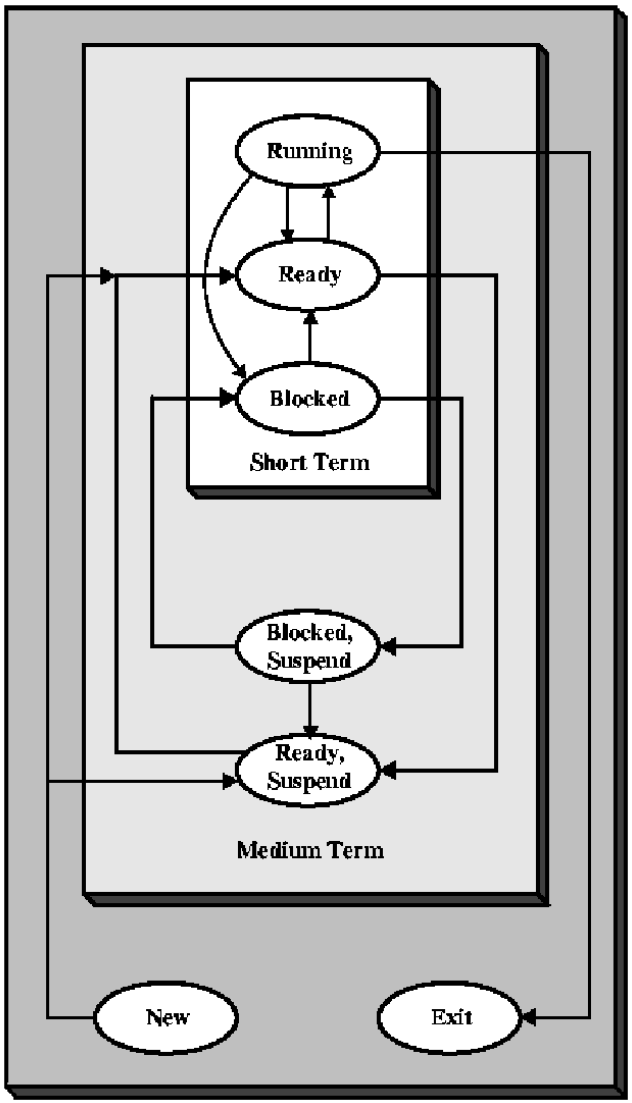
9. Planificación del procesador

9.1 Niveles de planificación

- **El planificador a medio plazo** (planificador de procesos suspendidos)
  - Decide si el proceso debe ser **suspendido** para reducir el grado de multiprogramación y devolverlo después a memoria cuando sea necesario.
  - Existe en sistemas de **tiempo compartido** y en aquellos que tengan gestión de memoria virtual o procesos intercambiables.
  - Su frecuencia de actividad es de **varios segundos**.
- **El planificador a corto plazo** (planificación del procesador)
  - Tiene la responsabilidad de decidir cómo y cuándo un proceso que está **preparado pasa a ejecutarse**, realiza las funciones de **multiprogramación**.
  - Siempre está en memoria, se suele ejecutar con muchas frecuencia (**milisegundos**) y es muy rápido.
  - Es en este nivel donde nos preocupamos de dar un buen servicio a los **procesos interactivos**.

9. Planificación del procesador

9.1 Niveles de planificación



9. Planificación del procesador

9.2 Objetivos de la planificación

- Cada planificador intenta alcanzar los siguientes objetivos:
  - **Justicia**
  - **Máxima capacidad de ejecución**
  - **Máximo número de usuarios interactivos**
  - **Minimización de la sobrecarga**
  - **Equilibrio del uso de recursos**
  - **Seguridad de las prioridades**
- Muchos de estos objetivos son contradictorios hay que llegar a un equilibrio.
- Resumen: el S.O debe ofrecer un **buen rendimiento y un buen servicio**.

9. Planificación del procesador

9.3 Criterios de planificación

- El diseñador estudiará el comportamiento del algoritmo según los **criterios** siguientes con objeto de observar si se alcanzan los objetivos fijados:
  - **Tiempo de respuesta**: velocidad con que el ordenador empieza a dar respuesta a una petición de usuario. Es sólo un factor psicológico.
  - **Tiempo de servicio** o retorno : tiempo que transcurre entre que el usuario da la orden de cargar hasta que finaliza su ejecución.

    $$T_{servicio} = T_{carga} + T_{preparado} + T_{tprocesador} + T_{bloqueado}$$

  - **Tiempo de ejecución**: representa el tiempo teórico que un proceso necesita para ejecutarse si estuviera solo.

    $$T_{ejecucion} = T_{carga} + T_{tprocesador} + T_{bloqueado}$$

  - **Tiempo de procesador**: tiempo que el proceso hace uso del procesador.
  - **Tiempo de espera**: tiempos de estancia en las colas de bloqueados y preparados.

9. Planificación del procesador

9.4 Medidas

- Sea ***t*** el tiempo de procesador, ***ti*** el momento de carga y ***tf*** el momento de finalización de un proceso ***p***.
- Definimos las siguientes medidas:
  - **Tiempo de Servicio**: ***Ts = tf − ti***
  - **Tiempo de Espera**: **Te = Ts − t**
  - **Índice de Servicio**: ***I = t / Ts***
    Es el porcentaje de tiempo que el proceso está usando el procesador con respecto a su tiempo de vida. Mide el rendimiento. Es **adimensional** y se suele expresar en %.

9. Planificación del procesador

9.4 Medidas

- **Interpretación del Indice de Servicio**:
  - Para **I=1** el proceso siempre estará ejecutándose (raro porque suelen necesitar E/S).
  - Para **I=0** el proceso nunca se ejecuta.
  - Cuanto más alto sea I menos desperdicio se produce.
  - Si I está próximo a 1 decimos que el proceso está **limitado por el procesador**.
  - Si I está próximo a 0 decimos que el proceso está **limitado por E/S**.

9. Planificación del procesador

9.4 Medidas

- Para sistema multitarea con n procesos, usaremos los promedios:
  - **Tiempo medio de servicio**. $Tms = \sum Ts / n$
  - **Tiempo medio de espera**. $Tme = \sum Te / n$
  - **Índice medio de servicio** $\mathrm{Im} = \sum I / n$
- Siendo n el número de procesos.

9. Planificación del procesador

9.5 Organización de la planificación

- La tarea de pasar un proceso de preparado a ejecución (realizada por el núcleo) se divide en:
  - Planificación de procesos (**planificador**): elige qué proceso entre los que están en la cola de preparados pasa a ejecución.
  - Despachar un proceso (activador o **despachador**): entrega al procesador, extrayéndolo de la cola de procesos preparados, cambiando su estado y reponiendo el estado del procesador.

9. Planificación del procesador

9.6 Algoritmos de planificación

- Nos basamos en un **grupo de procesos** que se deben ejecutar en el ordenador en los instantes dados en la siguiente tabla:

Proceso	t_i	$t_{ejecucion}$	Cola
A	0	3	1
B	1	5	1
C	4	2	2
D	5	6	3
E	8	4	3

- De esta forma **evaluando** los diferentes algoritmos de planificación bajo la **misma carga** podemos comparar la 'bondad' de dichos algoritmos.

