Tema 4

Planificación

Un aspecto clave de los sistemas de multiprogramación es la planificación del procesador, ya que se ocupa de seleccionar entre todos los procesos que están a la espera a cuál de ellos se le va a asignar el procesador. En este capítulo se describirán los tres niveles de planificación existentes en los sistemas actuales, para centrarnos posteriormente en el funcionamiento y características de los principales algoritmos de planificación del procesador, así como los diferentes métodos para su evaluación.

4.1. Introducción

Un sistema de multiprogramación se caracteriza por la existencia de varios procesos residiendo simultáneamente en memoria principal. Dado que estamos considerando un sistema en el que sólo existe un procesador, los

procesos se tienen que alternar en el uso de éste. Por este motivo, el sistema operativo debe ejercer una labor de planificación, determinando en cada instante qué proceso tomará el control del procesador.

La planificación produce un efecto de intercalamiento en la ejecución de los procesos, como se muestra en la figura 4.1. En ella, mientras un proceso espera a que termine de realizarse una operación de E/S, el procesador ejecuta otro. Esto permite aumentar el número de procesos que se ejecutan por unidad de tiempo, es decir, el rendimiento del sistema (*throughput*).

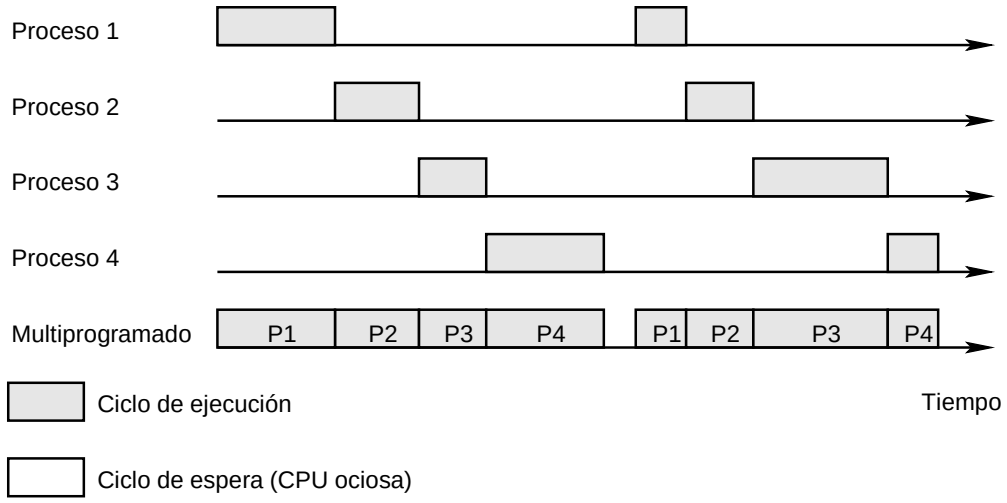

Figura 4.1: Intercalamiento en la ejecución de varios procesos

4.2. Niveles de planificación

La planificación afecta al rendimiento del sistema porque determina qué procesos esperarán y cuales progresarán. Fundamentalmente, la planificación es una cuestión de manejo de colas para minimizar la demora de los procesos en éstas y optimizar el rendimiento. En muchos sistemas, la actividad de planificación se divide en tres tipos: **planificación a largo**, **medio** y **corto plazo**, o también podemos encontrarnos con la denominación planificación de alto nivel, nivel intermedio y bajo nivel, respectivamente.

La figura 4.2 relaciona los distintos tipos de planificación con el diagrama de transiciones entre estados visto en el tema 3. La planificación a largo plazo se realiza con poca frecuencia, tomando la decisión de llevar o no un nuevo

