Planificación de Procesos

Planificación de procesos. Distinguiremos entre tres niveles o tipos de planificación (a largo, medio y corto plazo). Se explican distintos algoritmos de planificación.

Tabla de Contenidos

Autor: Lina García Cabrera

1 Introducción

Ocultar conocimiento avanzado

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Procesos: Planificación de Procesos e Hilos
La **planificación es básica en un sistema multiprogramado** para que el sistema operativo haga más productiva a la computadora. Consisten en solucionar el problema de **cuándo asignar un procesador y a qué proceso**. Distinguiremos entre tres niveles o tipos de planificación (a largo, medio y corto plazo).

A partir de aquí nos centraremos en la **planificación a corto plazo o de la CPU**. Discutiremos los principales **objetivos y criterios** a tener en cuenta a la hora de decidirnos por una determinada **política de planificación**. A continuación realizaremos una clasificación de estos criterios agrupándolos en a**propiativos y no apropiativos**. Hablaremos del **reloj de interrupciones,** con la intención de aclarar cómo es posible la intervención del sistema operativo para evitar la monopolización de la CPU por parte de los usuarios. Dedicaremos especial atención al mecanismo de planificación basado en prioridades. Terminaremos haciendo un estudio y evaluación cualitativo de los algoritmos de planificación que se pueden emplear. Durante este repaso haremos una reflexión sobre las repercusiones en cuanto a eficiencia y tiempo de respuesta del parámetro tamaño de cuanto.

El **scheduling** (traducido como **planificación**) consiste en seleccionar los procesos al procesador o a los procesadores para que sean ejecutados a lo largo del tiempo, forma que se cumplan objetivos del sistemas tales como el tiempo de respuesta, la productividad y la eficiencia del procesador.

OBJETIVOS

- Distinguir entre los distintos tipos de planificación.
- Conocer los distintos algoritmos para la planificación de la CPU.
- Valorar el comportamiento de los distintos algoritmos de planificación aplicando distintos parámetros.

2 Planificación de Procesos

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Procesos: Planificación de Procesos e Hilos
La planificación de la CPU, en el sentido de conmutarla entre los distintos procesos, es una de las funciones del sistema operativo. Este despacho es llevado a cabo por un pequeño programa llamado **planificador a corto plazo** o **dispatcher** (depachador). La misión del ***dispatcher*** consiste en asignar la CPU a uno de los procesos preparados del sistema, para ello sigue un determinado algoritmo. Para que el **dispatcher** conmute el procesador entre dos procesos es necesario realizar un cambio de proceso.

Los acontecimientos que pueden provocar la llamada al *dispatcher* dependen del sistema (son un subconjunto de las interrupciones), pero son alguno de estos:

1. El proceso en ejecución acaba su ejecución o no puede seguir ejecutándose (por una E/S, operación WAIT sobre un semáforo a cero, etc).
2. Un elemento del sistema operativo ordena el bloqueo del proceso en ejecución.
3. El proceso en ejecución agota su cuanto de estancia en la CPU.
4. Un proceso pasa a estado preparado.

Hay que destacar el hecho de que cuanto menos se llame al *dispatcher* menos tiempo ocupa la CPU un programa del sistema operativo, y, por tanto, se dedica más tiempo a los procesos del usuario (un cambio de proceso lleva bastante tiempo).

Así, si sólo se activa el *dispatcher* como consecuencia de los acontecimientos 1 y 2 se estará haciendo un buen uso del procesador. Este criterio es acertado en sistemas por lotes, en los que los programas no son interactivos. Sin embargo, en un sistema de tiempo compartido no es adecuado, pues un proceso que se dedica a realizar cálculos, y no realiza E/S, monopoliza el uso de la CPU. En estos sistemas hay que tener en cuenta el conjunto de todos los procesos, activándose el *dispatcher* debido a la circunstancia tercera y, posiblemente, a la cuarta. Los sistemas operativos en que las circunstancias 3 y 4 no provocan la activación del *dispatcher* muestran preferencia por el proceso en ejecución, si no ocurre esto se tiene más en cuenta el conjunto de todos los procesos.

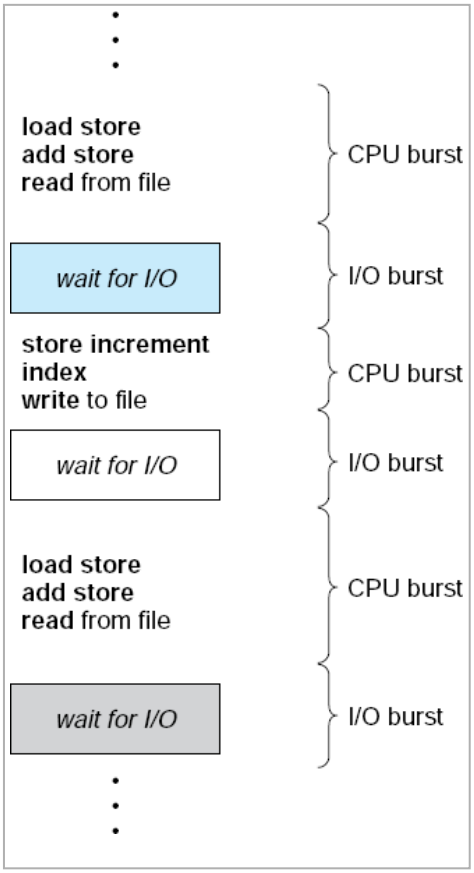


Figura 1. Secuencia de ráfagas de CPU y E/S.
Se puede definir el **scheduling** (traducido como **planificación**) como el conjunto de políticas y mecanismos construidos dentro del sistema operativo que gobiernan la forma de conseguir que los procesos a ejecutar lleguen a ejecutarse.
Por mecanismos se entiende los detalles de cómo se plasman las políticas. Por ejemplo, las estructuras de datos, el tipo de comunicación entre procesos, etc. Las políticas son las ideas que se pretenden realizar. Un ejemplo de política es que todos los procesos tengan igual tiempo de CPU.
El scheduling está asociado a las cuestiones de:

- Cuándo introducir un nuevo proceso en el sistema.
- Cuándo llevar parcial o completamente a memoria un proceso.
- Determinar el orden de ejecución de los procesos del sistema.
- Qué solicitud de E/S pendiente será tratada por un dispositivo de E/S disponible.

El scheduling relacionado con la gestión de procesos se divide en tres niveles:

- Planificador de la CPU, o a corto plazo.
- Planificador a medio plazo o intercambio.
- Planificador a largo plazo.

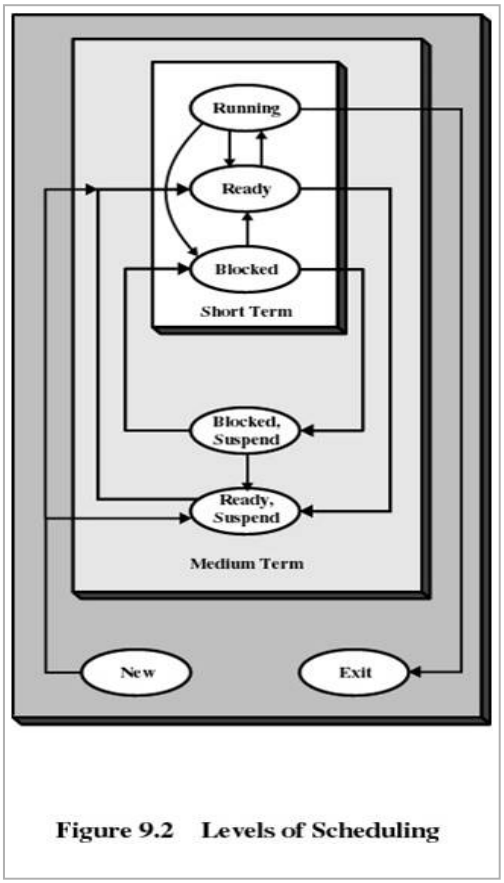


Figura 2. Niveles de Planificación.