9. Planificación del procesador

9.6 Algoritmos de planificación

- Suponemos que **no se realiza E/S**, con lo cual el tiempo de ejecución coincide con el tiempo de procesador
- La **unidad de tiempo es virtual**, con lo cual no nos importa su valor, aunque el valor más representativo del comportamiento de la política se mide en % y es adimensional (Índice de Servicio).
- Vamos a tener dos **tipos de política de planificación**:
  - **Apropiativas**: Interrumpen la ejecución de un proceso con el fin de estudiar si puede seguir siendo ejecutado o por el contrario debe ser otro el que lo haga.
  - **No apropiativas**: Un proceso nunca abandona el procesador una vez comenzada su ejecución.

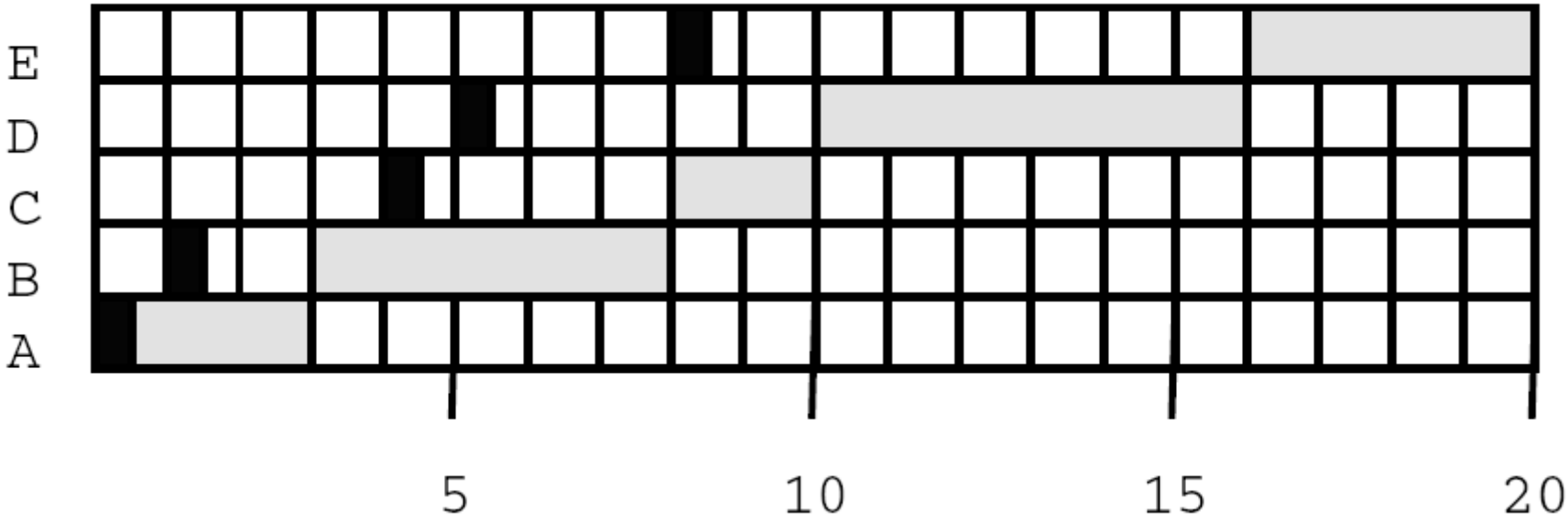
9. Planificación del procesador

9.6.1 Algoritmo primero en llegar, primero en servirse (FCFS - FIFO)

- Es una **política no apropiativa**, el proceso se ejecuta hasta que termine.
- Se basa única y exclusivamente en el **orden de llegada de los procesos.** Tras una E/S, el proceso se colocará al final de la cola de preparados.
- **Ventajas**:
  - Es fácil de llevar a la práctica.
  - Es justa.
  - Es predecible.
- **Desventajas**:
  - Penaliza los procesos cortos, sobre todo cuando llegan detrás de uno largo. Las esperas siempre dependen de los procesos que estén en la cola. Ts no suele ser el deseado.

9. Planificación del procesador

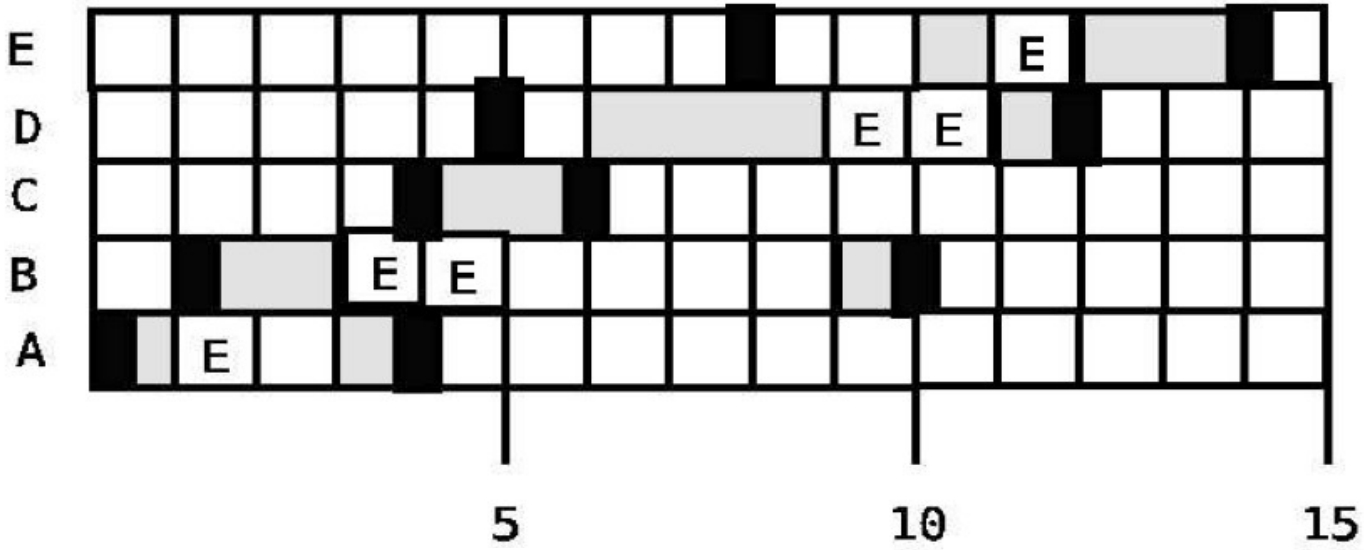
9.6.1 Algoritmo primero en llegar, primero en servirse (FCFS - FIFO)



Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	3	3	3	0	1
B	1	5	8	7	2	0.71
C	4	2	10	6	4	0.33
D	5	6	16	11	5	0.54
E	8	4	20	12	8	0.33
			Media	7.8	3.8	0.58

9. Planificación del procesador

9.6.1 Algoritmo primero en llegar, primero en servirse (FCFS - FIFO)



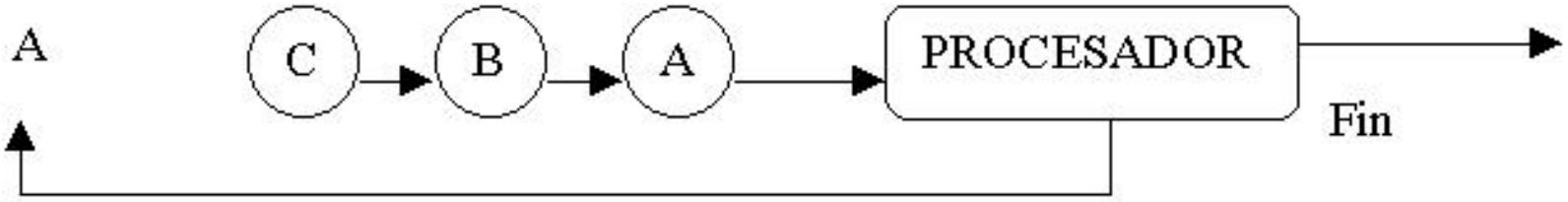
Proceso	t_i	Ejecucion	Cola
A	0	1 + E/S + 1	1
B	1	2 + 2 E/S + 1	1
C	4	2	2
D	5	3 + 2 E/S + 1	3
E	8	1 + 1 E/S + 2	3

Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	2	4	4	2	2/4 =0,5
B	1	3	10	9	6	3/9 = 0,33
C	4	2	6	2	0	2/2 = 1
D	5	4	12	7	3	4/7 = 0,57
E	8	3	14	6	3	3/6 = 0,5
			Media	5,6	2,8	0,58

9. Planificación del procesador

9.6.2 Algoritmo Round Robin (RR)

- Se traduce como rueda o asignación cíclica (Round Robin).
- Trata de ser más justa que la FIFO para procesos cortos.
- Es **apropiativa**, concede el procesador a un proceso durante un **quantum** 'q' de tiempo (10-100 msg.), antes de devolverlo a la cola de procesos preparados.
- La cola de procesos **preparados** se gestiona mediante una **FIFO** (aunque a veces se hace por prioridades).



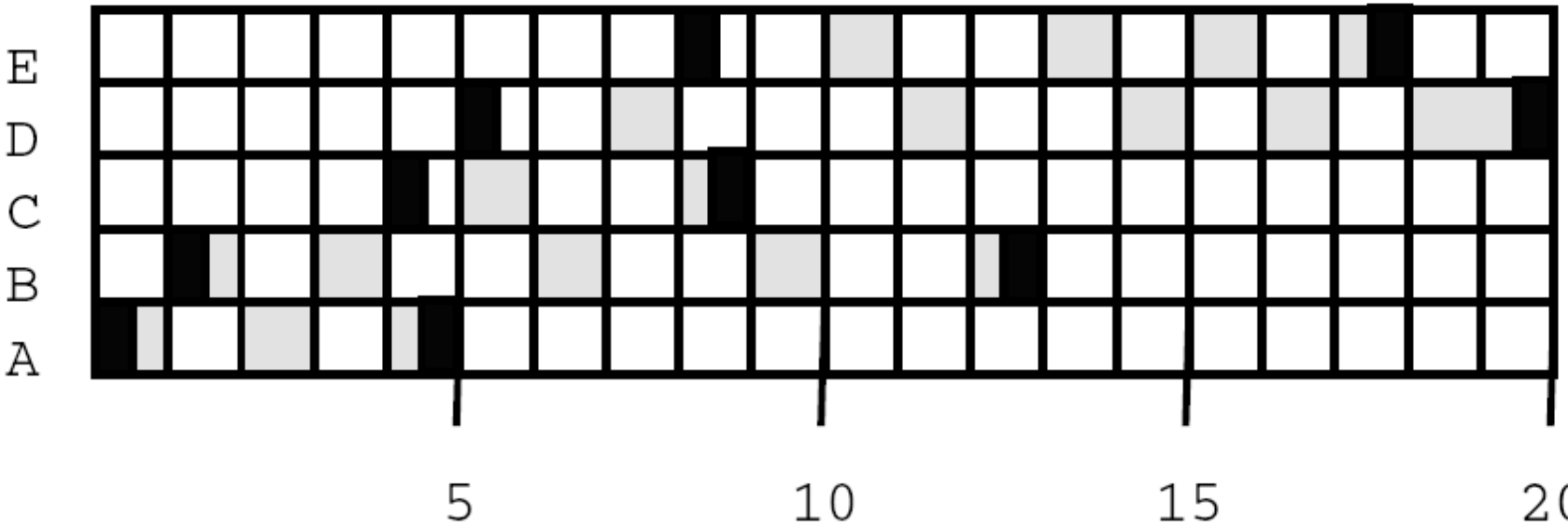
¿Influye 'q' en el comportamiento?

- q grande ⇒ FIFO.
- q pequeño ⇒ sobrecarga.

9. Planificación del procesador

9.6.2 Algoritmo Round Robin (RR)

Quantum = 1

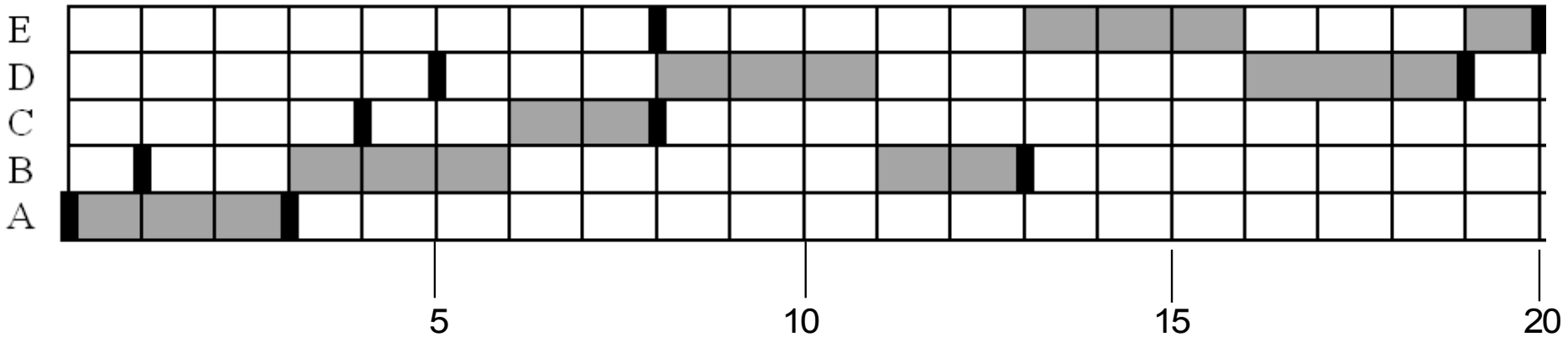


Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	3	5	5	2	0.6
B	1	5	13	12	7	0.42
C	4	2	9	5	3	0.40
D	5	6	20	15	9	0.40
E	8	4	18	10	6	0.40
			Media	9.5	5.4	0.44