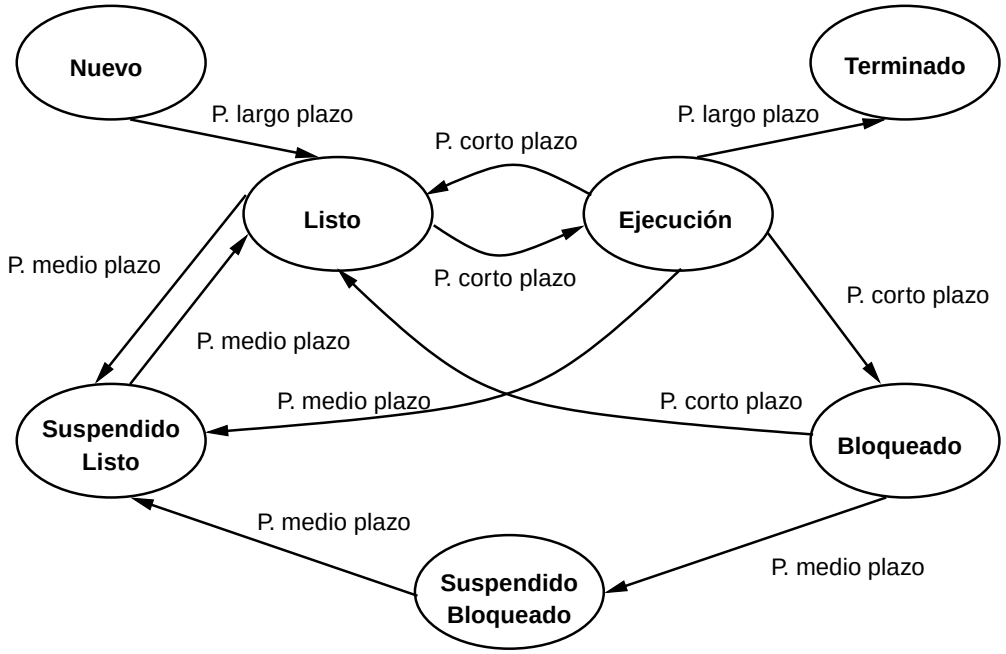

Figura 4.2: Planificación y transiciones de estado de los procesos

proceso al sistema y cuál. La planificación a medio plazo se ejecuta con más frecuencia que la anterior, siendo una parte de la función de intercambio. Se encarga de tomar la decisión de añadir o quitar un proceso a los que están disponibles para su ejecución. Finalmente, la planificación a corto plazo es la que se ejecuta con más frecuencia, y decide qué proceso, de todos los que se encuentran en estado listo, será el siguiente en ejecutarse.

4.2.1. Planificación a largo plazo

El planificador a largo plazo determina qué programas serán admitidos en el sistema para su ejecución, controlando el grado de multiprogramación. Cuando se admite un nuevo programa se convierte en proceso y se añade a la cola de listos, gestionada por el planificador a corto plazo.

En un sistema el planificador a largo plazo toma dos tipos de decisiones, si puede o no aceptar un nuevo proceso y, si esto es posible, cuál será.

Para tomar la decisión de crear o no un proceso nuevo se suele tener en cuenta el grado de multiprogramación, ya que cuantos más procesos haya en

el sistema menos tiempo que se le dedicará a cada uno. Por otro lado, si el grado de multiprogramación es muy bajo el procesador puede llegar a estar ocioso si están todos bloqueados a la espera de algún evento. El planificador a largo plazo va a controlar el grado de multiprogramación para ofrecer un buen servicio a los procesos activos y para obtener un buen rendimiento del sistema.

La decisión de qué trabajo admitir depende del tipo de sistema operativo. En un sistema por lotes, puede hacerse en el orden de llegada o usar otro criterio, tales como la prioridad, tiempo de ejecución estimado, etc. En un sistema de tiempo compartido, cuando un usuario intenta conectarse se genera una petición de creación de un proceso. Si el sistema no ha alcanzado el límite máximo de usuarios permitidos, se crea el proceso; en caso contrario, se niega la petición de conexión y se indica mediante un mensaje.

4.2.2. Planificación a medio plazo

La planificación a medio plazo es parte de la función de intercambio. Normalmente, la decisión de pasar un proceso de memoria principal a memoria secundaria está basada en la necesidad de modificar el grado de multiprogramación. En la asignatura de Sistemas Operativos II se verán los aspectos implicados en el intercambio.

4.2.3. Planificación a corto plazo

El objetivo principal de la planificación a corto plazo es asignar el procesador a los diferentes procesos que compiten por él, de forma que se optimicen uno o más aspectos del comportamiento del sistema. Para ello, el planificador a corto plazo evaluará en ciertos instantes una **función de selección** para todos los procesos listos. Ésta obtendrá un valor para cada proceso, de forma que al proceso que obtenga el valor más alto se le asignará la CPU. En caso de igualdad será necesaria una **regla de arbitraje** para resolver el conflicto. Si la probabilidad de que esto ocurra se considera baja, se podrá elegir el proceso al azar; en caso contrario, la regla de arbitraje deberá ser una nueva función de selección.

Los instantes en que se suele evaluar la función de selección, conocidos como **modo de decisión**, son los siguientes:

1. Cuando un proceso finaliza su ejecución.

2. Cuando un proceso cambia desde el estado de ejecución al estado bloqueado (por ejemplo, porque realiza una petición de E/S, o hace una llamada wait para esperar a que un proceso hijo termine).

3. Cuando un proceso cambia desde el estado de ejecución al estado de listo (por ejemplo, cuando ocurre una interrupción).

4. Cuando un proceso pasa desde el estado bloqueado, nuevo o suspendido-listo al estado listo (por ejemplo, se termina una operación de E/S).

Teniendo esto en cuenta, podemos distinguir dos tipos básicos de modos de decisión: **no apropiativo** y **apropiativo**.

Cuando la planificación tiene lugar sólo bajo las dos primeras circunstancias, tenemos una política de planificación no apropiativa, es decir, una vez que un proceso comienza a ejecutarse sólo se le puede retirar la CPU si se bloquea o termina.