2.1 Planificación a largo plazo

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Procesos: Planificación de Procesos e Hilos
Este planificador está presente en algunos sistemas que admiten además de procesos interactivos trabajos por lotes. Usualmente, se les asigna una prioridad baja a los trabajos por lotes, utilizándose estos para mantener ocupados a los recursos del sistema durante periodos de baja actividad de los procesos interactivos.

> Normalmente, los trabajos por lotes realizan tareas rutinarias como el cálculo de nóminas; en este tipo de tareas el programador puede estimar su gasto en recursos, indicándoselo al sistema. Esto facilita el funcionamiento del planificador a largo plazo.

El objetivo primordial del planificador a largo plazo es el de dar al planificador de la CPU una **mezcla equilibrada de trabajos**, tales como los limitados por la CPU (utilizan mucho la CPU) o la E/S. Así, por ejemplo, cuando la utilización de la CPU es baja, el planificador puede admitir más trabajos para aumentar el número de procesos preparados y, con ello, la probabilidad de tener algún trabajo útil en espera de que se le asigne la CPU. A la inversa, cuando la utilización de la CPU llega a ser alta, y el tiempo de respuesta (tiempo en que un programa interactivo tarda en responder a una petición de un usuario) comienza a reflejarlo, el planificador a largo plazo puede optar por reducir la frecuencia de admisión de trabajos.

Normalmente, **se invoca al planificador a largo plazo siempre que un proceso termina**. La frecuencia de invocación depende pues, de la **carga del sistema**, pero generalmente es mucho menor que la de los otros dos planificadores. Esta baja frecuencia de uso hace que este planificador pueda permitirse utilizar algoritmos complejos, basados en las estimaciones de los nuevos trabajos.

2.2 Planificación a medio plazo

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos

Bloque Procesos: Planificación de Procesos e Hilos
En los sistemas de multiprogramación y tiempo compartido **varios procesos residen en la memoria principal**. El tamaño limitado de ésta hace que el **número de procesos que residen en ella sea finito**. Puede ocurrir que todos los procesos en memoria estén bloqueados, desperdiciándose así la CPU. En algunos sistemas se intercambian procesos enteros entre memoria principal y memoria secundaria (normalmente discos), con esto se aumenta el número de procesos, y, por tanto, la probabilidad de una mayor utilización de la CPU.

El **planificador a medio plazo** es el ***encargado de regir las transiciones de procesos entre memoria principal y secundaria, actúa intentando maximizar la utilización de los recursos***. Por ejemplo, transfiriendo siempre a memoria secundaria procesos bloqueados, o transfiriendo a memoria principal procesos suspendidos-preparados en lugar de suspendidos-bloqueados.

3 Objetivos y Criterios de Planificación

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Procesos: Planificación de Procesos e Hilos
El principal objetivo de la **planificación a corto plazo** es repartir el tiempo del procesador de forma que se optimicen algunos puntos del comportamiento del sistema. Generalmente se fija un conjunto de criterios con los que evaluar las diversas estrategias de planificación. El criterio más empleado establece dos clasificaciones. En primer lugar, se puede hacer una distinción entre los **criterios orientados a los usuarios** y los **orientados al sistema**.

> Los **criterios orientados al usuario** se refieren al comportamiento del sistema tal y como lo perciben los usuarios o los procesos. Uno de los parámetros es el tiempo de respuesta. El **tiempo de respuesta** es el periodo de tiempo transcurrido desde que se emite una solicitud hasta que la respuesta aparece en la salida. Sería conveniente disponer de una política de planificación que ofrezca un buen servicio a diversos usuarios.
> Otros criterios **están orientados al sistema**, esto es, se centran en el uso efectivo y eficiente del procesador. Un ejemplo puede ser la **productividad**, es decir, el ritmo con que terminan los procesos. La productividad es una medida muy válida del rendimiento de un sistema, que sería deseable maximizar.

> Otra forma de clasificación es considerar los criterios relativos al **rendimiento del sistema** y los que no lo son. **Los criterios relativos al rendimiento son cuantitativos** y, en general, pueden evaluarse o ser analizados fácilmente. Algunos ejemplos son el **tiempo de respuesta** y la **productividad**. **Los criterios no relativos al rendimiento** son, en cambio, **cualitativos** y no pueden ser evaluados fácilmente. Un ejemplo de estos criterios es la **previsibilidad**. Sería conveniente que el servicio ofrecido a los usuarios tenga las mismas características en todo momento, independientemente de la existencia de otros trabajos ejecutados por el sistema.

En particular, una política de planificación debe:

- **Ser equitativa**: debe intentar hacer una planificación justa, esto es, se debe tratar a todos los procesos de la misma forma y no aplazar indefinidamente a ningún proceso. La mejor forma de evitarlo es emplear alguna técnica de envejecimiento; es decir, mientras un proceso espera un recurso, su prioridad debe crecer.
- **Ser eficiente**: debe maximizar el uso de los recursos tales como intentar que la ocupación de la CPU sea máxima. Al mismo tiempo se debe intentar reducir el gasto extra por considerar que es trabajo no productivo. Normalmente el idear algoritmos eficientes supone invertir recursos en gestión del propio sistema.
- **Lograr un tiempo de respuesta bueno**, es decir, que los usuarios interactivos reciban respuesta en tiempos aceptables.
- **Lograr un tiempo de proceso global predecible**. Esto quiere decir que un proceso debe ejecutarse aproximadamente en el mismo tiempo y casi al mismo costo con independencia de la carga del sistema.
- **Elevar al máximo la productividad** o el rendimiento, esto es, maximizar el número de trabajos procesados por unidad de tiempo.
    - Eso supone, por un lado, dar p**referencia a los procesos que ocupan recursos decisivos**. De este modo, se consigue liberar el recurso cuanto antes para que esté disponible para un proceso de mayor prioridad.
    - por otro, **favorecer a los procesos que muestran un comportamiento deseable**. Se escogen los procesos que no consumen muchos recursos dejándole al sistema mayor capacidad de actuación.

Estos criterios son dependientes entre sí y es imposible optimizar todos de forma simultánea.

- Obtener un **buen tiempo de respuesta** puede exigir un algoritmo de planificación que alterne entre los procesos con frecuencia, lo que **incrementa la sobrecarga del sistema y reduce la productividad**.

Por tanto, en el diseño de un política de planificación entran en juego compromisos entre requisitos opuestos; el peso relativo que reciben los distintos requisitos dependerá de la naturaleza y empleo del sistema.