9. Planificación del procesador

9.6.2 Algoritmo Round Robin (RR)

Quantum = 3

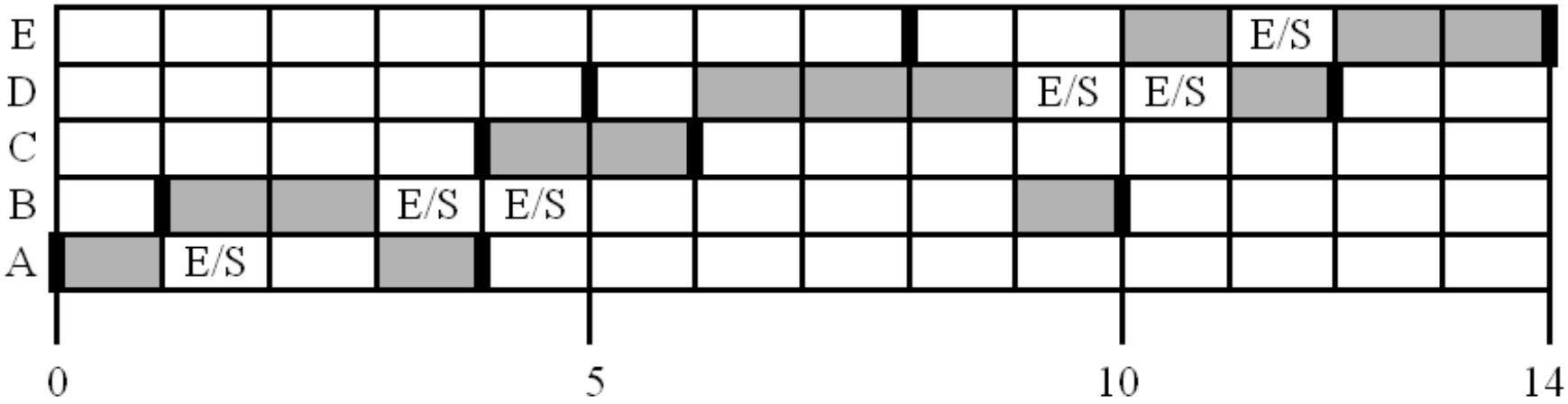


Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	3	3	3	0	1
B	1	5	13	12	7	0.42
C	4	2	8	4	2	0.50
D	5	6	19	14	8	0.43
E	8	4	20	12	8	0.33
			Media	9	5	0.54

9. Planificación del procesador

9.6.2 Algoritmo Round Robin (RR)

Quantum = 3



Proceso	t_i	Ejecucion	Cola
A	0	1 + E/S + 1	1
B	1	2 + 2 E/S + 1	1
C	4	2	2
D	5	3 + 2 E/S + 1	3
E	8	1 + 1 E/S + 2	3

Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	2	4	4	2	2/4 = 0,5
B	1	3	10	9	6	3/9 = 0,33
C	4	2	6	2	0	2/2 = 1
D	5	4	12	7	3	4/7 = 0,57
E	8	3	14	6	3	3/6 = 0,5
			Media	5,6	2,8	0,58

9. Planificación del procesador

9.6.2 Algoritmo Round Robin (RR)

- **Conclusiones**:
  - Con q=1 el tiempo de servicio permanece prácticamente constante.
  - Con q=3 el tiempo de espera crece con la longitud de los procesos.
- **Propiedades** de la política:
  - Baja sobrecarga si el cambio de contexto es eficiente y los procesos están en memoria principal.
  - Es la más usada para tiempo compartido.
  - Ofrece un índice de servicio uniforme para todos los procesos.

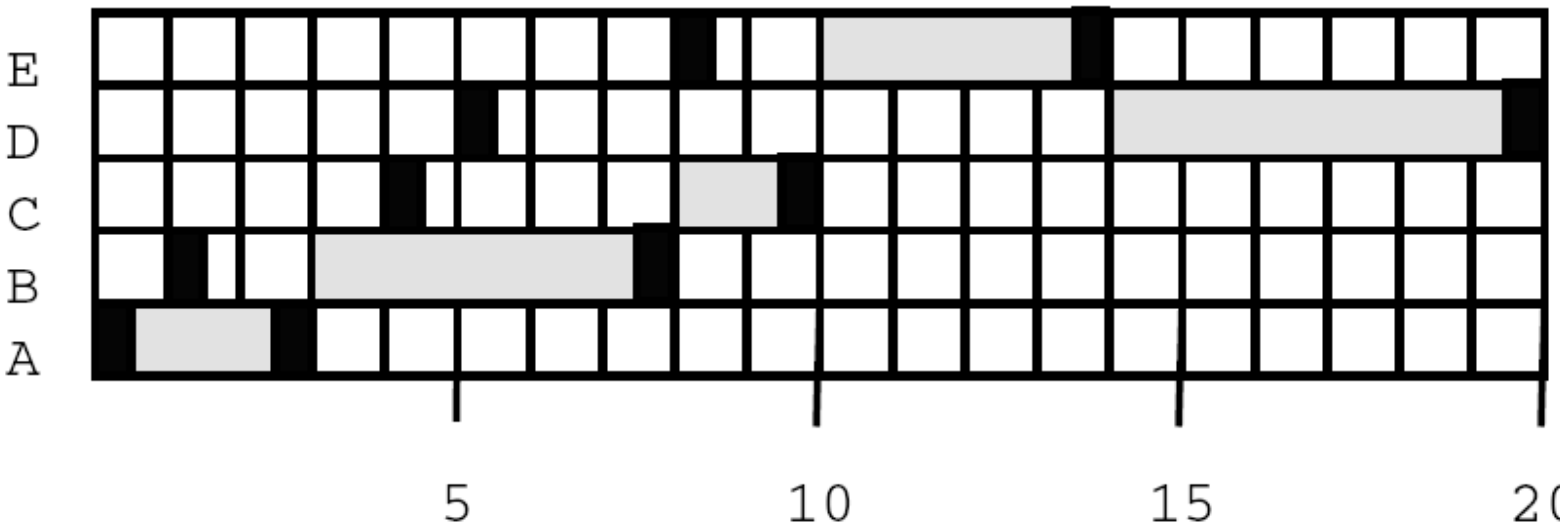
9. Planificación del procesador

9.6.3 El siguiente proceso el más corto (SJF)

- Es una política **no apropiativa**.
- Se escoge de la cola de procesos preparados el de **menor tiempo de ejecución**. En caso de igualdad por orden de llegada. Tras una entrada salida, el proceso que vuelve a la cola de preparados tiene la misma consideración que uno nuevo (se vuelve a calcular el tiempo restante).
- **¿Cómo conocer a priori el tiempo de ejecución de los procesos?**
  - Lo proporciona el usuario, poco fiable.
  - Cada proceso se auto describe, difícil de conseguir ya que los procesos no siempre se ejecutan igual.
  - Tomar la decisión de forma Heurística, basándose en información recabada estadísticamente. Sólo es posible en ambientes de ejecución muy repetitivos.

9. Planificación del procesador

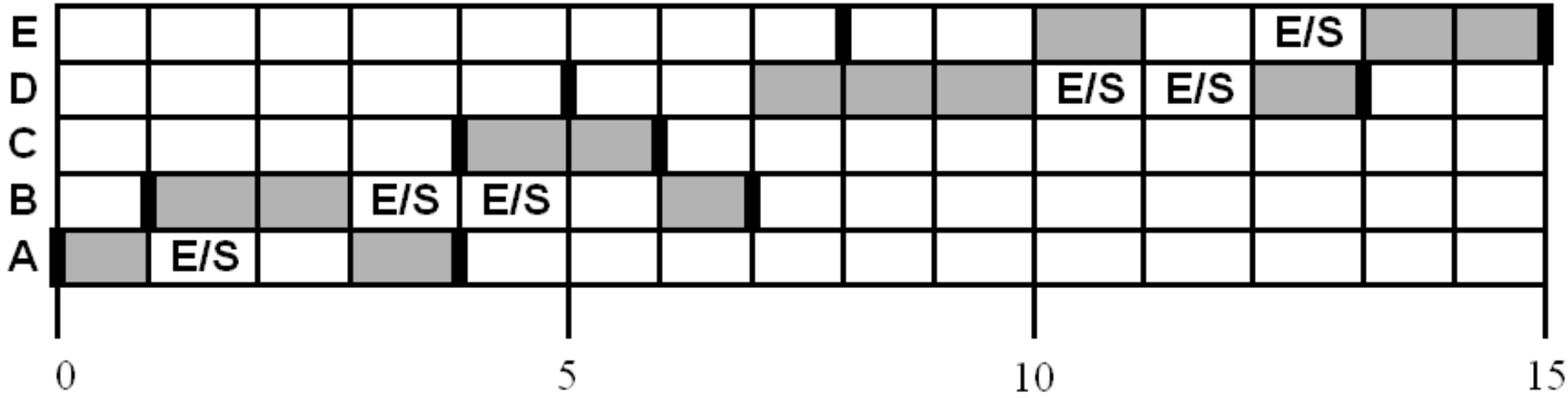
9.6.3 El siguiente proceso el más corto (SJF)