En los demás casos se habla de planificación apropiativa, es decir, el proceso que se está ejecutando actualmente puede ser interrumpido y pasar a estado listo. Esta decisión la puede tomar el sistema operativo cuando llega un nuevo proceso, cuando un proceso bloqueado pasa a estado listo, cuando se activa un proceso suspendido, o periódicamente al producirse una interrupción del reloj.

Aunque las políticas apropiativas introducen mayor sobrecarga que las no apropiativas, suelen proporcionar un mejor servicio a los procesos. Esto es debido a que reparten mejor el tiempo del procesador entre los procesos, evitando que uno solo lo monopolice.

4.3. Algoritmos de planificación

A continuación describiremos distintas políticas de planificación de la CPU. Con objeto de compararlas, utilizaremos como ejemplo el conjunto de procesos que aparece en la tabla 4.1. Podemos considerar los procesos del ejemplo como trabajos por lotes, siendo el tiempo de servicio su tiempo total de ejecución. Otra posibilidad es considerarlos como procesos en curso que de forma alternativa usan el procesador y realizan operaciones de E/S; en este caso, el tiempo de servicio representa la duración de la siguiente ráfaga de CPU.

Proceso	P1	P2	P3	P4	P5
Hora de llegada	0	1	3	5	8
Tiempo de servicio	4	7	3	6	2
Prioridad	4	1	2	3	5

Cuadro 4.1: Conjunto de procesos

4.3.1. Primero en llegar, primero en ser servido

El algoritmo de planificación de la CPU más simple es primero en llegar, primero en ser servido[1]. Se trata de una política de planificación no apropiativa, donde los procesos simplemente se van ejecutando en orden de llegada. Cuando un nuevo proceso llega al sistema se incorpora a la cola de procesos en estado listo; cuando el proceso actual deja de ejecutarse (bien porque termina su ejecución o porque se bloquea) se elige al más antiguo de la cola. De forma matemática, la función de selección para un proceso P_i se puede expresar como $f(P_i) = t_e$, donde t_e es el tiempo que lleva el proceso en la cola de listos.

La figura 4.3 muestra el diagrama de Gantt para la ejecución de los procesos de nuestro ejemplo. A partir de éste podemos determinar el tiempo de finalización de cada proceso, así como calcular el tiempo de retorno, es decir, el tiempo total que el proceso pasa en el sistema (el tiempo de servicio más el tiempo de espera). Una magnitud más significativa es el **tiempo de retorno normalizado**, que es el cociente entre el tiempo de retorno de un proceso y su tiempo de servicio. Este valor indica la demora relativa que sufre un proceso. El valor mínimo para este cociente es 1; valores mayores indican un nivel decreciente de servicio. Otro valor significativo es el tiempo de respuesta, que se define como el intervalo de tiempo que transcurre desde que se envía una petición hasta que se empieza a recibir respuesta.

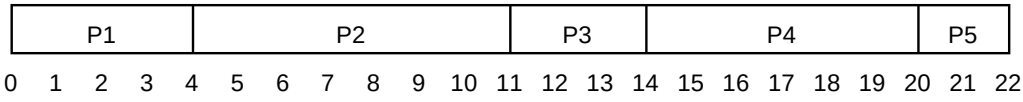

Figura 4.3: Algoritmo FIFO

La tabla 4.2 muestra los valores obtenidos para todos estos parámetros

[1]Este algoritmo puede aparecer en la bibliografía con las siglas FIFO (*First In First Out*) o bien FCFS (*First Come First Served*).

para cada proceso, así como los valores medios.

	P1	P2	P3	P4	P5	Media
T. de respuesta	0	3	8	9	12	6,4
T. de espera	0	3	8	9	12	6,4
T. de retorno	4	10	11	15	14	10,8
T. de retorno normalizado	1,00	1,43	3,67	2,50	7,00	3,12

Cuadro 4.2: Resultados para el algoritmo FIFO

Este algoritmo al ser no apropiativo perjudica a los procesos que necesitan ráfagas de CPU cortas (limitados por la E/S) frente a los que necesitan ráfagas largas (limitados por la CPU). Esto es debido a que estos últimos utilizan la CPU durante intervalos largos de tiempo, en los cuales todos los demás procesos deben esperar. Esto hace que los tiempos de retorno normalizado para los procesos cortos sean muy altos. Así, en el ejemplo anterior el proceso P5 con un tiempo de servicio de 2 unidades presenta un tiempo de retorno normalizado de 7.

Este inconveniente hace que FIFO por sí sola no sea una política de planificación atractiva, pero combinada con otras puede ser más interesante.

4.3.2. El proceso más corto primero

Para reducir la ventaja que se le da a los procesos largos en FIFO se puede utilizar la política del proceso más corto primero[2]. Se trata de una política no apropiativa en la que se selecciona para tomar el control de la CPU, el proceso con el tiempo de procesamiento esperado más corto. Matemáticamente, la función de selección para un proceso P_i se puede definir como $f(P_i) = 1/s$, siendo s el tiempo de servicio estimado para la siguiente ráfaga de CPU del proceso.

