10 – Planificación

1. Introducción

La planificación consiste en buscar una forma de que un conjunto de procesos se ejecute cumpliendo sus tiempos límites. Para ello, existen distintas formas de ejecutar los procesos involucrados en un programa concurrente, aunque las salidas idénticas en todos los casos (el comportamiento del sistema no debe verse influido), pero sí hay una variación en el comportamiento temporal en función de la organización llevada a cabo.

Esta organización hace uso de **esquemas de planificación** compuestos por:

- Algoritmo de ordenación del uso de los recursos (CPU): permite determinar el orden en que se ejecutan los procesos optimizando los recursos.
- Un modelo que permite predecir el comportamiento del sistema en el caso peor: comprobar el cumplimiento de la organización en función de los requisitos temporales de cada proceso.
- Los esquemas pueden ser **estáticos**, si las predicciones de la organización se realizan antes de empezar la ejecución, o **dinámicos**, cuando se toman las decisiones relativas a la planificación en tiempo de ejecución.

2. Modelo de proceso simple

Este modelo predice el comportamiento de un sistema de procesos en el peor caso de manera sencilla. Para poder aplicar este modelo, deben cumplirse una serie de supuestos:

- La planificación está compuesta por un conjunto fijo de procesos.
- Los procesos son periódicos, con periodo conocido (T).
- Los procesos son independientes entre sí. En este contexto aparece un instante crítico cuando todos los procesos son ejecutados a la vez.
- Las sobrecargas del sistema, cambios de contexto y demás se suponen con coste cero.
- Los tiempos límites de los procesos son iguales a sus periodos.
- Los procesos tienen tiempo de ejecución constante en el peor de los casos.

La notación de los tiempos que se utiliza en la planificación se realiza de la siguiente manera:

B	Tiempo de bloqueo del proceso en el peor caso
C	Tiempo de ejecución del proceso en el peor caso (WCET)
D	Tiempo límite del proceso
I	Tiempo de interferencia del proceso
J	Fluctuación en la ejecución del proceso
N	Número de procesos en el sistema
P	Prioridad asignada al proceso
R	Tiempo de respuesta del proceso en el peor caso
T	Tiempo mínimo entre ejecuciones del proceso (periodo)
U	Utilización de cada proceso (se define como C/T)
a-z	Nombre del proceso

3. El enfoque de ejecución cíclico

Si los procesos que se van a planificar son fijos y en un número constante y conocido, una forma común de la implementación de sistemas de tiempo real es el uso de un enfoque ejecutivo cíclico. Puede resumirse como una tabla de llamadas a procedimientos que consta de un ciclo principal y de varios ciclos secundarios de duración fija. Durante la ejecución, una interrupción de reloj va marcando los cambios de uno a otro de los ciclos secundarios.

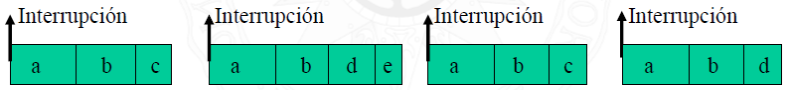
Este enfoque permite organizar la ejecución de los procesos en base a estos dos ciclos y la duración del ciclo secundario, por lo que se trata de una estrategia determinista.

Ejemplo:

Proceso	Periodo, T	Tiempo de ejecución, C
a	25	10
b	25	8
c	50	5
d	50	4
e	100	2

```
loop
  wait_for_interrupt;
  procedure_for_a; procedure_for_b; procedure_for_c;
  wait_for_interrupt;
  procedure_for_a; procedure_for_b; procedure_for_d;
  procedure_for_e;
  wait_for_interrupt;
  procedure_for_a; procedure_for_b; procedure_for_c;
  wait_for_interrupt;
  procedure_for_a; procedure_for_b; procedure_for_d;
end loop;
```

Se toma un intervalo de 25 ms para el ciclo secundario, que corresponde al menor periodo de los procesos que se tienen (A y B). En el primer ciclo secundario sobran 3 ms. En el segundo, se vuelven a ejecutar A y B y como no se llega al periodo de 50 ms, no se vuelve a ejecutar C sino D y E, sobrando 1 ms en ese ciclo secundario. En la tercera interrupción, se vuelven a ejecutar A y B (como tienen periodo igual al de la interrupción, se ejecutarán siempre) y vuelve a ejecutarse C al ser su periodo 50 ms. En la última interrupción se ejecutan A y B y D. Así el conjunto de procesos se puede ir ejecutando, cumpliendo cada uno su tiempo de ejecución y sus periodos.



Las características más importantes del enfoque de ejecución cíclico se pueden resumir en:

- Los procesos realmente no existen en tiempo de ejecución: cada ciclo secundario es una secuencia de llamadas a procedimientos, por lo que solamente existe el proceso principal.
- Los procedimientos comparten un espacio de direcciones: pueden compartir datos sin necesidad de protección.
- Los periodos deben ser múltiplos del tiempo de ciclo secundario.

Inconvenientes

- Dificultad para incorporar procesos esporádicos o a periódicos.
- Dificultad para incorporar procesos con periodos grandes, ya que el tiempo del ciclo mayor es el único que se puede usar sin replanificación.
- Dificultad para construir el ejecutivo cíclico, no es tarea sencilla.
- Procesos con tiempo notable tendrán que ser divididos en procedimientos de tamaño fijo, lo que es susceptible de llevar a errores.

A pesar de estos inconvenientes, si se tiene un conjunto de procesos para los cuales se puede plantear un enfoque de ejecución cíclico y conseguir una planificación con este mecanismo, no es necesario diseñar ningún test de planificabilidad.

4. Planificación basada en tareas

Este enfoque aporta como elemento diferenciador al ejecutivo cíclico que es un proceso más general y soporta la ejecución de procesos de forma directa y en cada instante se determina cuál es el proceso que deba ejecutarse. Para ello, se define un estado para cada proceso y en cada instante de tiempo, cada proceso estará en uno de esos estados:

- Ejecutable.
- Suspendido en espera de un evento temporizado.
- Suspendido en espera de un evento no temporizado.

De esta manera, admite la planificación de procesos esporádicos. Aunque existen diversos tipos de planificación basada en tareas, hay dos principales:

- Planificación de prioridad estática o fija (FPS).
- Primero el tiempo límite más temprano (EDF).

5. Planificación de prioridad fija (FPS)

Este esquema permite asignar una prioridad a cada proceso y se decide una actuación cuando un proceso de alta prioridad tenga que ser ejecutado mientras se está ejecutando otro de menor prioridad. En ese caso, se realiza una apropiación por derecho preferente que puede ser de distintas formas:

- **Esquema apropiativo:** se realiza un cambio de proceso inmediato. El proceso de mayor prioridad desaloja al que se estaba ejecutando, de menor prioridad.
- **Esquema no apropiativo:** el proceso de mayor prioridad espera a que finalice el de menor prioridad.
- **Esquema de apropiación diferida o de distribución cooperativa:** se establece un tiempo limitado para que el proceso de menor prioridad prosiga su ejecución y, finalizado este tiempo, se realiza el cambio antes de ser desalojado por el proceso de mayor prioridad.