Proceso	t_i	t	t_f	T_s	T_e	I
A	0	3	3	3	0	1
B	1	5	8	7	2	0.71
C	4	2	10	6	4	0.33
D	5	6	20	15	9	0.40
E	8	4	14	6	2	0.67
			Media	7.4	3.4	0.62

9. Planificación del procesador

9.6.3 El siguiente proceso el más corto (SJF)



Proceso	t_i	Ejecucion	Cola
A	0	1 + E/S + 1	1
B	1	2 + 2 E/S + 1	1
C	4	2	2
D	5	3 + 2 E/S + 1	3
E	8	1 + 1 E/S + 2	3

Proceso	t_i	t	t_f	T_s	T_e	I
A	0	2	4	4	2	2/4 = 0,5
B	1	3	7	6	3	3/6 = 0,5
C	4	2	6	2	0	2/2 = 1
D	5	4	13	8	4	4/8 = 0,5
E	8	3	15	7	4	3/7 = 0,43
			Media	5,4	2,6	0,59

9. Planificación del procesador

9.6.3 El siguiente proceso el más corto (SJF)

- **Propiedades**
  - El tiempo de espera aumenta con la longitud de los procesos.
  - Es poco predecible
  - No es justa, favorece a los procesos cortos.
  - Tiene un buen Ts (sobre todo para procesos cortos) e I.
  - Es difícil de poner en práctica por la dificultad de conocer los tiempos de ejecución a priori.

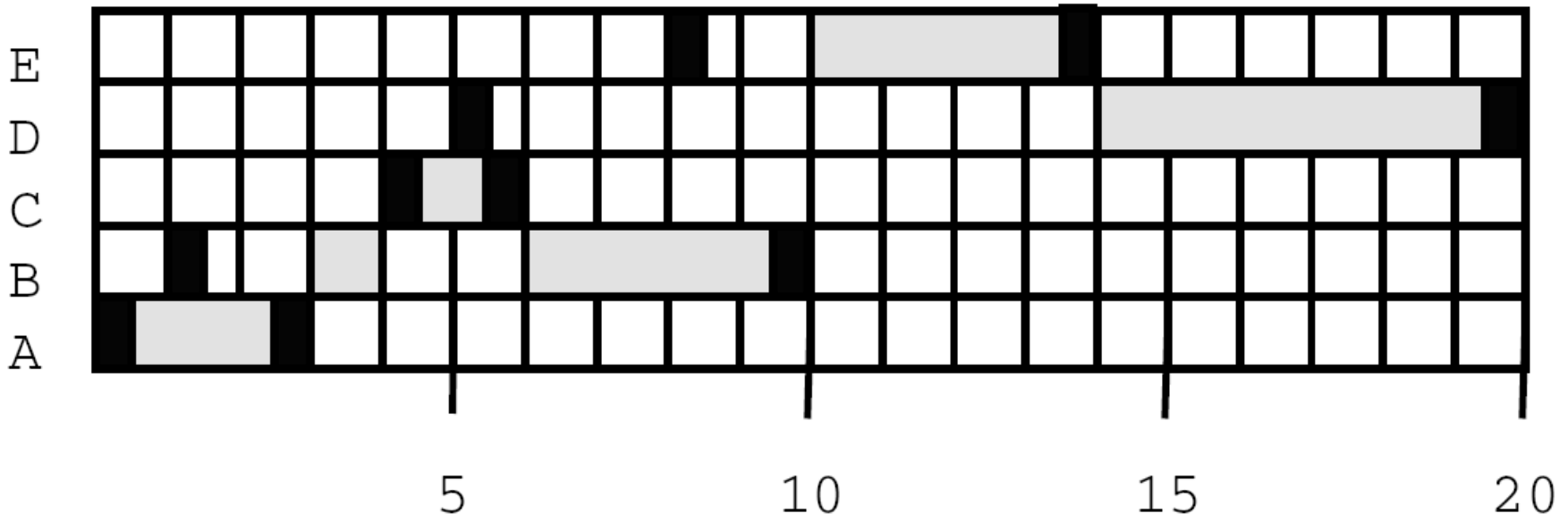
9. Planificación del procesador

9.6.4 Próximo proceso el de tiempo restante más corto (SRT)

- Es una política **apropiativa**.
- Trata de conjugar la ventaja de la RR (apropiación) con la SJF (información del tiempo de los procesos)
- Cambia el proceso que está en ejecución cuando un proceso entra en la cola de preparados con una **exigencia de tiempo de ejecución menor que le queda al proceso que está en curso**.

9. Planificación del procesador

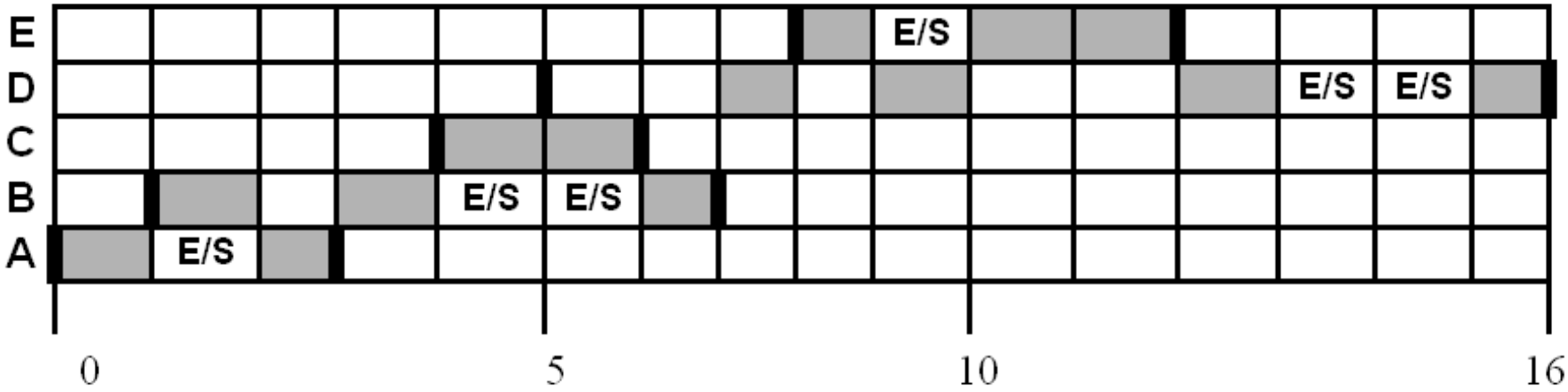
9.6.4 Próximo proceso el de tiempo restante más corto (SRT)



Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	3	3	3	0	1
B	1	5	10	9	4	0.55
C	4	2	6	2	0	1
D	5	6	20	15	9	0.4
E	8	4	14	6	2	0.67
			Media	7	3	0.72

9. Planificación del procesador

9.6.4 Próximo proceso el de tiempo restante más corto (SRT)



Proceso	t_i	Ejecucion	Cola
A	0	1 + E/S + 1	1
B	1	2 + 2 E/S + 1	1
C	4	2	2
D	5	3 + 2 E/S + 1	3
E	8	1 + 1 E/S + 2	3

Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	2	3	3	1	2/3 = 0,66
B	1	3	7	6	3	3/6= 0,5
C	4	2	6	2	0	2/2 = 1
D	5	4	16	11	7	4/11 = 0,37
E	8	3	12	4	1	3 /4 = 0,75
			Media	5.2	2,4	0,66

9. Planificación del procesador

9.6.4 Próximo proceso el de tiempo restante más corto (SRT)

- Las **características** de la política son:
  - Excelente I y un Te muy bajo (la cola de preparados es lo más corta posible ⇒alto rendimiento.).
  - Bajo Tiempo de Espera para la mayoría de los procesos.
  - Puede ser injusta, incluso causar 'inanición'.
  - Gran sobrecarga ya que hay que almacenar el tiempo de ejecución consumido por cada proceso.