4 Planificación apropiativa y no apropiativa

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Procesos: Planificación de Procesos e Hilos
Una disciplina de planificación es **no apropiativa** si una vez que la CPU ha sido asignada al proceso, ya no se le puede arrebatar. Y por el contrario, es **apropiativa**, si se le puede quitar la CPU.

La **planificación apropiativa** es útil en los sistemas en los que los procesos de alta prioridad requieren una atención rápida. En los de tiempo real, por ejemplo, las consecuencias de perder una interrupción pueden ser desastrosas. En los sistemas de tiempo compartido, la planificación apropiativa es importante para garantizar tiempos de respuesta aceptables.

La apropiación tiene un precio. El cambio de proceso implica gasto extra. Para que la técnica de apropiación sea efectiva deben mantenerse muchos procesos en memoria principal, de manera que el siguiente proceso se encuentre preparado cuando quede disponible la CPU. Conservar en memoria principal procesos que no están en ejecución implica gasto extra.

En **los sistemas no apropiativos**, los trabajos largos retrasan a los cortos, pero el tratamiento para todos los procesos es “más justo” (en el sentido de que una vez asignada la CPU a un proceso, no es desplazado por otro). Los tiempos de respuesta son más predecibles porque los trabajos nuevos de alta prioridad no pueden desplazar a los trabajos en espera.

Al diseñar mecanismos de planificación apropiativa no hay que perder de vista la arbitrariedad de casi todos los sistemas de prioridades. Se puede construir un mecanismo complejo para implantar fielmente un esquema de apropiación por prioridades sin que, de hecho, se hayan asignado prioridades de forma coherente.

5 El reloj de interrupciones

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Procesos: Planificación de Procesos e Hilos
Se dice que un proceso está en ejecución cuando tiene asignada la CPU. Si el proceso pertenece al sistema operativo, se dice que el sistema operativo está en ejecución y que puede tomar decisiones que afectan al sistema. Para **evitar que los usuarios monopolicen el sistema** (deliberada o accidentalmente), el sistema operativo tiene mecanismos para arrebatar la CPU al usuario.

El sistema operativo gestiona un **reloj de interrupciones** que genera una interrupción cada cierto tiempo. Un proceso mantiene el control de la CPU hasta que la libera voluntariamente (acaba su ejecución, o se bloquea), hasta que el reloj interrumpe o hasta que alguna otra interrupción desvía la atención de la CPU. **Si el proceso de usuario se encuentra en ejecución y el reloj interrumpe, el sistema operativo entra en ejecución** para comprobar, por ejemplo, si ha pasado el cuanto de tiempo del proceso que estaba en ejecución.

El **reloj de interrupciones** *asegura que ningún proceso acapare la utilización del procesador*. El sistema operativo, apoyándose en él, intenta distribuir el tiempo de CPU entre los distintos procesos ya sean de E/S o de cálculo. Por tanto, ayuda a garantizar tiempos de respuesta para los usuarios interactivos, evitando que el sistema quede bloqueado en un ciclo infinito de algún usuario, y permite que los procesos respondan a eventos dependientes del tiempo. Los procesos que deben ejecutarse periódicamente dependen del reloj de interrupciones.

No se debe confundir en ningún caso al reloj de interrupciones con el **reloj de la máquina o reloj hardware**. Veamos un pequeño ejemplo que aclara las diferencias entre estos dos relojes.
Como sabemos, todas las tareas de una computadora están sincronizadas por un *reloj hardware*. La velocidad de un procesador determina la rapidez con la que ejecuta un paso elemental o cambio en el sistema.

> Por ejemplo, si una máquina tiene un microprocesador que funciona a una frecuencia de 4000 MHz eso quiere decir que produce alrededor de cuatro mil millones de pasos elementales o cambios en el sistema en un segundo.
>
> Pero una instrucción consume algunos de estos pasos mínimos. Supongamos que en media una instrucción consume alrededor de 100 pasos elementales. No podemos interrumpir al procesador a la misma velocidad a la que opera porque entonces no se podría llegar nunca a ejecutar ninguna instrucción.
> Parece razonable que se elija una frecuencia menor para el reloj de interrupciones. Por ejemplo, se podría generar una interrupción cada 0'02 segundos (tener una frecuencia de 50 Hz) esto significa que se estaría interrumpiendo al procesador cada 80 millones de ciclos. En ese tiempo, bajo la suposición de que una instrucción consume 100 pasos, se habría ejecutado unas 800000 instrucciones. Esto sí es mucho más razonable.

En resumen, el **reloj de interrupciones tiene una frecuencia inferior al reloj hardware y superior al cuanto de tiempo o intervalo de tiempo en que se quiere controlar el sistema**.

6 Planificación por prioridades

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Procesos: Planificación de Procesos e Hilos
La mayoría de los algoritmos de planificación apropiativos emplean el uso de prioridades de acuerdo con algún criterio. Cada proceso tiene una prioridad asignada y el planificador seleccionará siempre un proceso de mayor prioridad antes que otro de menor prioridad.

Las prioridades pueden ser asignadas de forma automática por el sistema, o bien se pueden asignar externamente. Pueden ser estáticas o dinámicas. Pueden asignarse de forma racional, o de manera arbitraria en situaciones en las que un mecanismo del sistema necesita distinguir entre procesos, pero no le importa cuál de ellos es en verdad más importante.

Las **prioridades estáticas** no cambian. Los mecanismos de prioridad estática son fáciles de llevar a la práctica e implican un gasto extra relativamente bajo. Sin embargo, no responden a cambios en el entorno, que podrían hacer necesario un ajuste de prioridades.
Las **prioridades dinámicas** responden a los cambios. La prioridad inicial asignada a un proceso tiene una corta duración, después se ajusta a un valor más apropiado, a veces deducido de su comportamiento. Los esquemas de prioridad dinámica son más complejos e implican un mayor gasto extra que puede quedar justificado por el aumento en la sensibilidad del sistema.

7 Algoritmos de Planificación

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Procesos: Planificación de Procesos e Hilos
Se pueden utilizar distintos algoritmos para planificar la CPU. La elección de uno (o de una mezcla de varios) depende de decisiones de diseño. Antes de exponer los algoritmos hay que explicar ciertas medidas que se utilizan para evaluarlos.

1. Porcentaje de **utilización de la CPU** por procesos de usuario. La CPU es un recurso caro que necesita ser explotado, los valores reales suelen estar entre un 40% y un 90%.
2. **Rendimiento o productividad** (*throughput*) es el nº de ráfagas por unidad de tiempo. Se define una ráfaga como el periodo de tiempo en que un proceso necesita la CPU; un proceso, durante su vida, alterna ráfagas con bloqueos. Por extensión, también se define como el **nº de procesos por unidad de tiempo**.
3. **Tiempo de espera** (E) es el tiempo que una ráfaga ha permanecido en estado preparado, por extensión, es la **suma de los períodos invertidos en esperar en la cola de procesos preparados**.

4. **Tiempo de finalización** o **de retorno** (F) es el tiempo transcurrido desde que una ráfaga comienza a existir hasta que finaliza. F = E + t (t = tiempo de CPU de la ráfaga). Por extensión, el **tiempo que tarda en ejecutarse un proceso** que es la suma de los períodos que el proceso invierte en esperar a cargarse en memoria, esperar en la cola de procesos preparados, ejecutarse en la CPU y realizar las operaciones de E/S. Es todo el tiempo que pasa el proceso en el sistema, se puede calcular como la diferencia entre el tiempo de salida menos el tiempo de llegada.
5. **Penalización** (P) = (E + t) / t = F / t, es una medida adimensional que se puede aplicar homogéneamente a las ráfagas independientemente de su longitud.