A cada proceso se le asigna una prioridad en función de su periodo, por lo que se consigue dar mayor prioridad a los procesos con menor periodo (la prioridad 1 es la menor). Esta asignación se conoce como de **tasa monotónica**.

Proceso	Periodo, T	Prioridad, P
a	25	5
b	60	3
c	42	4
d	105	1
e	75	2

6. Planificación de primero el tiempo límite más temprano (EDF)

Este esquema de planificación se basa en el uso de prioridades para organizar la ejecución, pero se priorizar la ejecución de procesos con tiempo límite más cercano (D). Dado que los tiempos límite absolutos se calculan en tiempos de ejecución, se trata de un esquema dinámico (las prioridades van cambiando de forma dinámica). Se pueden usar dos tipos de tiempo límite:

- **Tiempo límite relativo**: tomado desde el comienzo de la ejecución del proceso.
- **Tiempo límite absoluto**: tomado desde el momento actual.

7. Test de planificabilidad basada en la utilización

Para cada uno de los esquemas de planificación se puede aplicar un **test de planificabilidad** para evaluar si el conjunto de procesos es planificable siguiendo cada uno de los esquemas

Planificación de prioridad fija FPS

Se realiza el siguiente cálculo en base a la utilización de cada proceso (Ci/Ti). Si se cumple esta condición, los N procesos cumplen sus tiempos límite y el sistema de procesos es planificable. Se trata de un esquema apropiativo basado en prioridades y que usa un algoritmo de tasa monotónica.

$$U \equiv \sum_{i=1}^{N} \frac{C_i}{T_i} \leq N(2^{1/N} - 1)$$

N	Límite de utilización
1	100.0%
2	82.8%
3	78.0%
4	75.7%
5	74.3%
10	71.8%

Como el test depende del número de procesos, si este es grande, el valor de U ha de ser menor de 0,69. $U \leq 0.69 \text{ cuando } N \to \infty$

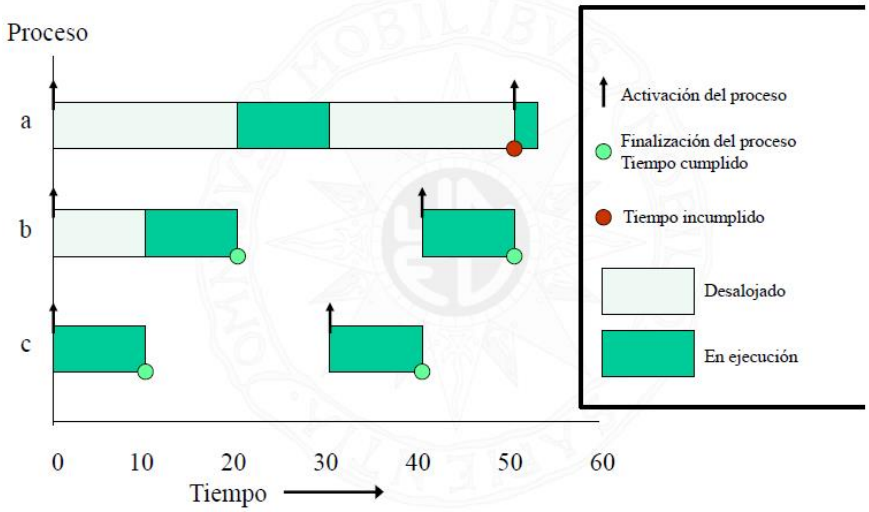
Existen así algunos límites de utilización principales para conjuntos de procesos dependiendo del valor de N.

Ejemplo A:

Proceso	Periodo T	Tiempo ejecución C	Prioridad P	Utilización U
a	50	12	1	0.24
b	40	10	2	0.25
c	30	10	3	0.33

- **Utilización combinada: 0,82 (82%) -> No pasa el test para N = 3**

Una representación gráfica de lo que ocurriría en este caso es:



Ejemplo B:

- **Utilización combinada: 0,775 (77%) < 78% → Pasa el test**
- **Se cumplirán todos los tiempos límites**

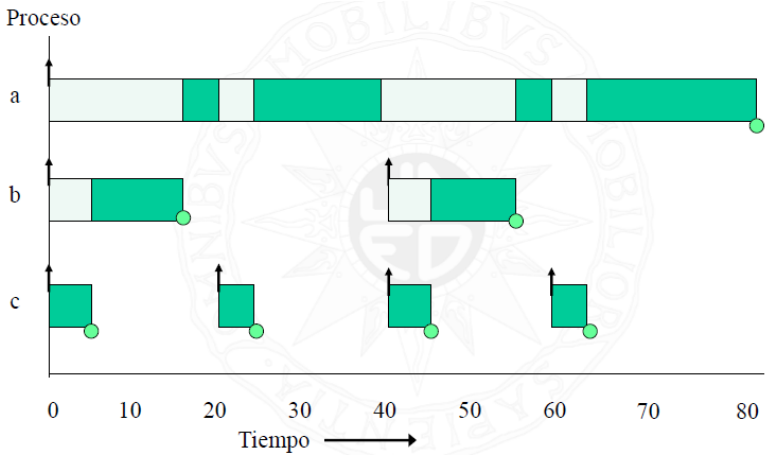
Proceso	Periodo T	Tiempo ejecución C	Prioridad P	Utilización U
a	80	32	1	0.400
b	40	5	2	0.125
c	16	4	3	0.250

Ejemplo C:

No pasar el test no supone que no sea planificable. Existen casos en los que el test no se satisfaga, pero el conjunto de procesos sea planificable.

- **Utilización combinada: 1,0 (100%) → Por encima del límite**
- **No pasa el test**
- **Se cumplen todos los tiempos límites**
- **El test supone una condición suficiente pero no necesaria**

Proceso	Periodo T	Tiempo ejecución C	Prioridad P	Utilización U
a	80	40	1	0.50
b	40	10	2	0.25
c	20	5	3	0.25



Planificación de primero el más temprano EDF

$$\sum_{i=1}^{N} \frac{C_i}{T_i} \leq 1$$

Este test es parecido al de FPS, pero es más sencillo. Si el conjunto de procesos requiere de una utilización menor o igual a la unidad (es decir, no mayor que la capacidad total del procesador), los tiempos límite de dichos procesos se cumplirán.

De este modo, es fácil entender que el esquema de planificación EDF es superior al de FPS, pues es más fácil cumplir el requisito del test de planificabilidad y, por tanto, es más probable que los procesos de un conjunto dado puedan cumplir todos sus tiempos límite.