9. Planificación del procesador

9.6.5 Próximo el de más alto índice de respuesta (HRN)

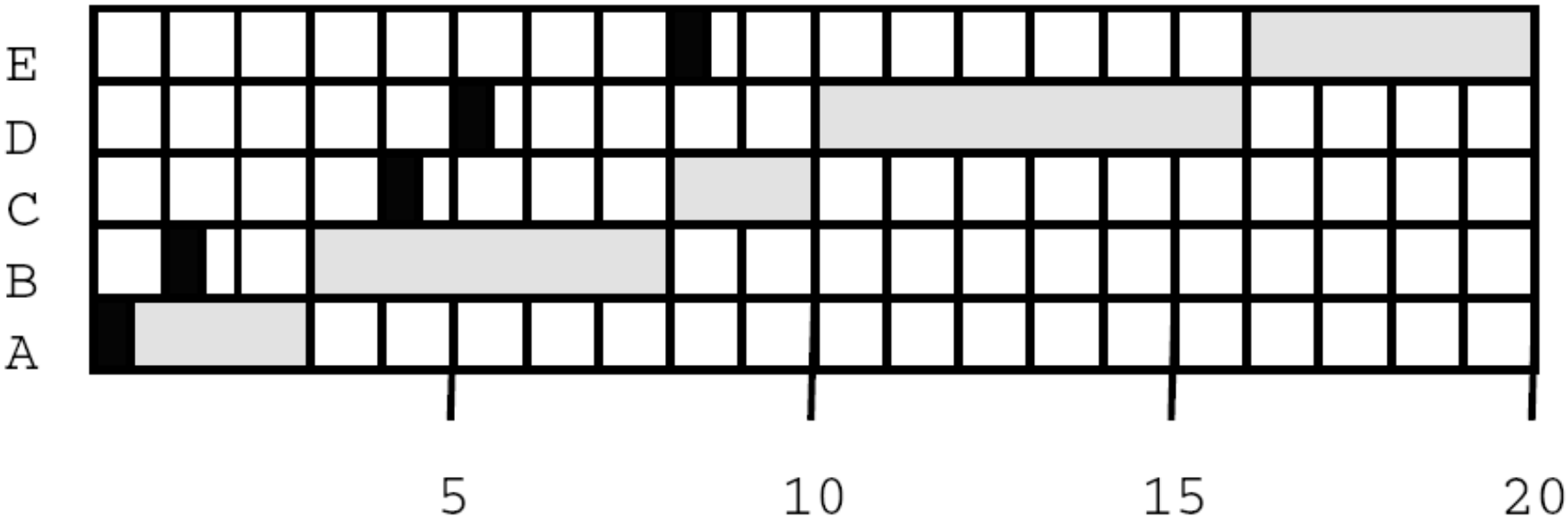
- Es una política **no apropiativa**.
- Trata de corregir la desventaja de los procesos largos en el SJF.
- Calcula la expresión de la prioridad interna (P) como:

$$P = \frac{w + t_e}{t_e}$$

  - donde w es el tiempo de espera en la cola de preparados y te es el tiempo de ejecución, cada vez que sea necesario planificar. Tras una E/S se considerará el tiempo de ejecución total, no el restante, y el tiempo de espera total en preparados.
- Favorece a los procesos que más tiempo lleven en el sistema y a los más cortos.

9. Planificación del procesador

9.6.5 Próximo el de más alto índice de respuesta (HRN)



Cálculo de la prioridad interna				
	C	D	E	Seleccionado
t=8	P=6/2=3	P=9/6=1.5	P=4/4=1	C
t=10		P=11/6=1.8	P=6/4=1.5	D

Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	3	3	3	0	1
B	1	5	8	7	2	0.71
C	4	2	10	6	4	0.33
D	5	6	16	11	5	0.54
E	8	4	20	12	8	0.33
			Media	7.8	3.8	0.58

9. Planificación del procesador

9.6.5 Próximo el de más alto índice de respuesta (HRN)

- **Propiedades**:
  - Un proceso corto tras uno largo espera mucho sólo si el largo ha comenzando su ejecución.
  - Es justa.
  - Es muy costosa al tener que calcular la prioridad interna P, lo que produce mucha sobrecarga

9. Planificación del procesador

9.6.6 Colas Múltiples (MQ)

- Se divide la cola de procesos preparados en varias colas, los procesos se asignan de forma permanente a una determinada.
- Cada cola tendrá asociado un algoritmo de planificación.
- ¿Cuándo se pasa el control a cada cola? ⇒ algoritmo de planificación entre colas.

9. Planificación del procesador

9.6.6 Colas Múltiples (MQ)

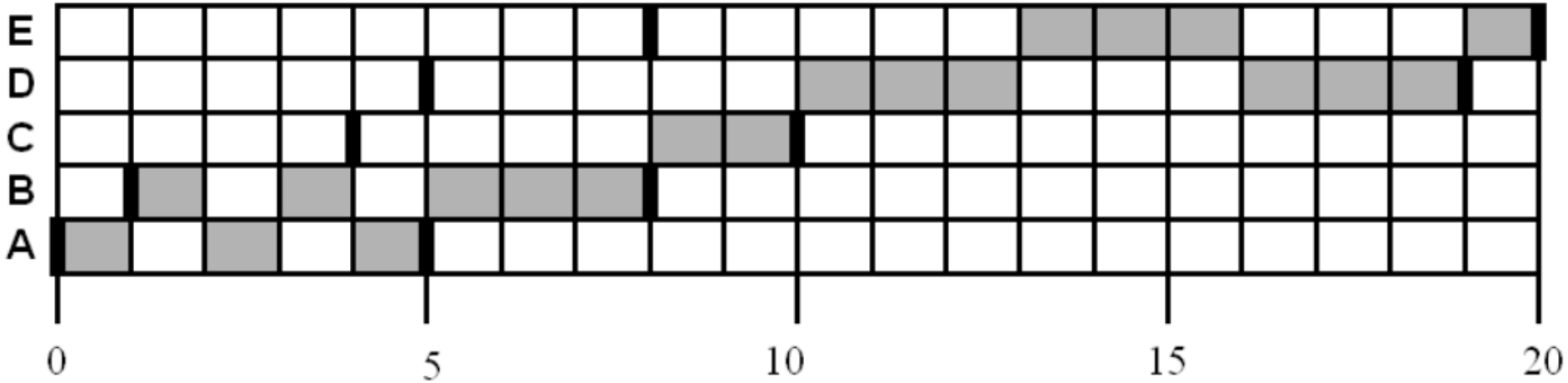
Ejemplo 1:

- Tres colas con algoritmo RR en cada una de ellas con q = 1, 2 y 3 respectivamente.
- El algoritmo entre colas es de prioridades apropiativo, la cola 1 es la más prioritaria

Proceso	t_i	t	Cola
A	0	3	1
B	1	5	1
C	4	2	2
D	5	6	3
E	8	4	3

9. Planificación del procesador

9.6.6 Colas Múltiples (MQ)



Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	3	5	5	2	0.6
B	1	5	8	7	2	0.71
C	4	2	10	6	4	0.3
D	5	6	19	14	8	0.43
E	8	4	20	12	8	0.33
			Media	8.8	4.8	0.47

9. Planificación del procesador

9.6.7 Realimentación de colas múltiples (FB)

- Si queremos tratar de formas justa a los procesos necesitamos conocer su longitud, memoria, si está limitado por proceso o por E/S, etc...
- Nos conformamos con asumir las siguientes reglas:
  - Favorecer a los procesos cortos.
  - Favorecer a los procesos limitados por E/S, porque así ocupan los recursos y dejan libres el procesador para procesos limitados por proceso.
  - Determinar la naturaleza del trabajo a realizar para poder realizar su planificación.

9. Planificación del procesador

9.6.7 Realimentación de colas múltiples (FB)

- Se divide la cola de procesos preparados en varias, de la forma cola1, cola2, cola3, ...
- Para cada cola se le concede al proceso un determinado tiempo de procesador Tn, de forma que si excede su tiempo se **pasa a la cola de nivel inmediatamente inferior** (con esto vamos disminuyendo su prioridad mientras más tiempo está en el sistema)
- El tiempo que cada proceso está en una cola de un determinado nivel vendrá dado como parámetro de la política. El tiempo de la última fila será infinito.
- **NOTA**: en colas múltiples, si un proceso de una cola menos prioritaria, es interrumpido por la llegada de un proceso a una cola de prioridad mayor, tomaremos como criterio, que se volverá a evaluar la situación de la cola menos prioritaria aplicando el criterio de la cola.

9. Planificación del procesador

9.6.7 Realimentación de colas múltiples (FB)

- **Parámetros** necesarios para definir la política:
  - Número de colas a usar.
  - Algoritmo de planificación de cada cola.
  - Método de paso de una cola a otra.
  - Cola en la que comienza cada proceso.
  - Algoritmo de planificación entre las distintas colas.

9. Planificación del procesador

9.6.7 Realimentación de colas múltiples (FB)

- Se intenta dar un trato justo, separando los procesos por categoría.
- Los procesos limitados por procesador terminarán en las colas del nivel más bajo. Dejando en las de mayor prioridad los procesos más interactivos.
- **Características** de la política:
  - Soporta bien cargas de procesos altas. Los procesos se reparten entre las colas.
  - Es adaptable a las necesidades del sistema (Según gestionemos cada cola)
  - Es apropiativa entre colas.

9. Planificación del procesador

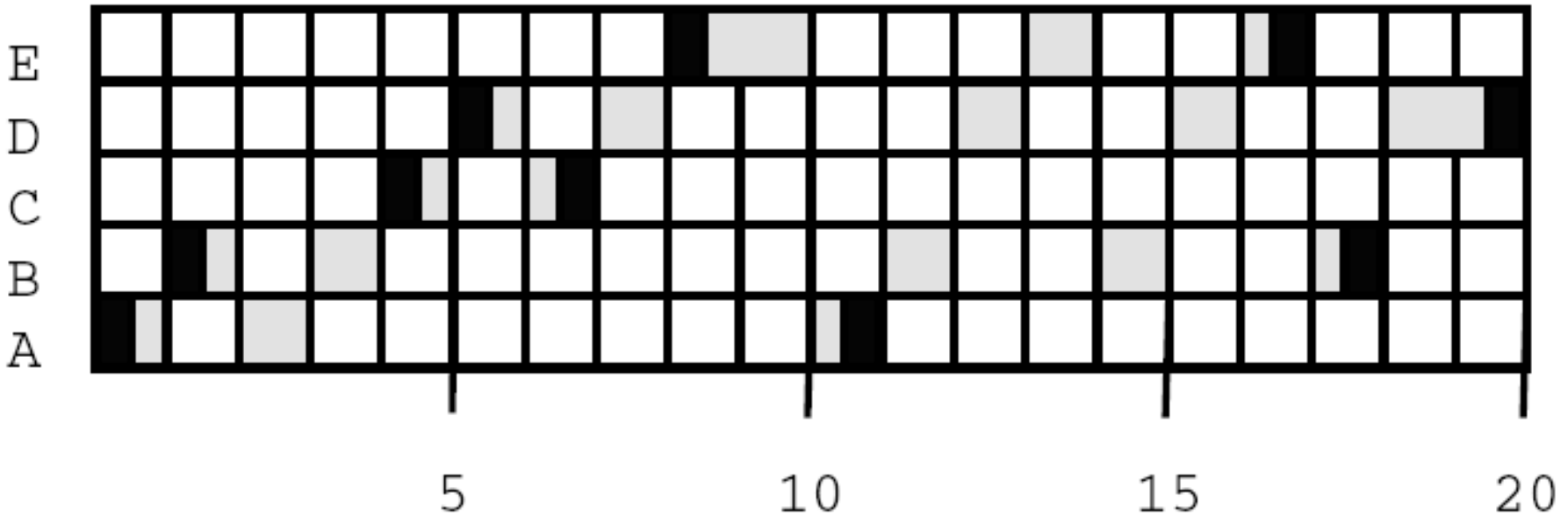
9.6.7 Realimentación de colas múltiples (FB)

Ejemplo 1:

- Número de colas a usar: **Las necesarias**
- Algoritmo de planificación de cada cola: **RR de q=1**
- Método de paso de una cola a otra: **Agotar el quantum**
- Cola en la que comienza cada proceso: **La más alta**
- Algoritmo de planificación entre las distintas colas: **Prioridades Apropiativas**

9. Planificación del procesador

9.6.7 Realimentación de colas múltiples (FB)



Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	3	11	11	8	0.27
B	1	5	18	17	12	0.29
C	4	2	7	3	1	0.67
D	5	6	20	15	9	0.40
E	8	4	17	9	5	0.44
			Media	7.8	3.8	0.41

9. Planificación del procesador

9.6.8 La prioridad y los algoritmos de planificación

- Hasta ahora no hemos tenido en cuenta las **prioridades externas**.
- Normalmente los sistemas desdoblan las colas en tantas como prioridades puedan alcanzar los procesos (de la forma en que lo hacen las colas múltiples)
- El problema de las prioridades es la **postergación indefinida** o **inanición** que pueden sufrir ciertos procesos. Para evitarlo se usa la técnica del **'envejecimiento de prioridades'**, mediante la cual el proceso va ganando prioridad conforme más tiempo está en el sistema.
- Estos algoritmos de envejecimiento de prioridades pueden ser además **apropiativos** o **no apropiativos**, según las acciones que tomen con el proceso que está en ejecución en un momento dado.