La figura 4.4 muestra el orden en que se ejecutarían los procesos de nuestro ejemplo y la tabla 4.3 los valores obtenidos para los tiempos.

Este algoritmo favorece a los procesos limitados por la E/S frente a los limitados por la CPU. Si comparamos los resultados obtenidos con los de FIFO, podemos ver que el tiempo de retorno se ha reducido significativamente para los procesos más cortos (P5 y P3), mientras que el proceso más largo

[2]Este algoritmo puede aparecer en la bibliografía con las siglas SPN (*Shortest Process Next*) o bien SJF (*Shortest Job First*).

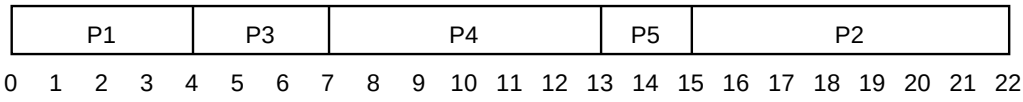

Figura 4.4: Algoritmo SPN

	P1	P2	P3	P4	P5	Media
T. de respuesta	0	14	1	2	5	4,4
T. de espera	0	14	1	2	5	4,4
T. de retorno	4	21	4	8	7	8,8
T. de retorno normalizado	1,00	3,00	1,33	1,33	3,50	2,03

Cuadro 4.3: Resultados para el algoritmo SPN

(P2) obtiene peor resultado que con el algoritmo FIFO. Sin embargo, el servicio global a los procesos mejora puesto que los valores medios para los tiempos de respuesta, de retorno y de retorno normalizado han disminuido.

Al favorecer a los procesos con ráfagas de CPU cortas, este algoritmo puede presentar el **bloqueo indefinido** o **muerte por inanición** de los procesos que necesitan ráfagas de CPU largas; es decir, un proceso de estas características se puede quedar esperando indefinidamente el uso de la CPU mientras estén llegando a la cola de listos procesos que necesitan ráfagas cortas.

SPN hace uso del tiempo de servicio estimado para la siguiente ráfaga de CPU. En los sistemas por lotes se puede pedir al programador que lo suministre al sistema. Para prevenir un uso incorrecto del sistema, si la estimación del programador está muy por debajo del valor real el sistema puede dar por finalizada la ejecución del proceso.

Sin embargo, en los sistemas interactivos este parámetro es difícil de conocer a priori, por lo que se hace necesario disponer de un método de estimación. El más simple se basa en la duración de las ráfagas anteriores:

$$S_{n+1} = \frac{1}{n} \sum_{i=1}^{n} T_i$$

donde:

T_i es la duración real de la i-ésima ráfaga de CPU.

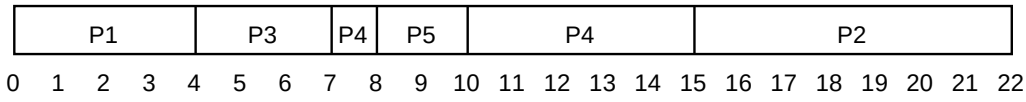

Figura 4.5: Algoritmo SRT

S_i es el valor estimado para la i-ésima ráfaga.

S_1 es el valor estimado para la primera, que no se puede calcular.

Para facilitar los cálculos, se puede escribir la ecuación anterior de la siguiente forma:

$$S_{n+1} = \frac{1}{n}T_n + \frac{n-1}{n}S_n$$

4.3.3. Tiempo restante más corto

El algoritmo del tiempo restante más corto[3] es la versión apropiativa del SPN. Por tanto, cada vez que un proceso llega a la cola de listos se evalúa la función de selección, que es tiempo de ejecución que le queda para completar la ráfaga actual. Así, si llega a la cola de listos un proceso con una ráfaga de CPU menor que el tiempo de servicio que le queda al que se está ejecutando, éste será apropiado por el nuevo proceso. Matemáticamente, la función de selección para un proceso P_i se puede definir como $f(P_i) = 1/(s-e)$, donde s es el tiempo de servicio total estimado para la siguiente ráfaga de CPU del proceso y e el tiempo de ejecución que ya ha dedicado a esa ráfaga. Al igual que en SPN, es necesario estimar los tiempos de servicio de cada ráfaga de CPU del proceso.

La figura 4.5 muestra el patrón de ejecución de los procesos del ejemplo y en la tabla 4.4 aparecen los resultados que se obtienen.

SRT proporciona un servicio inmediato a los procesos con ráfagas de CPU más cortas. En nuestro ejemplo, los dos procesos más cortos (P4 y P5) obtienen un tiempo de retorno normalizado de 1.

Al igual que el algoritmo SPN, puede presentar la inanición de los procesos con ráfagas de CPU largas, siendo más acusado aún que con SPN, debido a

[3]Se suele conocer con las siglas SRT (*Shortest Remaining Time*).