Alternativas al test de planificabilidad para FPS

El test de utilización para FPS presenta algunos inconvenientes:

- No es exacto.
- No aplicables a modelos más generales.
- Representa una condición suficiente pero no necesaria.

Esta alternativa de test está basada en el tiempo de respuesta y consiste en predecir el tiempo de respuesta en el peor caso para cada proceso (calculando dicho tiempo respuesta), comparar dicho tiempo con el tiempo límite de cada uno y, si los tiempos de respuesta calculados de todos los procesos son inferiores a sus límites, el conjunto de procesos es planificable.

El proceso de mayor prioridad tiene un tiempo de respuesta igual al de su ejecución (Ri=Ci) y no va a sufrir ninguna interferencia, mientras que el resto de los procesos pueden sufrir algunas interferencias que se suman a su tiempo de ejecución (Ri=Ci+Ii). Sólo interferirán en un proceso aquellos que tengan más prioridad que él El valor máximo de la interferencia viene dado por:

$$\sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Donde Tj es el periodo de los procesos j con mayor prioridad de i. el tiempo de respuesta para un proceso i queda como:

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Para resolver esta ecuación se utiliza una ecuación de concurrencia dada por:

$$w_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil C_j$$

Para cada proceso se calculan los valores wi^n que forman una sucesión monótona no decreciente. Si en una iteración concreta se tiene que:

- Si $wi^n = wi^{n+1}$, entonces $Ri = wi^n$. Dos valores, elementos de dicha sucesión tienen igual valor.
- Si $wi^n > Ti$, entonces el proceso no cumplirá su tiempo límite.

El primer término de la sucesión se calcula teniendo en cuenta sólo el tiempo de ejecución de los procesos más prioritarios.

$$w_i^1 = C_i + \sum_{j \in hp(i)} C_j$$

Ejemplo D:

Proceso	Periodo T	Tiempo ejecución C	Prioridad P	Utilización U
a	80	40	1	0.50
b	40	10	2	0.25
c	20	5	3	0.25

Para el proceso C, Rc = 5, al ser el más prioritario. Para el resto de los procesos se calculan los términos de la sucesión. Para el proceso B, se llega a que el segundo término y el primero son iguales, por tanto, dicho valor es su tiempo de respuesta.

$$w_b^1 = C_b + C_c = 15$$

$$w_b^2 = C_b + \left\lceil \frac{w_b^1}{T_c} \right\rceil C_c = 10 + \left\lceil \frac{15}{20} \right\rceil 5 = 15$$

Para el proceso A ocurre lo mismo en los términos 3 y 4:

$$w_a^1 = C_a + C_c + C_b = 55$$

$$w_a^2 = C_a + \left\lceil \frac{w_a^1}{T_c} \right\rceil C_c + \left\lceil \frac{w_a^1}{T_b} \right\rceil C_b = 40 + \left\lceil \frac{55}{20} \right\rceil 5 + \left\lceil \frac{55}{40} \right\rceil 10 = 75$$

$$w_a^3 = C_a + \left\lceil \frac{w_a^2}{T_c} \right\rceil C_c + \left\lceil \frac{w_a^2}{T_b} \right\rceil C_b = 40 + \left\lceil \frac{75}{20} \right\rceil 5 + \left\lceil \frac{75}{40} \right\rceil 10 = 80$$

$$w_a^4 = C_a + \left\lceil \frac{w_a^4}{T_c} \right\rceil C_c + \left\lceil \frac{w_a^4}{T_b} \right\rceil C_b = 40 + \left\lceil \frac{80}{20} \right\rceil 5 + \left\lceil \frac{80}{40} \right\rceil 10 = 80$$

El análisis del tiempo de respuesta muestra así que todos los procesos cumplen sus *deadline* y el conjunto de procesos es planificable.

Proceso	Periodo T	Tiempo ejecución C	Prioridad P	Utilización U	
a	80	40	1	0.50	$R_a = 80$
b	40	10	2	0.25	$R_b = 15$
c	20	5	3	0.25	$R_c = 5$

Este análisis del tiempo de respuesta sí es suficiente y necesaria. Un conjunto de procesos que no cumpla este test no será planificable.

8. Comparación de planificadores FPS y EDF

Con el esquema de planificación EDF no es sencillo obtener el tiempo de respuesta en el peor caso para todos los procesos por lo que el test de análisis basado en el tiempo de respuesta no se suele aplicar a este planificador. Esto se debe a que, en dicha situación, sólo los procesos con los tiempos límite relativos más pequeños interferirían en la ejecución de otros procesos, pero, después, puede producirse una situación en la que la gran mayoría de los procesos (o incluso todos) tengan un tiempo límite absoluto más corto. Por tanto, calcular el término I puede ser muy complejo en este caso.

Las principales ventajas del esquema FPS basado en prioridades fijas respecto al de EDF dinámico son:

- FPS es más sencillo de implementar, así como a la hora de incorporar procesos sin tiempos límites.
- FPS es más predecible en situaciones de sobrecargas.
- Es más fácil trabajar con prioridades fijas (FPS) que basadas en deadlines (EDF).
- Aunque el test de planificación basado en la utilización es más fácil de cumplir para sistemas EDF, en sistemas con FPS es posible que un conjunto de procesos no supere el test, pero sin embargo sí se consigan mayores utilizaciones de las que el test informa. Se puede paliar esta diferencia con FPS haciendo uso del test de respuesta.
- EDF no proporciona el peor caso del tiempo de interferencia ni el de respuesta, por lo que no se pueden usar los test basados en respuesta.

9. Tiempo de ejecución en el peor caso

Este aspecto es fundamental porque la mayoría de los esquemas de planificación dan por hecho que se conoce este tiempo para poder hacer el análisis de planificabilidad. El **tiempo de ejecución en el peor caso** (WCET) consiste en obtener C por análisis o medición.

- **Medición**: realizando una observación experimental en función de los distintos casos.
- **Análisis**: se necesita disponer de un modelo efectivo del procesador y consisten en descomponer primero el código de cada proceso en un grafo dirigido de bloques básicos, para posteriormente centrarse en utilizar el modelo disponible del procesador que se vaya a utilizar para estimar el tiempo de ejecución en el peor caso. Los procesadores más modernos incluyen mecanismos para optimizar el tiempo de ejecución, lo que hace más difícil el análisis

Ejemplo de cálculo mediante análisis:

```
for(int i=0; i<10; i++){
   if (condicion){
      // Bloque básico con coste 5
   }
   else {
      // Bloque básico con coste 20
   }
}
```

❖ **Suponiendo el coste del bucle y la condición igual a 2**

- 10x20 + 2 + 2 = 204 unidades de tiempo

❖ **Si se dispone de la frecuencia de cumplimiento (3 de cada 5)**

- 10 x (3/5 x 5 + 2/5 x 20) + 2 + 2 = 115 unidades de tiempo

Cuanta más información se tenga de los casos de ejecución, más acertado y afinado será el cálculo que se haga. Las estrategias de análisis dependerán ello. También puede ocurrir que cuanto más se quiera afinar, se caiga en imprecisiones que hagan que el resultado de coste sea menor que el que se pueda obtener en el peor caso, por lo que hay que encontrar un punto medio y evitar estas situaciones.