9. Planificación del procesador

9.7 Planificación en POSIX

- POSIX es el acrónimo de *Portable Operating System Interface*; la X viene de UNIX como seña de identidad de la API.
- POSIX especifica las interfaces de usuario y software al Sistema Operativo en 15 documentos diferentes.
- Con respecto a la planificación especifica una serie de políticas de planificación aplicable a procesos y procesos ligeros.
- Cada unidad ejecutable lleva asociada una **política** y una **prioridad**, que pueden ser modificados.
- Cada política de planificación lleva asociada un rango de prioridades (al menos 32 niveles). Se elegirá siempre al proceso o hilo con la prioridad más alta.

9. Planificación del procesador

9.7 Planificación en POSIX

Las políticas disponibles en POSIX son:

- **FIFO** modificada:
  - El proceso que es expulsado por otro de mayor prioridad pasa a ser el primero de su cola.
  - Un desbloqueo ⇒ último de su cola.
  - Cambio de prioridad o política ⇒ replanificación. Si tras esta el proceso resulta expulsado ⇒ último de su cola.
- **Cíclica** (Round-Robin).
  - Fin quantum ⇒ final de la cola de su prioridad.
  - El proceso que es expulsado por otro de mayor prioridad pasa a ser el primero de su cola pero con el resto de quantum que aún no había consumido.
- **Otra**: Esta tercera política es opcional y dependerá de la implementación que se realice de la norma POSIX.

9. Planificación del procesador

9.7 Planificación en POSIX

Implementación LINUX del estándar POSIX

- Soporta tres tipos de planificación: **dos algoritmos de TR** (tiempo real) y **un algoritmo de TC** (tiempo compartido).
- Las prioridades de los procesos en TR son siempre mayores que las de TC.
- Las políticas de planificación son:
  - Las de los procesos en TR son las definidas en el estándar POSIX con prioridades estáticas. Los niveles de prioridad van desde 1 a 99.
  - La política de TC es la tercera política de planificación del estándar POSIX, y está implementada mediante una **RR con prioridades dinámicas**. La prioridad estática es 0.

9. Planificación del procesador

9.7 Planificación en POSIX

Implementación LINUX del estándar POSIX

- Funcionamiento de la política de TC:
  - Todo proceso tiene asignada una prioridad base (dinámica).
  - Cada vez que se produce la interrupción del reloj se resta una unidad a la prioridad del proceso en ejecución.
  - Se elige al proceso más prioritario de la cola.
  - Si llega un momento en que todos los procesos preparados tienen una prioridad igual a 0 se produce un reajuste de las prioridades.
  - La nueva prioridad se calcula dividiendo por 2 la actual (redondeando por exceso) y sumando la prioridad base ⇒ procesos preparados vuelven a su prioridad base y bloqueados aumentan (prioridad mayor a la de los que ya estaban)
  - ¿Qué ocurre cuando un proceso del tipo 'tiempo compartido' es expulsado cuando llega un proceso en tiempo real? No está documentado.

9. Planificación del procesador

9.8 Planificación Windows

- Se utiliza un algoritmo de planificación **Round Robin apropiativo basado en prioridades.**
- La unidad de planificación son las **hebras**. Siempre se ejecuta la **hebra más prioritaria**. Si llega a preparado una hebra más prioritaria que la que está en ejecución, ésta es desalojada.
- Una hebra en ejecución sólo abandona ese estado si:
  - Es desalojada por una hebra de mayor prioridad
  - Termina
  - Concluye su quantum de tiempo
  - Se bloquea
- La parte del kernel de Windows que gestiona la planificación se llama **despachador**. El despachador **usa un esquema de prioridades de 32 niveles.**

9. Planificación del procesador

9.8 Planificación Windows

- Las prioridades se dividen en dos clases:
  - **Clase variable**: prioridades de 1 a 15 (cambian arriba o abajo pero nunca a prioridades mayores de 15)
  - **Clase de tiempo real**: prioridades de 16 a 31 (son fijas, nunca cambian)
- Existe una hebra (usada para gestión de memoria) con prioridad 0
- Hay una cola para cada prioridad.
- Si no hay ninguna hebra preparada se ejecuta una hebra especial llamada **hebra inactiva**.

9. Planificación del procesador

9.8 Planificación Windows

- La **prioridad inicial de un hilo** de la clase de prioridad variable se determina en base a **dos cantidades**:
  - la prioridad base del proceso (0-15)
  - la prioridad base del hilo (prioridad base del proceso ± 2)
- Si un hilo **agota su quantum se baja su prioridad**.
- Si un hilo se interrumpe para esperar por una **E/S se sube su prioridad.**
- Los hilos limitados por procesador tienden a prioridades menores y los hilos limitados por E/S tienden a prioridades mayores.
- Dentro de los hilos limitados por E/S **se sube más la prioridad a los que realizan esperas interactivas** (teclado, pantalla) que a los que realizan otro tipo de esperas de E/S (disco), por lo que los hilos interactivos tienden a tener mayores prioridades.

9. Planificación del procesador

9.9 Planificación clásica en UNIX

- Está diseñado para entornos de TC aunque SVR4 también cubres las necesidades del TR.
- Emplea **realimentación multinivel** usando **RR con, q = 1 sg**., en cada una de las colas de prioridad.
- ¿Cómo se calcula la prioridad? $CPU_j(i) = CPU_j \frac{(i-1)}{2}$

$$P_j(i) = Base_j + CPU_j(i) + Nice_j$$

  - donde ***CPU$_j$(i)*** es la media ponderada de la utilización de la C.P.U del proceso *j* en el intervalo ***i***, ***P$_j$(i)*** es la prioridad del proceso ***j*** al principio del intervalo ***i***, ***Base$_j$*** es la prioridad base del proceso ***j*** y ***nice$_j$*** es un factor de ajuste controlado por el usuario.
- El propósito de la prioridad base es **dividir los procesos en bandas**

9. Planificación del procesador

9.9 Planificación clásica en UNIX

- Los valores de C.P.U y *nice* están restringidos para impedir que el proceso salga de la banda asignada.
- Las bandas que se definen, en orden decreciente de prioridad son:
  - Intercambio
  - Control de dispositivos de E/S de bloques.
  - Gestión de archivos.
  - Control de dispositivos de E/S de caracteres.
  - Procesos de usuario.
- Esta jerarquía garantiza la utilización eficiente de los dispositivos de E/S.
- Dentro de la banda de usuario se trata de penalizar a los procesos con carga de procesador a costa de los procesos con carga de E/S.

9. Planificación del procesador

10 Comunicación y sincronización entre procesos

- Los procesos interaccionan entre si.
- Normalmente son controlados por los programadores para beneficiarse de la **concurrencia**.
- Los procesos deben **sincronizarse** cuando se van a usar recursos compartidos.
- Los procesos deben ser ellos mismos los que se encarguen de sincronizar sus operaciones, aunque como ayuda a los usuarios los S.O multitarea y algunos lenguajes de programación proporcionan una colección de **primitivas de sincronización** entre procesos

9. Planificación del procesador

10 Comunicación y sincronización entre procesos

- Hay tres formas de interacción entre procesos:
  - **Sincronización** entre procesos: protocolos y mecanismos usados para preservar la integridad y consistencia del sistema.
  - **Señalización** entre procesos: intercambio de señales de sincronización usados para coordinar su progreso colectivo
  - **Comunicación** entre procesos: los procesos se comunican con propósitos tales como intercambiar datos, acumular resultados colectivos, ... (Si lo hacen mediante memoria compartida deben sincronizar sus accesos a dicha memoria compartida).
- La **concurrencia** da lugar a un **aumento de la productividad si se implementa bien**, pero puede degradar la fiabilidad cuando el sistema se contamina con sincronizaciones indeseadas.