En general, hay que

- maximizar los dos primeros parámetros (uso de la CPU y productividad) y
- minimizar los tres últimos (tiempo de espera, tiempo de finalización, penalización).

Sin embargo, estos **objetivos** son **contradictorios**, el dedicar más tiempo de CPU a los procesos de usuario se hace a costa de llamar menos al algoritmo de planificación (menos cambios de proceso), y de simplificarlo. Esto provoca que la CPU se reparta menos equitativamente entre los procesos, en detrimento de los últimos tres parámetros.

Así pues, dependiendo de los objetivos se elegirá cierto algoritmo. En los sistemas por lotes suele primar el rendimiento del sistema, mientras que en los sistemas interactivos es preferible minimizar, por ejemplo, el tiempo de espera.

7.1 Planificación Primero en Entrar-Primero en Salir (FIFO, First In First Out)

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Procesos: Planificación de Procesos e Hilos
Cuando se tiene que elegir a qué proceso asignar la CPU **se escoge al que llevara más tiempo preparado.** El proceso se mantiene en la CPU hasta que se bloquea voluntariamente. La implementación de la política FIFO se gestiona con una cola. Cuando un proceso entra en la cola de preparados, su BCP se coloca al final de la cola. Cuando la CPU queda libre, se asigna al proceso que esté al principio de la cola.

La ventaja de este algoritmo es su fácil implementación, sin embargo, no es válido para entornos interactivos ya que un proceso de mucho cálculo de CPU hace aumentar el tiempo de espera de los demás procesos.

Para implementar el algoritmo (ver Figura 3) sólo se necesita mantener una lista con los procesos preparados ordenada por tiempo de llegada. Cuando un proceso pasa de **bloqueado a preparado** se sitúa el último de la cola.

- En a) el proceso P7 ocupa la CPU, los procesos P2, P4 y P8 se mantienen en la lista de preparados.
- En b) P7 se bloquea (ya sea al realizar una E/S, una operación WAIT sobre un semáforo a cero u otra causa) y P2 pasa a ocupar la CPU.
- En c) ocurre un evento (finalización de la operación de E/S, operación SIGNAL, ...) que desbloquea a P7, esto lo vuelve al estado preparado, pasando al final de la lista de procesos preparados.

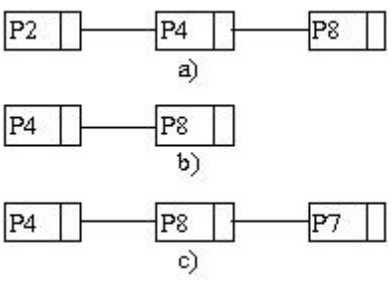


Figura 3. Lista de procesos listos o preparados en FIFO.

Algunas de las características de este algoritmo es que es **no apropiativo y justo en el sentido formal, aunque injusto** en el sentido de que: los **trabajos largos hacen esperar a los cortos** y los

trabajos sin importancia hacen esperar a los importantes. Por otro lado es **predecible**, pero **no garantiza buenos tiempos de respuesta**, por ello se emplea como esquema secundario.

7.2 Planificación por Turno Rotatorio (Round Robin)

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Procesos: Planificación de Procesos e Hilos
Este es uno de los algoritmos más antiguos, sencillos y equitativos en el reparto de la CPU entre los procesos, muy válido para **entornos de tiempo compartido**. Cada proceso tiene asignado un intervalo de tiempo de ejecución, llamado **cuantum o cuanto**. Si el proceso agota su cuanto de tiempo, se elige a otro proceso para ocupar la CPU. Si el proceso se bloquea o termina antes de agotar su cuanto también se alterna el uso de la CPU.

El round robin es muy fácil de implementar. Todo lo que necesita el planificador es mantener una lista de los procesos preparados, como se muestra en la Figura 4.

- En esta figura en a) el proceso P7 ocupa la CPU.
- En b) P7 se bloquea pasando P2 a ocupar la CPU.
- En c) P2 agota su cuanto con lo que pasa al final de la lista y P4 ocupa la CPU.

La Figura 5 representa un ejemplo más largo de la ocupación de la CPU utilizando el algoritmo round robin.

Este algoritmo presupone la existencia de un reloj en el sistema (el **reloj de interrupciones**). Un reloj es un dispositivo que **genera periódicamente interrupciones**. Esto es muy importante, pues garantiza que el sistema operativo (en concreto la rutina de servicio de interrupción del reloj) coge el mando de la CPU periódicamente. El cuanto de un proceso equivale a un número fijo de pulsos del reloj. **Al ocurrir una interrupción de reloj que coincide con la expiración del cuanto se llama al *dispatcher***.

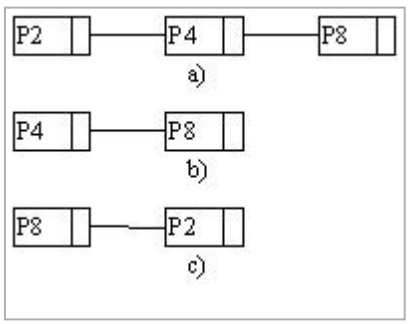


Figura 4. Lista de procesos preparados en Round-Robin.

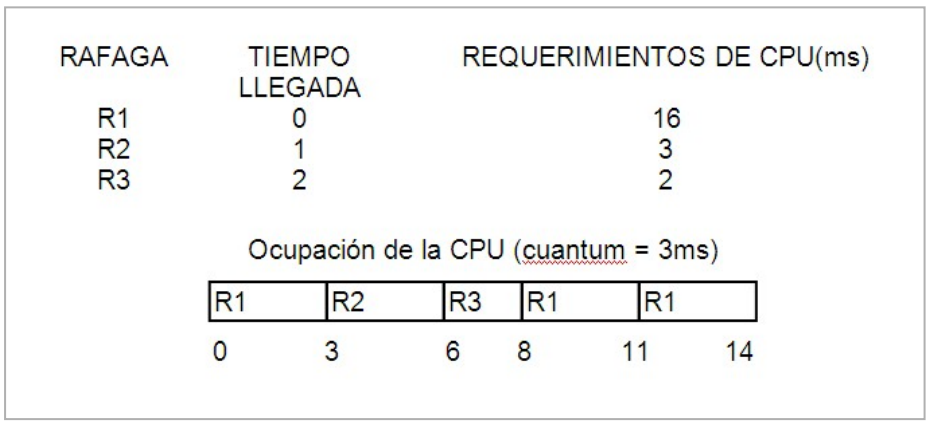


Figura 5. Ejemplo de Round-Robin.

7.3 Tamaño del Cuanto

2º Grado en Ingeniería en Informática

teoría de Sistemas Operativos
Bloque Procesos: Planificación de Procesos e Hilos
La determinación del tamaño del cuanto es vital para la operación efectiva de un sistema de cómputo. ¿Debe el cuanto ser pequeño o grande?, ¿fijo o variable?, ¿el mismo para todos los usuarios o debe determinarse por separado para cada uno?