10. Procesos esporádicos y aperiódicos

A diferencia de los procesos simples, es posible incluir variaciones para poder considerar procesos que presentan un comportamiento temporal aperiódico y/o esporádico. Para ello se utiliza un modelo distinto, con las siguientes características:

- Redefinir el valor T. Si antes era entendido como el periodo de un proceso, ahora es el valor mínimo del intervalo entre ejecuciones.
- En el modelo simple se considera que el límite de un proceso y su periodo eran iguales (D = T), en este modelo se permite también que D < T. Esto permite considerar aperiódicos como los de rutinas de gestiones de error que son por lo general urgentes y críticas y con un tiempo de ejecución muy pequeño.

La tasa de llegada de los procesos, en el caso peor, es superior a la media. Por tanto, se establecen dos reglas para poder encontrar los requisitos mínimos de planificabilidad:

- **Todos los procesos deben ser planificables** utilizando los tiempos y las tasas medias de llegadas. No obstante, pueden existir situaciones en las que no sea posible cumplir los tiempos límite (**sobrecarga transitoria**).
- **Todos los procesos estrictos deben ser planificables** utilizando los peores casos del tiempo de ejecución y de la tasa de llegada de todos los procesos, incluyendo los flexibles. Se asegura así que ningún proceso estricto incumplirá su límite y que se podrían obtener utilizaciones inaceptablemente bajas.

Procesos aperiódicos y servidores FPS

Existe la posibilidad de planificar la ejecución de los procesos asignando prioridades mayores a los procesos estrictos que a los flexibles o aperiódicos. De este modo, se asegura la ejecución de los procesos estrictos, pero sin embargo los procesos flexibles incumplirían frecuentemente sus tiempos.

Para evitar esto, se hace uso de servidores que permiten que los procesos flexibles se ejecuten lo antes posible y hacen todo lo posible para añadir procesos nuevos con máxima prioridad. Existen varios métodos para estos servidores como el uso de **servidores diferibles** (DS) o **servidores esporádicos** (SS) que limitan la capacidad para los procesos flexibles y aperiódicos para que no se ejecuten más de lo que debieran.

Otra opción es hacer uso de **planificadores de prioridad dual** en los que se definen tres rangos de prioridades para los procesos (alta, media y baja). Todos los procesos aperiódicos se ejecutan en la banda media y los procesos estrictos se mueven entre las bandas baja y alta, en función de sus tiempos límite.

Procesos aperiódicos y servidores EDF

En EDF se calcula el tiempo límite cada vez que un proceso se va a ejecutar al ser un sistema dinámico. Así, en este caso el algoritmo asigna al servidor el tiempo límite más pequeño si, y sólo si, existe un proceso aperiódico pendiente de servido (si existe capacidad suficiente de procesamiento). Una vez que la capacidad de procesamiento se agota, el servidor queda en suspenso hasta que la capacidad fuera repuesta.

11. Sistemas de procesos con D < T

En los procesos aperiódicos donde el deadline (D) es igual al valor de T, la ordenación de prioridad de tasa monotónica resulta óptima. Para los casos en que D < T, es posible definir una formulación similar mediante DMPO (Deadline Monotonic Priority Ordering), donde si Di < Dj entonces Pi > Pj, esto es, un proceso es más prioritario que otro si su deadline es menor.

DMPO es óptima si cualquier conjunto de procesos Q, planificable por el esquema de prioridades W, también lo es mediante DMPO. La prueba de optimalidad implica transformar las prioridades de Q hasta obtener una ordenación DMPO (cada paso de la transformación preservará la planificabilidad).

Ejemplo:

Proceso	Periodo T	Tiempo límite D	Tiempo ejecución C	Prioridad P	Tiempo respuesta R
a	20	**5**	3	**4**	3
b	15	**7**	3	**3**	6
c	10	**10**	4	**2**	10
d	20	**20**	3	**1**	20

12. Interacciones y bloqueos entre procesos

Es necesario eliminar en los esquemas de planificabilidad que los procesos no sean independientes, ya que es frecuente encontrar situaciones en las que los procesos tienen interacciones entre sí.

La **inversión de prioridad** ocurre cuando un proceso suspendido está a la espera de que otro proceso de menor prioridad realice algún cálculo, por lo que el proceso de menor prioridad ha invertido la prioridad con respecto al de mayor prioridad. El proceso de mayor prioridad se dice que está **bloqueado**. Este bloqueo debe estar limitado y ser ponderable.

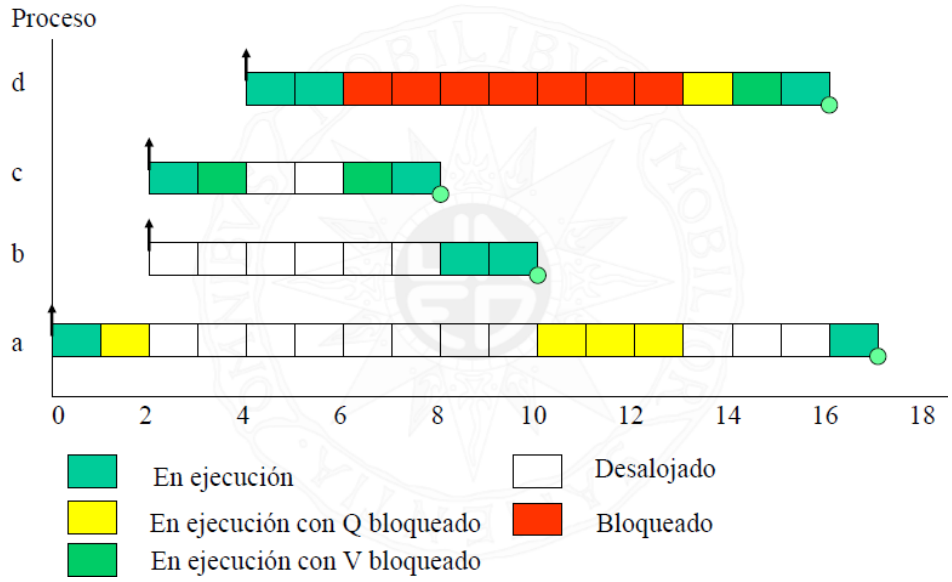
Ejemplo:

Proceso	Prioridad	Secuencia ejecución	Instante de activación
a	1	EQQQQE	0
b	2	EE	2
c	3	EVVE	2
d	4	EEQVE	4

E -> el proceso está en ejecución.