	P1	P2	P3	P4	P5	Media
T. de respuesta	0	14	1	2	0	3,4
T. de espera	0	14	1	4	0	3,8
T. de retorno	4	21	4	10	2	8,2
T. de retorno normalizado	1,00	3,00	1,33	1,67	1,00	1,60

Cuadro 4.4: Resultados para el algoritmo SRT

que incluso cuando el proceso largo tenga el control de la CPU, la llegada de un proceso corto provocará que se le quite.

Este algoritmo introduce una cierta sobrecarga en el sistema con respecto a los anteriores, ya que se deben registrar los tiempos de ejecución transcurridos.

4.3.4. Tasa de respuesta más alta

El algoritmo de tasa de respuesta más alta[4] intenta minimizar el tiempo de retorno normalizado de cada proceso. Para ello, define la **tasa de respuesta** como el cociente entre el tiempo de espera más el de servicio y el tiempo de servicio. La función de selección para un proceso P_i será $f(P_i) = (t_e + s)/s$, donde t_e es el tiempo que lleva el proceso esperando en la cola de listos, y s es el tiempo de servicio estimado para la siguiente ráfaga de CPU. Al igual que ocurría con SPN y SRT, es necesario conocer de antemano el tiempo de servicio de los procesos.

Se trata de un algoritmo no apropiativo, por lo que sólo cuando un proceso termina o se bloquea, el planificador escoge el de tasa de respuesta más alta.

Este método favorece a los procesos más cortos, pero al tener en cuenta el tiempo de espera se evita la inanición de los procesos largos.

En la figura 4.6 se muestra el patrón de ejecución de los procesos del ejemplo siguiendo este algoritmo, recogiéndose los valores de los tiempos obtenidos en la tabla 4.5.

Si comparamos los resultados obtenidos con los del algoritmo SPN, podemos observar que los procesos largos obtienen mejores tiempos de retorno normalizado (P2), mientras que los cortos obtienen tiempos ligeramente mayores (P5).

[4]Puede aparecer con las siglas HRRN (*Highest Response Ratio Next*).

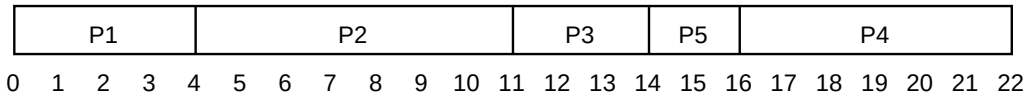

Figura 4.6: Algoritmo HRRN

	P1	P2	P3	P4	P5	Media
T. de respuesta	0	3	8	11	6	5,6
T. de espera	0	3	8	11	6	5,6
T. de retorno	4	10	11	17	8	10,0
T. de retorno normalizado	1,00	1,43	3,67	2,83	4,00	2,59

Cuadro 4.5: Resultados para el algoritmo HRRN

4.3.5. Asignación por turnos

El algoritmo de asignación por turnos[5] fue diseñado especialmente para los sistemas de tiempo compartido. Atiende a los procesos en el orden de llegada a la cola de listos, pero introduce la apropiación basada en las interrupciones producidas por un reloj. Cuando el planificador elige un proceso para su ejecución, inicializa el temporizador para producir una interrupción transcurrido cierto tiempo. Ese intervalo de tiempo se denomina **cuanto**. Si el proceso en ejecución agota el cuanto, se producirá la interrupción y el planificador lo situará al final de la cola de listos, eligiendo el primer proceso de ésta para su ejecución. Si antes de agotar el cuanto el proceso abandona la CPU (se bloquea o termina), el planificador elegirá un nuevo proceso para su ejecución.

El principal aspecto de diseño de este algoritmo es el tamaño del cuanto, es decir, el intervalo de tiempo de uso de la CPU que se le da a cada proceso. No existe una regla fija que determine cuál es su tamaño óptimo. Si elegimos un cuanto muy pequeño, los procesos cortos tendrán un tiempo de retorno relativamente pequeño, pero se producirán muchas interrupciones y cambios de proceso con la consiguiente sobrecarga que esto conlleva. Si se elige un cuanto muy grande, este algoritmo se comportaría como un FIFO.

Este algoritmo es especialmente idóneo para sistemas interactivos donde se requiere que los tiempos de respuesta sean cortos para todos los procesos.

Las figuras 4.7 y 4.8 muestra cómo se ejecutarían los procesos del ejemplo

[5]En la bibliografía en lengua inglesa se le denomina *Round robin*.

siguiendo el algoritmo de asignación por turnos con un cuanto de 2 y 4, respectivamente. En las tablas 4.6 y 4.7 aparecen los valores de los tiempos. Se puede observar como los procesos más cortos, como el P5, mejoran de forma significativa con este algoritmo.