Si el cuanto de tiempo es muy grande, cada proceso tendrá el tiempo necesario para terminar, de manera que el esquema de planificación por turno degenera en uno de primero-en-entrar-primero-en-salir.

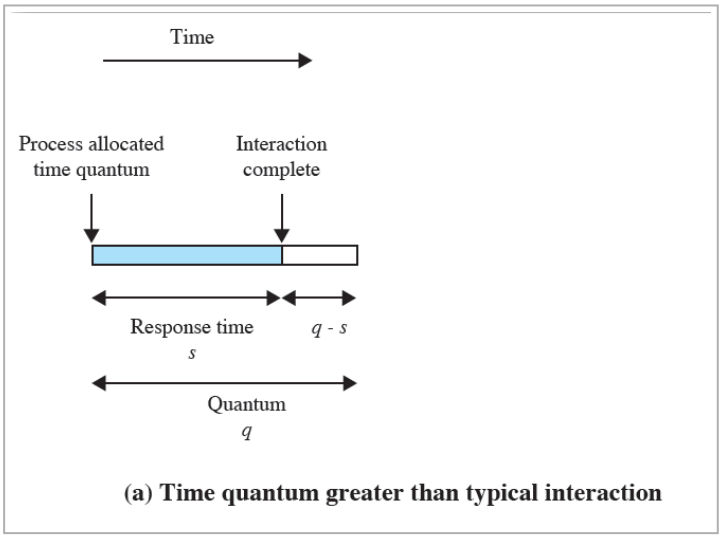
Si el cuanto es muy pequeño, el gasto extra por cambio de proceso se convierte en el factor dominante y el rendimiento del sistema se degradará hasta el punto en que la mayor parte del tiempo se invierte en la conmutación del procesador, con muy poco o ningún tiempo para ejecutar los programas de los usuarios.

¿Exactamente dónde, entre cero e infinito, debe **fijarse el tamaño del cuanto**? La respuesta es, **lo bastante grande como para que la mayoría de las peticiones interactivas requieran menos tiempo que la duración del cuanto**.

- Pongamos un ejemplo, supongamos que el cambio de proceso tarda 5 mseg., y la duración del cuanto es de 20 mseg.
- Con estos parámetros, se utiliza un mínimo del 20% del tiempo de la CPU en la ejecución del sistema operativo.
- Para incrementar la utilización de la CPU por parte de los procesos de usuario podríamos establecer un cuanto de 500 mseg., el tiempo desperdiciado con este parámetro sería del 1%.
- Pero consideremos lo que ocurriría si diez usuarios interactivos oprimieran la tecla `enter` casi al mismo tiempo. Diez procesos se colocarían en la lista de procesos preparados. Si la CPU está inactiva, el primero de los procesos comenzaría de inmediato, el segundo comenzaría medio segundo después, etc. Partiendo de la hipótesis de que todos los procesos agoten su cuanto, el último proceso deberá de esperar 4'5 seg. para poder ejecutarse. Esperar 4'5 seg. para la ejecución de una orden sencilla como pwd parece excesivo.

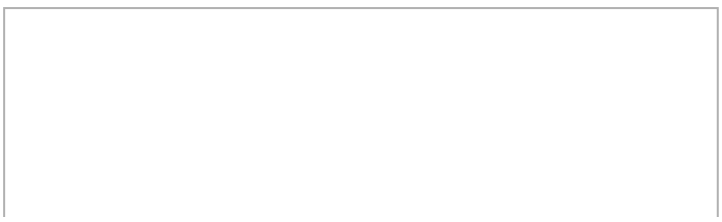
En conclusión, un cuanto corto disminuye el rendimiento de la CPU, mientras que un cuanto muy largo empobrece los tiempos de respuesta y degenera en el algoritmo FIFO. La solución es adoptar un término medio como 100 mseg.

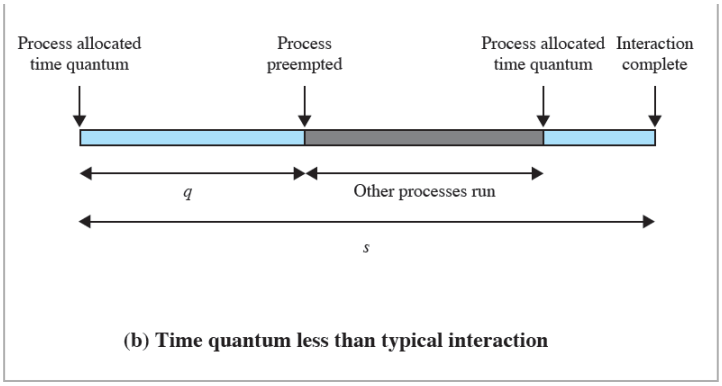
Tamaño Cuanto grande



(a) Time quantum greater than typical interaction

Tamaño Cuanto pequeño





(b) Time quantum less than typical interaction

7.4 Planificación por Prioridad al más Corto (SJF, Shortest Job First)

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Procesos: Planificación de Procesos e Hilos
Al igual que en el algoritmo FIFO las ráfagas se ejecutan sin interrupción, por tanto, sólo es útil para entornos *batch*. Su característica es que cuando se activa el planificador, éste elige la ráfaga de **menor duración**. Es decir, introduce una noción de prioridad entre ráfagas. Hay que recordar que en los entornos *batch* se pueden hacer estimaciones del tiempo de ejecución de los procesos.

La ventaja que presenta este algoritmo sobre el algoritmo FIFO es que **minimiza el tiempo de finalización promedio**, como puede verse en el siguiente ejemplo:

> Supongamos que en un momento dado existen tres ráfagas R1, R2 y R3, sus tiempos de ejecución respectivos son 24, 3 y 3 ms. El proceso al que pertenece la ráfaga R1 es el que lleva más tiempo listo, seguido del proceso al que pertenece R2 y del de R3. Veamos el tiempo medio de finalización (F) de las ráfagas aplicando FIFO y SJF:
>
> FIFO F = (24 + 27 + 30) / 3 = 27 ms.
> SJF F = (3 + 6 + 30) / 3 = 13 ms.

Se puede demostrar que este algoritmo es el óptimo. Para ello, consideremos el caso de cuatro ráfagas, con tiempos de ejecución de a, b, c y d. La primera ráfaga termina en el tiempo a, la segunda termina en el tiempo a+b, etc. El tiempo promedio de finalización es `(4a+3b+2c+d)/4`. Es evidente que a contribuye más al promedio que los demás tiempos, por lo que debe ser la ráfaga más corta, b la siguiente, y así sucesivamente. El mismo razonamiento se aplica a un número arbitrario de ráfagas.

No obstante, este algoritmo **sólo es óptimo cuando se tienen simultáneamente todas las ráfagas**.

> Como contraejemplo, considérense cinco ráfagas desde A hasta E, con tiempo se ejecución de 2, 4, 1, 1 y 1 respectivamente. Sus tiempos de llegada son 0, 0, 3, 3 y 3. Primero se dispone de A y B, puesto que las demás ráfagas no han llegado aún. Con el algoritmo SJF las ejecutaríamos en orden A, B, C, D, y E con un tiempo de finalización promedio de 4.6. Sin embargo, al ejecutarlas en orden B, C, D, E y A se tiene un promedio de finalización de 4.4.