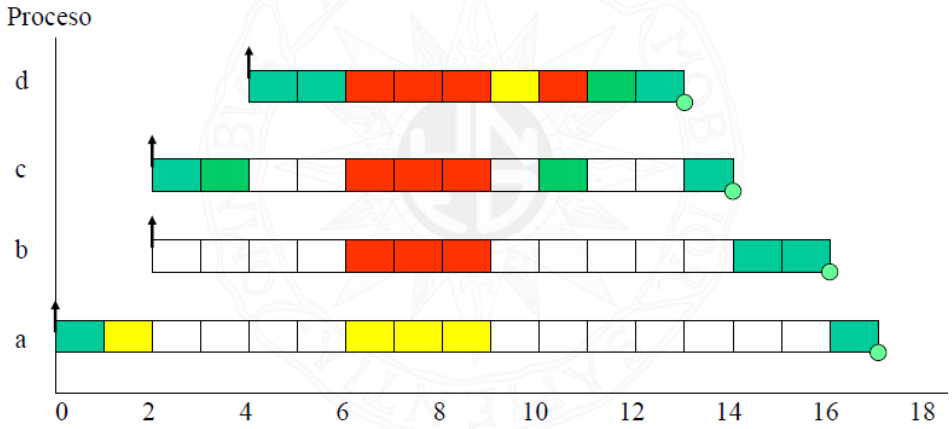
Q y V-> el proceso está en ejecución accediendo a una sección crítica o de uso compartido.

De esta forma, el proceso puede estar en distintos estados.



Existe una forma de resolver estas situaciones de bloqueo mediante **herencia de prioridad**, donde la prioridad de un proceso se modifica y será el máximo entre su prioridad y las prioridades de los procesos que dependan de él.

Ejemplo (mismo que antes):



En el instante 6, el proceso A hace uso de la prioridad de D para terminar de usar el recurso Q y liberarlo en el instante 9. En el instante 10, D se ve bloqueado de nuevo al intentar acceder al recurso V bloqueado por C. C hereda la prioridad de D para terminar su ejecución y liberar el recurso para que D pueda continuar su ejecución en el instante 11. De esta forma, aunque los procesos finalizan en el mismo instante, el proceso D, más prioritario, ha sido el primero en finalizar su ejecución.

Tiempo de bloqueo

En estas situaciones de bloqueo hay que tener en cuenta el tiempo de bloqueo que puede producirse en el retraso de la ejecución de un proceso. Si un proceso tiene m secciones críticas que puedan provocar su bloqueo, el máximo número de veces que puede ser bloqueado es m.

A partir de este valor, se puede calcular el tiempo máximo de bloqueo (B) siendo y k el número de secciones críticas, el proceso i tiene un límite superior de bloqueo dado por:

$$B_i = \sum_{k=1}^{K} utilización(k,i)C(k)$$

El sumatorio de la utilización de cada una de las secciones críticas por el tiempo de ejecución. El valor de utilización(k,i) es igual a 1 si el recurso k se utiliza al menos por un proceso con una prioridad menor que la del proceso i, y al menos por un proceso con prioridad mayor o igual a la de i. En otro caso valdrá 0.

A partir de este tiempo de bloqueo, se pueden actualizar los tiempos de respuesta de un proceso teniendo en cuenta el factor de bloqueo:

$$R_i = C_i + B_i + I_i$$

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

La ecuación de concurrencia se resuelve ahora a partir de:

$$w_i^{n+1} = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil C_j$$

13. Protocolos de acotación de la prioridad

Estos protocolos están diseñados para minimizar las situaciones de cadenas de bloqueo e intentar eliminar condiciones de fallo. Para esquemas de prioridad fija FPS hay dos tipos de protocolos

- Protocolo original de acotación de la prioridad (OCPP).
- Protocolo inmediato de acotación de la prioridad (ICPP).

Cuando se utiliza cualquiera de estos dos protocolos en un sistema monoprocesador, se puede decir que:

- Un proceso de alta prioridad puede ser bloqueado por procesos de prioridad baja en una sola ocasión como máximo.
- Se previenen los bloqueos mutuos.
- Se previenen los bloqueos transitivos.
- Se aseguran los accesos mutuamente excluyentes a recursos.

OCPP: Protocolo original de acotación de la prioridad

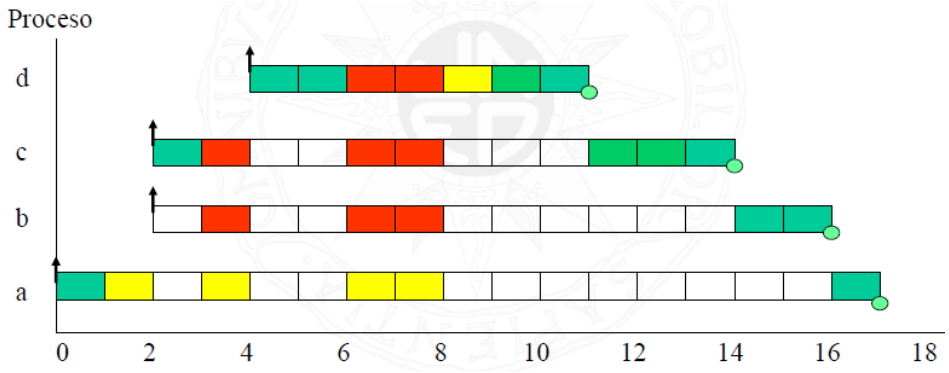
En este protocolo:

- Cada proceso tiene una prioridad estática por defecto.
- Cada recurso tiene definido un valor de cota estático, que es la prioridad máxima de los procesos que la están utilizando.
- Un proceso tiene una prioridad dinámica que se calcula como el máximo de su prioridad estática o de cualquiera de las que herede debido a bloqueos.
- Un proceso solo puede bloquear un recurso si su prioridad dinámica es mayor que la cota máxima de cualquier recurso actualmente bloqueado (excluyendo a los que estén bloqueados por él).

Con estas características, el tiempo máximo que un proceso puede estar bloqueado se puede redefinir como el máximo de sus valores de utilización, no el sumatorio de los mismos:

$$B_i = \max_{k=1}^{k} utilización(k,i)C(k)$$

Ejemplo (mismo que antes):



En el instante 3, cuando C quiere acceder al recurso V, no puede hacerlo porque A ya tenía un recurso bloqueado, por lo que éste puede continuar su ejecución. Lo mismo se repite en el instante 6 cuando D quiere acceder al recurso que tiene A bloqueado.

La diferencia está en que como C no ha llegado a bloquear el recurso V, el proceso D puede terminar su ejecución en el instante 11.