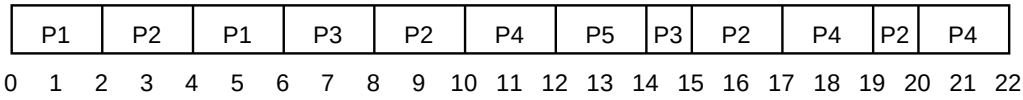

Figura 4.7: Algoritmo RR con cuanto de 2 unidades

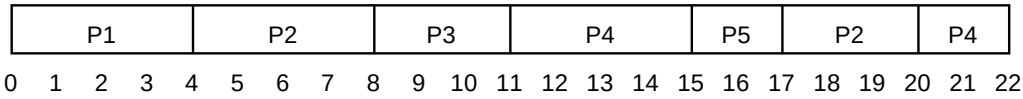

Figura 4.8: Algoritmo RR con cuanto de 4 unidades

	P1	P2	P3	P4	P5	Media
T. de respuesta	0	1	3	5	4	2,6
T. de espera	2	12	9	11	4	7,6
T. de retorno	6	19	12	17	6	12,0
T. de retorno normalizado	1,50	2,71	4,00	2,83	3,00	2,81

Cuadro 4.6: Resultados para el algoritmo RR con un cuanto de 2 unidades

Si comparamos los resultados obtenidos en los ejemplos anteriores (tablas 4.6 y 4.7), podemos observar que al aumentar el cuanto el tiempo medio de respuesta aumenta. Por otro lado, al aumentar el cuanto, se verán favorecidos aquellos procesos cuya ráfaga de CPU sea menor o igual que el nuevo cuanto (P3), disminuyendo su tiempo de retorno normalizado. Sin embargo, los procesos cuya ráfaga de CPU era menor o igual que el cuanto pequeño (P5) ahora tendrán que esperar más, aumentando su tiempo de retorno normalizado.

4.3.6. Prioridades

Un aspecto importante de la planificación es el uso de prioridades. En muchos sistemas, a cada proceso se le asigna una prioridad, y el planificador elegirá siempre un proceso de mayor prioridad.

	P1	P2	P3	P4	P5	Media
T. de respuesta	0	3	5	6	7	4,2
T. de espera	0	12	5	11	7	7,0
T. de retorno	4	19	8	17	9	11,4
T. de retorno normalizado	1,00	2,71	2,67	2,83	4,50	2,74

Cuadro 4.7: Resultados para el algoritmo RR un cuanto de 4 unidades

La planificación por prioridad puede ser apropiativa o no apropiativa. Si se está utilizando un algoritmo apropiativo, cuando llega un proceso a la cola de listos, se compara su prioridad con la del que se está ejecutando actualmente; si la prioridad del nuevo proceso es mayor, se le retirará la CPU al proceso actual para cedérsela al nuevo. Un algoritmo no apropiativo pondrá al nuevo proceso en la cola de listos.

En el ejemplo que estamos utilizando se ha dado una prioridad a cada proceso, donde el número mayor indica una prioridad superior. Las figuras 4.9 y 4.10 muestran cómo se ejecutarían los procesos del ejemplo siguiendo un algoritmo por prioridades no apropiativo y apropiativo, respectivamente. Las tablas 4.8 y 4.9 muestran los resultados obtenidos para estos algoritmos.

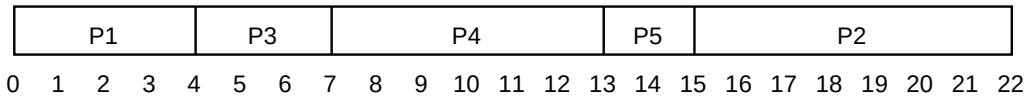

Figura 4.9: Algoritmo por prioridades no apropiativo

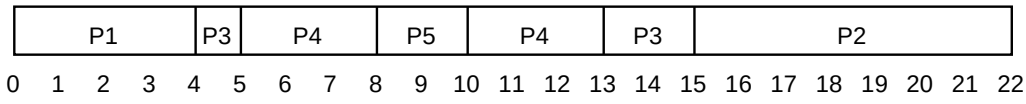

Figura 4.10: Algoritmo por prioridades apropiativo

Comparando los resultados para las dos versiones del algoritmo podemos ver cómo el proceso de mayor prioridad (P5) se ve favorecido en la versión apropiativa, ya que disminuye su tiempo de retorno normalizado. Por contra, un proceso de baja prioridad (P3) incrementa su tiempo de retorno normalizado en la versión apropiativa. Además, el proceso de menor prioridad (P2) se ve discriminado en las dos versiones.