7.5 Planificación por Prioridad al Tiempo Restante más Corto (SRT, Shortest Remaining Time)

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Procesos: Planificación de Procesos e Hilos
Es similar al SJF con la diferencia de que si un nuevo proceso pasa a preparado se activa el *dispatcher* para ver **si es más corto que lo que queda por ejecutar del proceso en ejecución**. Si es así, el proceso en ejecución pasa a listo y su tiempo de estimación se decrementa con el tiempo que ha estado ejecutándose.

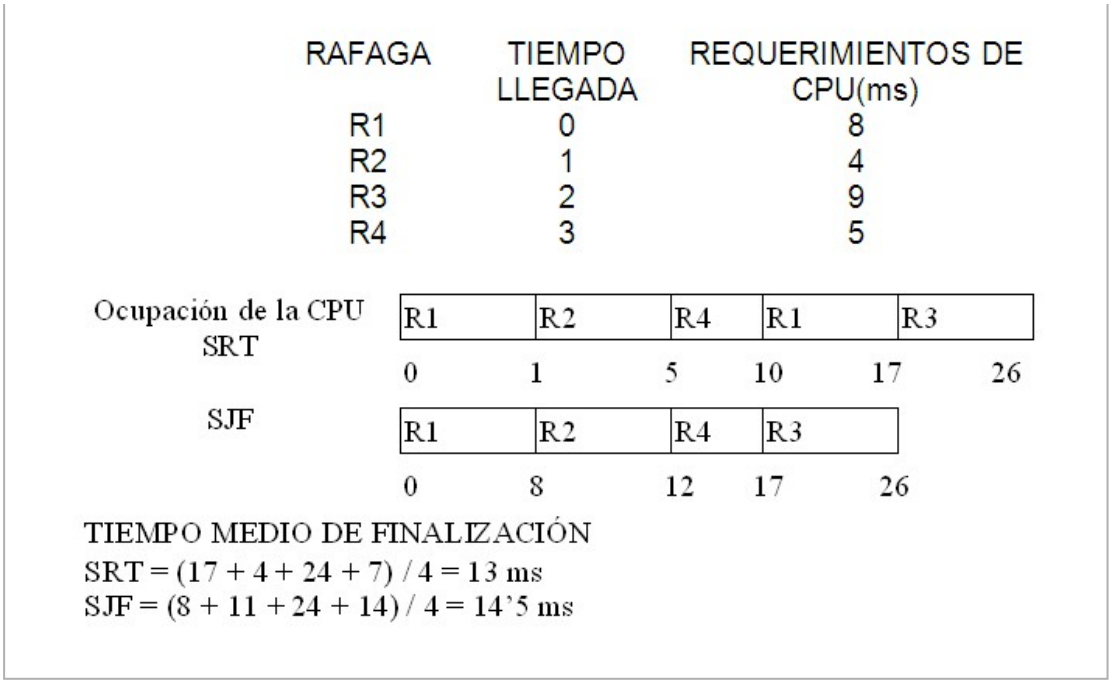


Figura 6. Ejemplo de SRT y SJF.

En la Figura 6 tenemos un ejemplo de funcionamiento del algoritmo en el que se observa cómo **se penaliza a las ráfagas largas** (como en SJF). Un punto débil de este algoritmo se evidencia cuando **una ráfaga muy corta suspende a otra un poco más larga**, siendo más larga la ejecución en este orden al ser preciso un cambio adicional de proceso y la ejecución del código del planificador.

7.6 Planificación a la Tasa de Respuesta más Alta

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Procesos: Planificación de Procesos e Hilos
Brinch Hansen desarrolló la estrategia de **prioridad a la tasa de respuesta más alta** (HRN, *Highest-Response-ratio-Next*) que corrige algunas deficiencias de SJF, particularmente, el retraso excesivo de trabajos largos, y el favoritismo excesivo por los trabajos cortos.

HRN es un disciplina de **planificación no apropiativa** en la cual la **prioridad** de cada proceso no sólo se calcula en **función del tiempo de servicio o de ejecución** (tiempo de CPU y bloqueos), sino también del **tiempo que ha esperado** para ser atendido. Cuando un trabajo obtiene el procesador, se ejecuta hasta terminar. Las prioridades dinámicas en HRN se calculan de acuerdo con la siguiente expresión:

```
Prioridad = (tiempo de espera + tiempo de servicio) / tiempo servicio
```

Como el **tiempo de CPU aparece en el denominador**, los **procesos cortos tendrán preferencia**. Pero como el **tiempo de espera aparece en el numerador**, los **procesos largos que han esperado también tendrán un trato favorable**. La suma tiempo de espera y tiempo de servicio es el tiempo de finalización para el proceso.

7.7 Colas Multinivel de Retroalimentación

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Procesos: Planificación de Procesos e Hilos
Cuando un proceso obtiene la CPU, sobre todo cuando todavía no ha tenido oportunidad de establecer un patrón de comportamiento, el **planificador no tiene idea de la cantidad de tiempo de CPU que necesitará el proceso**.

Los procesos limitados por la E/S normalmente usan la CPU sólo un momento antes de generar una solicitud de E/S; los procesos limitados por la CPU pueden usar el procesador durante horas si está disponible de forma no apropiativa.

Un mecanismo de planificación debe:

- **favorecer a los trabajos cortos**.
- **favorecer a los trabajos limitados por la E/S** para lograr un mejor aprovechamiento de los dispositivos de E/S (esto es, debe favorecer el paralelismo), y
- determinar la naturaleza de un trabajo lo más pronto posible, y **planificarlo de acuerdo con su naturaleza**.

- Las **colas multinivel de retroalimentación** (Figura 7) ofrecen una estructura que cumple con estos objetivos. Un proceso nuevo entra en la red de colas al final de la primera cola.
- Se desplaza en esa cola mediante Round Robin hasta que obtiene la CPU.
    - Si el trabajo termina o cede la CPU para esperar la terminación de una operación de E/S o de algún evento, el trabajo abandona la red de colas.
    - Si el cuanto expira antes de que el proceso ceda voluntariamente la CPU, el proceso se colocará al final de la cola del siguiente nivel. El proceso será atendido otra vez cuando llegue a la cabeza de esa cola si está vacía la primera.
- Mientras el proceso utilice todo el cuanto proporcionado en cada nivel, continuará desplazándose al final de la siguiente cola inferior.
- Por lo general, existe una cola en el nivel más bajo en la cual el proceso circula por turno rotatorio hasta que termina.