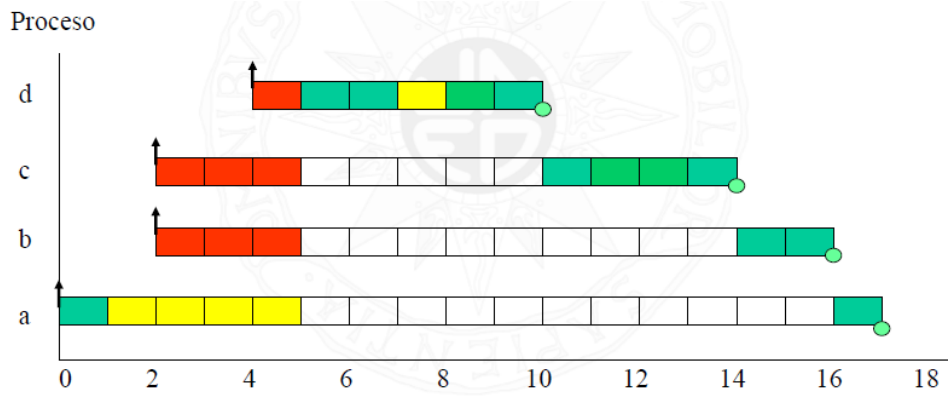
ICPP: Protocolo inmediato de acotación de la prioridad

Este protocolo presenta un enfoque más simple:

- Cada proceso tiene asignada una prioridad estática por defecto.
- Cada recurso tiene definido un valor cota estático que se corresponde con la prioridad máxima de los procesos que lo utilizan.
- Un proceso tiene también una prioridad dinámica, que es el máximo entre su prioridad y los valores techo de cualquier recurso que tenga bloqueado. Un proceso solamente podrá ser bloqueado únicamente al principio de su ejecución. Cuando el proceso comience, todos los recursos deben estar libres. Si no es así, algún proceso tendrá una prioridad mayor o igual, y la ejecución del proceso será postpuesta por defecto.

En RT-Java, este protocolo está implementado mediante la clase `Priority Ceiling Emulation`.

Ejemplo (mismo que antes):



El proceso A no suelta la ejecución del bloque con el recurso Q hasta que no la finaliza. El proceso D retrasa su inicio de ejecución al instante 5 cuando A libera los recursos, incluso siendo de mayor prioridad. Finalmente, termina su ejecución en el instante 10, 1 antes que con OCPP.

Así, los procesos tienden a completarse una vez comienzan su ejecución.

OCPP vs ICPP

- El comportamiento en el peor caso es idéntico
- ICPP produce menos cambios de contexto: el bloqueo es previo a la ejecución.
- ICPP requiere más cambios de prioridad: se producen con todas las utilizaciones de recursos y además OCPP sólo modifica la prioridad si se producen bloqueos.
- ICPP es más sencillo de implementar: no hay que monitorizar tantas relaciones de bloqueo.

Bloqueos y EDF

Para los esquemas de planificación EDF, los protocolos de acotación de la prioridad son más complejos debido a las relaciones de prioridades dinámicas. Se usa una **política de pila de recursos** (SRP) que funciona de forma similar a ICPP para FPS. Las características de esta política son:

- Se asigna un nivel de apropiación a cada proceso en base a los tiempos límite relativos de los procesos de forma que cuanto más cortos sean estos tiempos límites, mayor será el nivel de del proceso.
- Se establece una cota para cada recurso en tiempo de ejecución basada en el máximo nivel de apropiación de los procesos que lo usan.

- Se tiene así una regla que cuando un proceso se activa, sólo puede desalojar al proceso en ejecución si su tiempo límite absoluto es más corto y su nivel de apropiación es mayor que la cota más alta de los recursos bloqueados.

Con este protocolo, los procesos sólo tienen un bloqueo (como en ICPP) que se produce al ser activados. Se previenen así los bloqueos mutuos facilitando dar una fórmula sencilla para el tiempo de bloqueo.

14. Un modelo de proceso extensible

Hasta ahora, las ampliaciones sobre el modelo de planificación simple incluían:

- Los tiempos límite pueden ser menores que el periodo (D < T).
- Se pueden soportar procesos esporádicos y aperiódicos.
- Se contemplan las interacciones entre procesos que causarán bloqueo y los bloqueos se tienen en cuenta en los tiempos de respuesta.

Se pueden añadir nuevas generalizaciones, especialmente para FPS.

Planificación cooperativa

La planificación cooperativa hace uso de la **apropiación diferida**, que permite que se ejecuten procesos de menor prioridad dando lugar a situaciones de bloqueo causadas por el uso de datos compartidos (exclusión mutua), por el sistema de ejecución o por el núcleo del sistema operativo.

La planificación cooperativa consiste en dividir el código en bloques no desalojables, calculando un tiempo de ejecución limitado por un valor Bmax y al final de cada bloque se realiza una petición al núcleo del sistema para "desplanificar". En este contexto, si un proceso de alta prioridad está en estado ejecutable, se hará un cambio de contexto.

Se utiliza la planificación cooperativa porque aporta una serie de ventajas:

- Incrementa la planificación del sistema, ya que puede conducir a valores menores del tiempo de ejecución C.
- No presenta interferencias en el último bloque no desalojable (tiempo de ejecución del último bloque, Fi).
- Presenta una mayor precisión en la predicción de los tiempos de ejecución de los bloques no desalojables.

En este contexto, el tiempo de respuesta del último bloque se representa con la ecuación:

$$w_i^{n+1} = B_{MAX} + C_i - F_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil C_j$$

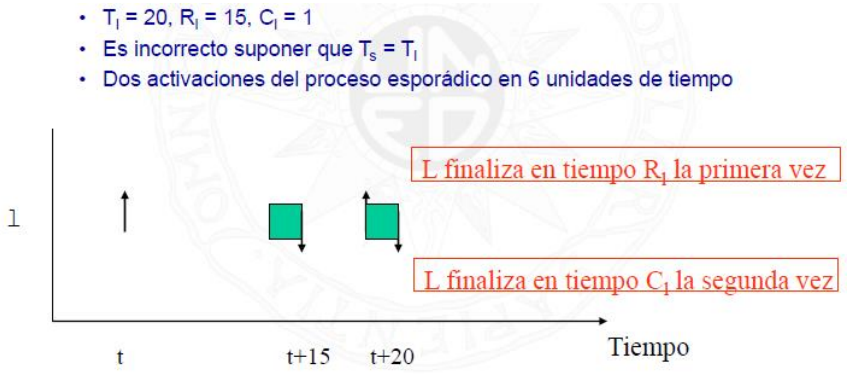
Y cuando converge, el tiempo de respuesta del último bloque queda:

$$R_i = w_i^n + F_i$$

Fluctuaciones en la activación

En el modelo simple se supone un tiempo de llegada para procesos esporádicos igual a T, pero este caso no es siempre realista.

Ejemplo: un proceso esporádico s es activado por un proceso periódico l (de periodo Tl) en otro procesador.