	P1	P2	P3	P4	P5	Media
T. de respuesta	0	14	1	2	5	4,4
T. de espera	0	14	1	2	5	4,4
T. de retorno	4	21	4	8	7	8,8
T. de retorno normalizado	1,00	3,00	1,33	1,33	3,50	2,03

Cuadro 4.8: Resultados para el algoritmo de prioridades no apropiativo

	P1	P2	P3	P4	P5	Media
T. de respuesta	0	14	1	0	0	3,0
T. de espera	0	14	9	2	0	5,0
T. de retorno	4	21	12	8	2	9,4
T. de retorno normalizado	1,00	3,00	4,00	1,33	1,00	2,07

Cuadro 4.9: Resultados para el algoritmo de prioridades apropiativo

Al igual que los algoritmos anteriores también se puede dar la inanición de los procesos con una baja prioridad, siendo este problema más grave en la versión apropiativa. Una posible solución a este problema es considerar el **envejecimiento** de los procesos. Esta técnica incrementa gradualmente la prioridad de los procesos en espera, de forma, que pasado cierto tiempo su prioridad llega a ser mayor que la de los demás, permitiéndole recuperar el control de la CPU. La función de selección que representaría esta técnica podría ser $f(P_i) = p_i + \alpha \times t_e$, donde p_i es la prioridad base del proceso, t_e es el tiempo que lleva esperando y α es un coeficiente que regula el envejecimiento ($\alpha \geq 1$).

4.3.7. Planificación en varios niveles

Todas las estrategias estudiadas anteriormente siguen un único criterio de planificación. Sin embargo es posible combinar varios criterios en un algoritmo. Un ejemplo de esto es la planificación en varios niveles.

En un sistema es posible encontrar procesos con diferentes necesidades de planificación, por ejemplo, las necesidades en cuanto al tiempo de respuesta no son las mismas para un proceso por lotes y para uno interactivo. Teniendo esto en cuenta se podrían clasificar los procesos según sus necesidades. Esto podría emplearse para dividir la cola de procesos listos en varias, cada una con una prioridad diferente. A su vez cada cola podría tener su propio algoritmo de planificación, que respondería a las necesidades de los procesos que van a

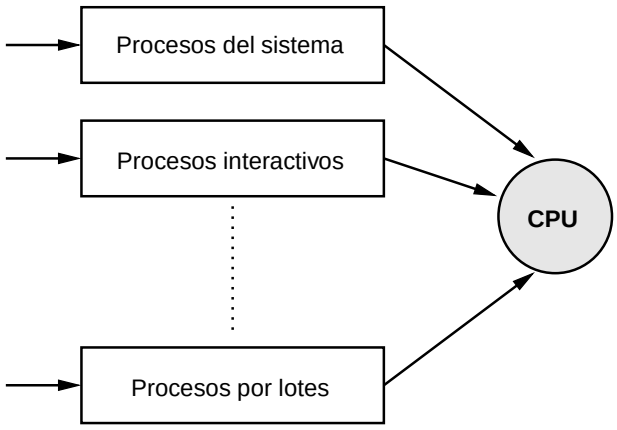

Figura 4.11: Planificación en varios niveles

ella.

Cuando un proceso entra al sistema se asigna a una cola, que será la empleada cada vez que se encuentre en estado listo. Cuando hay que elegir un proceso, se tomará uno de la cola de mayor prioridad; si durante su ejecución llegase un proceso a una cola con prioridad superior, tomaría el control de la CPU. Un posible implementación puede verse en la figura 4.11.

4.3.8. Planificación en varios niveles con realimentación

Otra posible combinación de estrategias de planificación es el algoritmo de varios niveles con realimentación. Una posible implementación de éste se puede ver en la figura 4.12.

Cuando un proceso entra por primera vez en el sistema, se coloca en la cola de mayor prioridad. Cuando vuelve al estado listo después de haberse ejecutado por primera vez pasa a la cola de nivel inmediatamente inferior, y así sucesivamente va bajando a colas de prioridad inferior. Dentro de cada cola, excepto en la de más baja prioridad, se utiliza un algoritmo FIFO. Una vez que un proceso está en la cola de menor prioridad no puede seguir bajando, por lo que siempre vuelve a ésta, es decir, se utiliza la asignación por turnos.

Hay múltiples variaciones de este esquema. Una posibilidad es establecer un cuanto para cada cola, de forma que si el proceso lo agota pasa a la