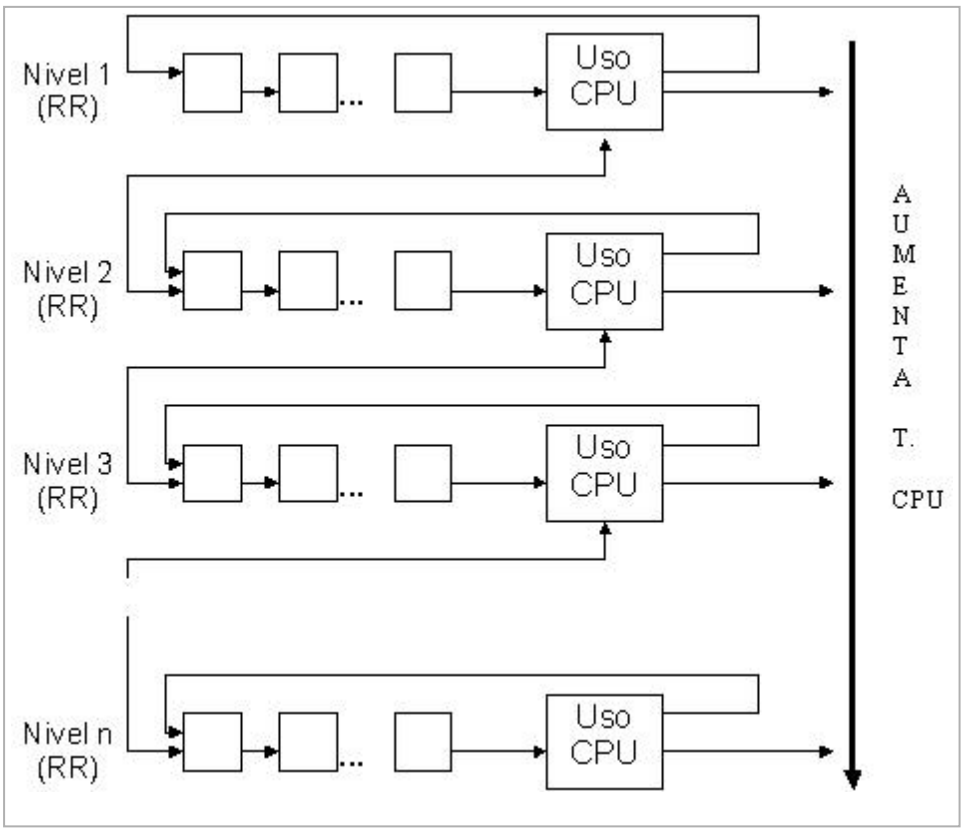


Figura 7. Colas de Retroalimentación de Niveles Múltiples.

En muchos esquemas de retroalimentación de niveles múltiples, **el cuanto asignado a un proceso cuando pasa a una cola de nivel inferior alcanza un valor mayor**.

- De esta forma, cuanto más tiempo se encuentre un proceso en la red de colas más grande será el cuanto asignado cada vez que obtenga la CPU, pero tal vez no obtenga la CPU muy a menudo, porque los procesos de las colas de nivel superior tienen mayor prioridad.
- **Un proceso situado en una cola no puede ejecutarse a menos que estén vacías las colas de nivel superior**.
- Un proceso en ejecución será desposeído por un proceso que llegue a una cola superior.

Considérese ahora cómo responde un mecanismo de este tipo a diferentes tipos de procesos.

El **mecanismo debe favorecer a los procesos limitados por la E/S** para lograr un buen aprovechamiento de los dispositivos y una respuesta buena para los usuarios interactivos; y de hecho lo hace, porque los procesos limitados por la E/S entrarán en la red con prioridad alta y se les asignará rápidamente la CPU.

El tamaño del cuanto de la primera cola se elegirá lo suficientemente grande para que la gran mayoría de los trabajos limitados por la E/S generen una petición de E/S antes

de que expire el primer cuanto. Cuando el proceso solicita E/S, abandona la red y ha obtenido un tratamiento favorable, tal como se deseaba.

Ahora considérese una **tarea limitada por la CPU** que necesita mucho tiempo de procesador.

1. Esa tarea entra en la cola más alta de la red con prioridad alta.
2. Recibe rápidamente su primera asignación de la CPU, pero su cuanto expira y el proceso se coloca en la cola del siguiente nivel inferior.
3. En ese momento, el proceso tiene una prioridad menor que la de los procesos que llegan al sistema, en particular los trabajos limitados por la E/S, que obtienen primero la CPU.
4. El proceso limitado por la CPU acaba recibiendo ésta, obtiene un cuanto mayor que en la cola más alta y vuelve a utilizar la totalidad de su cuanto.
5. Luego es situado al final de la siguiente cola inferior.
6. El proceso sigue desplazándose a colas inferiores, espera más entre divisiones de tiempo y utiliza todo su cuanto cada vez que obtiene la CPU (a menos que sea arrebatada por un proceso entrante).
7. En algún momento, el proceso limitado por la CPU llega a la cola de nivel inferior, en donde entrará en una planificación por turno hasta terminar.

- Las colas de retroalimentación de niveles múltiples son ideales para **separar procesos en categorías basadas en su necesidad de la CPU**.
- En un sistema de tiempo compartido, cada vez que un proceso abandona la red de colas puede "marcarse" con la identidad de la última cola en donde estuvo, y cuando el proceso entra de nuevo en la red de colas, puede enviarse directamente a la cola en la cual terminó su operación por última vez.
- En este caso, el planificador está usando un **razonamiento heurístico**, según el cual el comportamiento anterior del proceso es un buen indicador de su comportamiento en un futuro inmediato.
- De esta forma, un proceso limitado por la CPU que vuelve a la red de colas no se coloca en las colas de nivel alto donde interferiría con el servicio a los procesos cortos de prioridad alta o con los limitados por la E/S.

Si los procesos se colocan siempre dentro de la red en la cola que ocuparon la última vez, será imposible que el sistema responda a cambios de un proceso, por ejemplo, de estar limitado por la CPU, a estar limitado por la E/S.

El problema puede resolverse **marcando al proceso también con su duración dentro de la red** la última vez que estuvo en ella. Así, cuando el proceso entra de nuevo en la red puede colocarse en la cola correcta. Entonces, si el proceso entra en una fase nueva en la cual deja de estar limitado por la CPU y empieza a estar limitado por la E/S, el proceso experimentará en principio un tratamiento lento mientras el sistema determina que la naturaleza del proceso está cambiando. Pero el mecanismo de planificación responderá con rapidez a este cambio.

Otra forma de hacer que el sistema responda a los cambios de comportamiento de los procesos es **permitir que un proceso ascienda un nivel en la red de colas cada vez que abandona voluntariamente la CPU antes de que expire su cuanto**.

El mecanismo de colas de retroalimentación de niveles múltiples es un buen ejemplo de **mecanismo adaptativo**, que responde a los cambios en el comportamiento del sistema que controla. Los mecanismos adaptativos implican, en general, una carga extra mayor que los no adaptativos, pero **la sensibilidad ante los cambios en el sistema da como resultado una mejor capacidad de respuesta, y justifica el aumento en el gasto extra**.

Una variante común del mecanismo de colas de retroalimentación de niveles múltiples consiste en hacer que un proceso circule por turno varias veces en cada cola antes de pasar a la siguiente cola inferior. El número de ciclos en cada cola crece por lo regular cuando el proceso pasa a la siguiente cola inferior.

8 Comparación de las políticas de planificación

La figura 8 muestra las pautas de ejecución del ejemplo de la tabla 1 y la tabla 2 muestra algunos resultados.