En el primer periodo, L no puede ejecutarse por haber una interferencia y tiene que hacerlo en t+15, mientras que, en el segundo periodo, no hay interferencia y puede acabar en t+20. De esta forma, las dos activaciones distan en 6 unidades y no las 20 del modelo simple.

De esta manera, hay que modificar las ecuaciones de recurrencia de estudio del tiempo para poder incluir el efecto de las fluctuaciones en la activación. Esta **fluctuación** o **jitter** (J) es el valor máximo de la variación en la activación de un proceso (en el ejemplo, sería J = 15). La activación se podría producir así en los instantes 0, T-J, 2T-J,3T-J, etc.

Un proceso i las interferencias de s tal que:

- Una interferencia si $R_i \in [0, T-J)$
- Dos interferencias si $R_i \in [T-J, 2T-J)$
- Tres interferencias si $R_i \in [2T-J, 3T-J)$
- …

El tiempo de respuesta queda entonces:

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i + J_j}{T_j} \right\rceil C_j$$

Además, el tiempo de respuesta relativo al tiempo de respuesta real se define como:

$$R_i^{periodic} = R_i + J_i$$

Tiempos límite arbitrarios

Es posible que, debido a las fluctuaciones, los deadline (D) y los tiempos de respuesta (R) sean mayores a su periodo. En este caso se puede considerar que existen varias activaciones en un mismo periodo y para cada activación solapada se define una ventana w(q), donde q representa las distintas activaciones (0, 1, 2, …). Por tanto, la ecuación de recurrencia queda ampliada a:

$$w_i^{n+1}(q) = B_i + (q+1)C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n(q)}{T_j} \right\rceil C_j$$

El tiempo de respuesta viene dado ahora por:

$$R_i(q) = w_i^n(q) - qT_i$$

Por ejemplo, q = 2 implica que habrá tres activaciones en la misma ventana (0, 1 y 2) y si el proceso comenzó en 2Ti, el tiempo de respuesta será el tamaño de la ventana menos 2Ti.

El número de activaciones está por tanto limitado por el valor más pequeño de q para el que se cumpla:

$$R_i(q) \leq T_i$$

El tiempo de respuesta en el peor caso se calcula como el máximo valor para cada q:

$$R_i = \max_{q=0,1,2,\ldots} R_i(q)$$

Combinando este efecto con el de la fluctuación, la ecuación de recurrencia queda:

$$w_i^{n+1}(q) = B_i + (q+1)C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n(q) + J_j}{T_j} \right\rceil C_j$$

En este contexto, dos ventanas pueden solaparse si el tiempo de respuesta más la activación supera el periodo. En este caso, el tiempo de respuesta será:

$$R_i(q) = w_i^n(q) - qT_i + J_i$$

Tolerancia a fallos

La tolerancia a fallos también ha de ser tenida en cuenta, ya que puede implicar un consumo extra de tiempo que repercute en los cálculos del tiempo de respuesta, ya sea esta tolerancia hacia delante o hacia atrás. Con esta reformulación y añadiendo tiempos extra hay que continuar asegurando que los tiempos límite de los procesos se sigan cumpliendo.

Para este modelo de fallos se incorpora un nuevo término Ci^f que representa el tiempo extra que resulta del error, por lo que el tiempo de respuesta queda:

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j + \max_{k \in hep(i)} FC_k^f$$

Donde hep(i) representa los procesos con mayor o igual prioridad y siendo F el número máximo de fallos permitidos. Es posible indicar un intervalo de llegada mínimo para fallos (Tf) que se puede incluir en la ecuación:

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j + \max_{k \in hep(i)} \left(\left\lceil \frac{R_i}{T_f} \right\rceil C_k^f \right)$$

Introducción de desplazamientos

Las estrategias para incluir los desplazamientos buscan evitar los instantes críticos (cuando se activan todos los procesos a la vez, instante 0). Eso puede hacer que alguno de los procesos no llegue a su tiempo de respuesta (fuera de su límite).

Ejemplo:

- Ej: el proceso c tiene un tiempo de respuesta fuera de su límite

Process	T	D	C	R
a	8	5	4	4
b	20	10	4	8
c	20	12	4	16

Se ejecuta primero el proceso A agotando 4 unidades de tiempo y a continuación B con otras 4. Ambos cumplirían sus tiempos límite. Al volver al instante 8, volvería a ejecutarse A y luego B y el proceso C quedaría relegado, que no cumpliría su deadline.

Para paliar este efecto, se añade un desplazamiento (offset) sobre el proceso C, añadiendo un tiempo de activación más tardío para que así pueda cumplir su deadline.

- El proceso c recibe un desplazamiento (O) de 10
  - El conjunto de procesos resultante es planificable

Process	T	D	C	O	R
a	8	5	4	0	4
b	20	10	4	0	8
c	20	12	4	10	8

El análisis de los instantes críticos en la introducción de los desplazamientos no es una tarea sencilla y elegir los desplazamientos óptimos de un conjunto de procesos es un problema NP-completo. Existe una estrategia que lo soluciona parcialmente y es elegir un desplazamiento de T/2. Así, cuando dos procesos tienen el mismo periodo, a uno de ellos se le asigna un desplazamiento de T/2.

Para analizar el sistema resultante en el instante crítico, se pueden reemplazar ambos procesos por uno artificial de periodo T/2 y sin desplazamiento (más el máximo de los tiempos de ejecución de los dos procesos y el mínimo de los deadline). Si este sistema es planificable, también lo será el original asignando un desplazamiento de T/2 a uno de los procesos.

Process	T	D	C	O	R
a	8	5	4	0	4
n	10	10	4	0	8

Asignación de prioridades

Este teorema permite realizar una asignación automática de prioridades de manera recursiva y satisfacer la planificabilidad de un conjunto de procesos:

Si un proceso p tiene asignada la menor de las prioridades y es realizable, entonces, si existe una ordenación de prioridades realizable para el conjunto completo de procesos, ese orden asignará al proceso p la menor de las prioridades.

```
boolean ok;
for(int i=0; i<n; i++){
   ok = false;
   for (int next=i; next<n; next++){
      intercambia(conjunto, i, next++);
      ok = testProceso(conjunto, i);
      if (ok) break;
   }
   if (!ok) break; //Abortamos procedimiento por imposibilidad de
                   //encontrar proceso planificable
}
```

Se tiene un array que va ordenando los procesos para que sean planificables.