Proceso	Tiempo de Llegada	Tiempo de CPU
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Tabla 1. Ejemplo de planificación de procesos

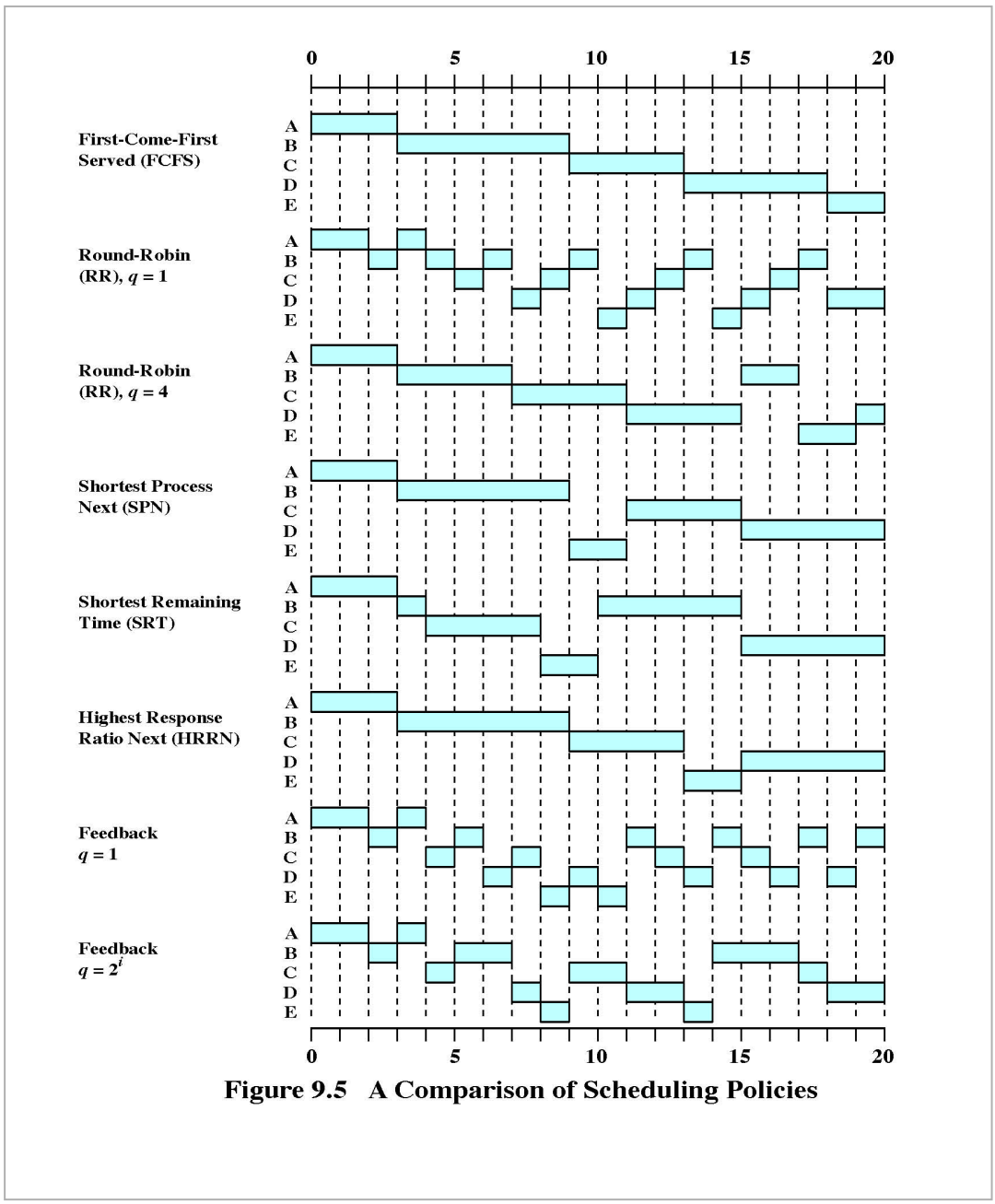


Figure 9.5 A Comparison of Scheduling Policies

Figura 8. Comparativa de políticas de planificación.

Process	A	B	C	D	E	
Arrival Time	0	2	4	6	8	
Service Time (T_S)	3	6	4	5	2	Mean
FCFS						
Finish Time	3	9	13	18	20	
Turnaround Time (T_r)	3	7	9	12	12	8.60
T_r/T_S	1.00	1.17	2.25	2.40	6.00	2.56
RR q = 1						
Finish Time	4	18	17	20	15	
Turnaround Time (T_r)	4	16	13	14	7	10.80
T_r/T_S	1.33	2.67	3.25	2.80	3.50	2.71
RR q = 4						
Finish Time	3	17	11	20	19	
Turnaround Time (T_r)	3	15	7	14	11	10.00
T_r/T_S	1.00	2.5	1.75	2.80	5.50	2.71
SPN						
Finish Time	3	9	15	20	11	
Turnaround Time (T_r)	3	7	11	14	3	7.60
T_r/T_S	1.00	1.17	2.75	2.80	1.50	1.84
SRT						
Finish Time	3	15	8	20	10	
Turnaround Time (T_r)	3	13	4	14	2	7.20
T_r/T_S	1.00	2.17	1.00	2.80	1.00	1.59
HRRN						
Finish Time	3	9	13	20	15	
Turnaround Time (T_r)	3	7	9	14	7	8.00
T_r/T_S	1.00	1.17	2.25	2.80	3.5	2.14
FB q = 1						
Finish Time	4	20	16	19	11	
Turnaround Time (T_r)	4	18	12	13	3	10.00
T_r/T_S	1.33	3.00	3.00	2.60	1.5	2.29
FB $q = 2^i$						
Finish Time	4	17	18	20	14	
Turnaround Time (T_r)	4	15	14	14	6	10.60
T_r/T_S	1.33	2.50	3.50	2.80	3.00	2.63

Process	A	B	C	D	E	
Arrival Time	0	2	4	6	8	
Service Time (T_s)	3	6	4	5	2	Mean
FCFS						
Finish Time	3	9	13	18	20	
Turnaround Time (T_r)	3	7	9	12	12	8.60
T_r/T_s	1.00	1.17	2.25	2.40	6.00	2.56
RR $q = 1$						
Finish Time	4	18	17	20	15	
Turnaround Time (T_r)	4	16	13	14	7	10.80
T_r/T_s	1.33	2.67	3.25	2.80	3.50	2.71
RR $q = 4$						
Finish Time	3	17	11	20	19	
Turnaround Time (T_r)	3	15	7	14	11	10.00
T_r/T_s	1.00	2.5	1.75	2.80	5.50	2.71
SPN						
Finish Time	3	9	15	20	11	
Turnaround Time (T_r)	3	7	11	14	3	7.60
T_r/T_s	1.00	1.17	2.75	2.80	1.50	1.84
SRT						
Finish Time	3	15	8	20	10	
Turnaround Time (T_r)	3	13	4	14	2	7.20
T_r/T_s	1.00	2.17	1.00	2.80	1.00	1.59
HRRN						
Finish Time	3	9	13	20	15	
Turnaround Time (T_r)	3	7	9	14	7	8.00
T_r/T_s	1.00	1.17	2.25	2.80	3.5	2.14
FB $q = 1$						
Finish Time	4	20	16	19	11	
Turnaround Time (T_r)	4	18	12	13	3	10.00
T_r/T_s	1.33	3.00	3.00	2.60	1.5	2.29
FB $q = 2^i$						
Finish Time	4	17	18	20	14	
Turnaround Time (T_r)	4	15	14	14	6	10.60
T_r/T_s	1.33	2.50	3.50	2.80	3.00	2.63

Tabla 2. Resultados comparativa de planificación de procesos.