15. Sistemas dinámicos y análisis en línea

El análisis de planificabilidad de un sistema se puede hacer mediante sistemas dinámicos y análisis en línea y se puede realizar de dos maneras:

- **Análisis off-line**: se utiliza para sistemas estrictos ya que requiere que los patrones de llegada del trabajo entrante sean conocidos y limitados, que los tiempos de computación sean limitados y que el esquema de planificación con ejecución predecible. Así, este análisis es el que se lleva a cabo para planificación de prioridad estática.
- **Esquema de planificación en línea**: es más adecuado para sistemas EDF ya que no cumple con alguno de los requisitos anteriores y se puede utilizar para aplicaciones flexibles donde no se conocen los tiempos de llegada ni de computación. La mayoría de estos análisis en línea cuentan con mecanismos de gestión de sobrecargas:
    - Módulo de control de admisión que limita el número de procesos que compiten por los recursos.
    - Rutina de distribución EDF para procesos admitidos en dicha competición.

Para decidir cuáles son los procesos que se admiten o se rechazan, es necesario conocer la importancia relativa de cada proceso, lo que suele hacerse asignando un valor a los mismos. Estos valores pueden clasificarse de alguna de estas tres formas:

- **Estático**: el proceso siempre tiene el mismo valor y requiere de especialistas en el dominio de la aplicación.
- **Dinámico**: el valor del proceso se calcula cuando se activa, pero no mientras se ejecuta.
- **Adaptativo**: el valor del proceso cambia durante su ejecución.

Este tipo de análisis se podrían implementar como un análisis off-line cada vez que llegue un proceso nuevo. Si el sistema no es planificable, se elimina el proceso de menor valor. También se pueden utilizar sistemas híbridos que contienen componentes estrictos y dinámicos.

16. Programación en Java de sistemas basados en prioridad

La programación basada en prioridades es más propia de los SO que de los lenguajes, pero el enfoque dado en los SO no es adecuado para los sistemas en tiempo real. En RT-Java existen dos entidades planificables que soportan la interfaz `Schedulable`:

- `RealtimeThreads` (y `NoHeapRealtimeThread`).
- `AsynEventHandler` (y `BoundAsyncEventHandler`).

La política de despacho que utiliza es apropiativa y basada en prioridades:

- Se exigen al menos 28 niveles de prioridad.
- A mayor valor, mayor prioridad.
- Los hilos que no sean de tiempo real reciben una prioridad por debajo de la mínima.

- Los parámetros de planificación se ligan a los hilos cuando éstos se crean, por lo que la modificación de los parámetros tiene efecto inmediato.

```
public interface Schedulable extends Runnable
{
  public void addToFeasibility();
  public void removeFromFeasibility();

  public MemoryParameters getMemoryParameters();
  public void setMemoryParameters(MemoryParameters memory);

  public ReleaseParameters getReleaseParameters();
  public void setReleaseParameters(ReleaseParameters release);

  public SchedulingParameters getSchedulingParameters();
  public void setSchedulingParameters(
          SchedulingParameters scheduling);

  public Scheduler getScheduler();
  public void setScheduler(Scheduler scheduler);
}

public abstract class SchedulingParameters{
  public SchedulingParameters(); }

public class PriorityParameters extends SchedulingParameters{
  public PriorityParameters(int priority);

  public int getPriority(); // at least 28 priority levels
  public void setPriority(int priority) throws
                          IllegalArgumentException;
  ...
}

public class ImportanceParameters extends PriorityParameters{
  public ImportanceParameters(int priority, int importance);
  public int getImportance();
  public void setImportance(int importance);
  ...
}
```

RT-Java también incorpora un planificador de alto nivel que se encarga de fijar la prioridad de los objetos planificables mediante un algoritmo de asignación de prioridades asociado al algoritmo de realizabilidad. Por otro lado, permite elegir cuándo se admiten nuevos objetos planificables, dependiendo de los recursos disponibles y el algoritmo de factibilidad.

```
public abstract class Scheduler{
  public Scheduler();
  protected abstract void addToFeasibility(Schedulable s);
  protected abstract void removeFromFeasibility(Schedulable s);

  public abstract boolean isFeasible();
  // checks the current set of schedulable objects

  public boolean changeIfFeasible(Schedulable schedulable,
         ReleaseParameters release, MemoryParameters memory);

  public static Scheduler getDefaultScheduler();
  public static void setDefaultScheduler(Scheduler scheduler);

  public abstract String getPolicyName();
}
```

El método `isFeasible()` de la clase `Scheduler` se utiliza para considerar únicamente el conjunto de objetos planificables que se hayan incorporado a la lista de factibilidad. Para añadir o eliminar objetos de esta lista se utilizan los métodos `addToFeasibility()` y `removeFromFeasibility()`.

El método `changeIfFeasible()` se encarga de verificar si el conjunto de objetos es realizable y de si los parámetros de activación y memoria del objeto han cambiado o no. En caso de haberlo hecho, se modifican dichos parámetros. Los dos

métodos estáticos (`getDefaultScheduler()` y `setDefaultScheduler()`) de la clase sirven para interrogar y modificar el planificador por defecto.

A partir de la clase anterior, la clase `PriorityScheduler` implementa la planificación apropiativa basada en prioridad:

```
class PriorityScheduler extends Scheduler
{
  public PriorityScheduler()

  protected void addToFeasibility(Schedulable s);
  ...

  public void fireSchedulable(Schedulable schedulable);

  public int getMaxPriority();
  public int getMinPriority();
  public int getNormPriority();

  public static PriorityScheduler instance();
  ...
}
```

Soporte de herencia de prioridad

RT-Java permite utilizar algoritmos de herencia de prioridad mediante las clases:

- `MonitorControl`: permite establecer las políticas dadas por defecto para un objeto.
- `PriorityCeilingEmulation`: implementa el protocolo inmediato de acotaciones de prioridad.
- `PriorityInheritance`: especifica el protocolo de herencia de prioridad

```
public abstract class MonitorControl {
  public MonitorControl();

  public static void getMonitorControl();
  public static void getMonitorControl(Object obj);
  public static void getMonitorControl(MonitorControl policy);
  public static void getMonitorControl(Object obj,
                                       MonitorControl policy);
}

public class PriorityCeilingEmulation extends MonitorControl{
  public static PriorityCeilingEmulation instance(int ceiling);
  public int getCeiling();
}

public class PriorityInheritance extends MonitorControl{
  public static PriorityInheritance instance();
}
```

```
public class SynchronizeClass{
  public void method1() { … }
  public void method2() { … }
}
```

```
SynchronizeClass sc = new SynchronizeClass();
PriorityCeilingEmulation pce = new PriorityCeilingEmulation(20);
…
MonitorControl.setMonitorControl(sc,pce);
```

Mas opciones en RT-Java

- En Java, las colas para tiempo real están ordenadas por prioridad.
- RT-Java también soporta el uso de hilos aperiódicos: se realiza bajo una fórmula de grupos de procesos donde es posible asociar un grupo de hilos aperiódicos y asignarles algunas características para poder realizar el análisis de factibilidad (por ejemplo, que no consuman más de una cantidad de tiempo de procesador en un tiempo determinado asignando un tiempo máximo de CPU).