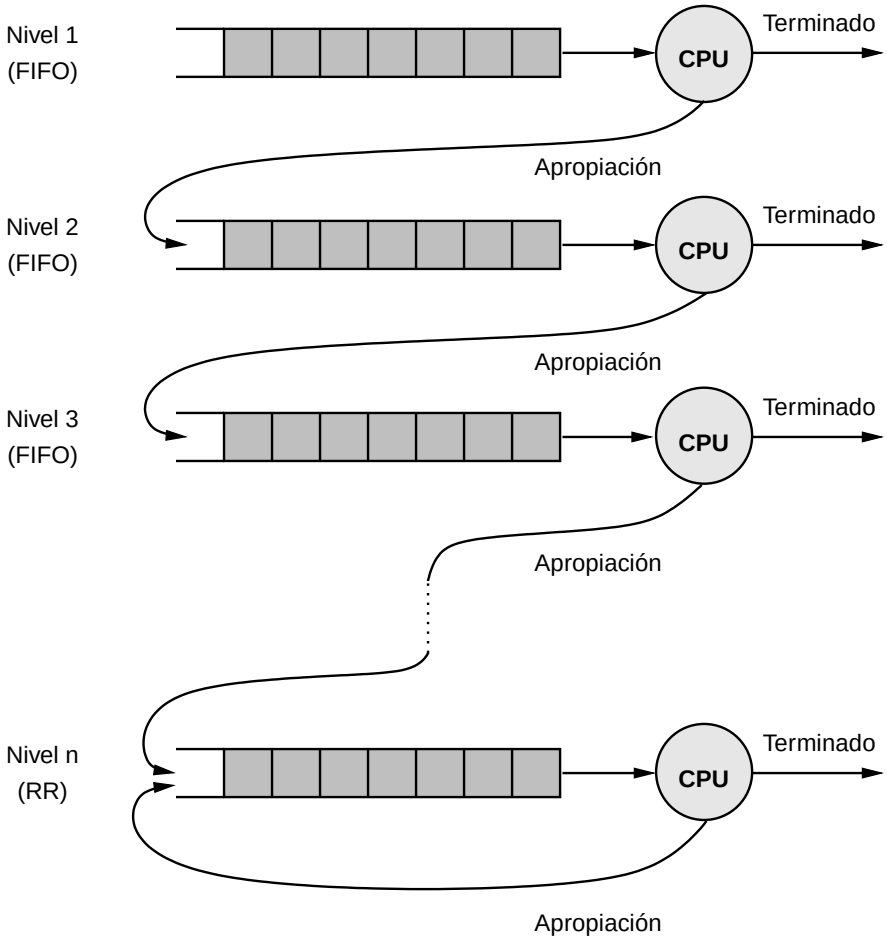

Figura 4.12: Planificación en varios niveles con realimentación

cola de nivel inferior. Los cuantos pueden ser iguales para todas las colas o diferentes. La figura 4.13 muestra el orden de ejecución de los procesos de nuestro ejemplo utilizando un cuanto de una unidad en las 3 colas disponibles. La tabla 4.10 muestra los valores que se obtienen para los tiempos.

Como podemos ver, un proceso corto se completará de forma rápida sin bajar demasiado en la jerarquía de colas. Sin embargo, un proceso largo irá bajando más, pudiendo llegar a la última cola. Este algoritmo, por tanto, favorece a los procesos cortos frente a los largos, sin necesidad de conocer a priori su tiempo de servicio. El problema que aparece es el aumento que experimenta el tiempo de retorno para los procesos largos. Incluso es posible que éstos sufran inanición si regularmente van entrando nuevos procesos

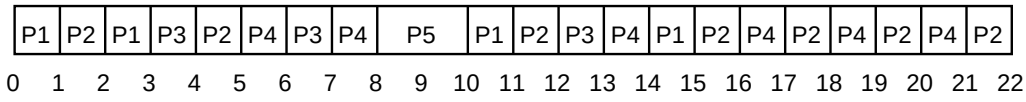

Figura 4.13: Planificación en varios niveles con realimentación para q=1

	P1	P2	P3	P4	P5	Media
T. de respuesta	0	0	0	0	0	0
T. de espera	11	14	7	10	0	8,4
T. de retorno	15	21	10	16	2	12,8
T. de retorno normalizado	3,75	3,00	3,33	2,67	1,00	2,75

Cuadro 4.10: Resultados para varios niveles con realimentación (q=1)

al sistema. Para compensar esto, podemos variar el valor del cuanto según la cola, es decir, a medida que tenemos una cola de menor prioridad se le asignará un cuanto mayor.

La figura 4.14 muestra el orden de ejecución de los procesos de nuestro ejemplo en un sistema con 3 colas y un cuanto variable igual a 2^n donde n es el número de la cola, siendo el valor inicial $n = 0$. La tabla 4.11 indica los resultados de los tiempos obtenidos. En la figura 4.15 podemos ver el estado de las distintas colas en el instante $t = 8$, cuando acaba de llegar el proceso P5.

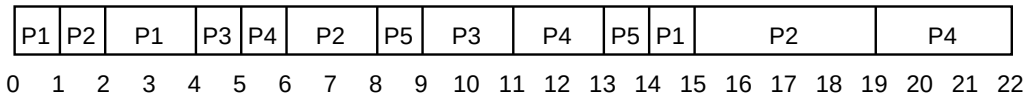

Figura 4.14: Planificación en varios niveles con realimentación para q=2^n

	P1	P2	P3	P4	P5	Media
T. de respuesta	0	0	1	0	0	0,2
T. de espera	11	11	5	11	4	8,4
T. de retorno	15	18	8	17	6	12,8
T. de retorno normalizado	3,75	2,57	2,67	2,83	3,00	2,96

Cuadro 4.11: Resultados para varios niveles con realimentación (q=2^n)

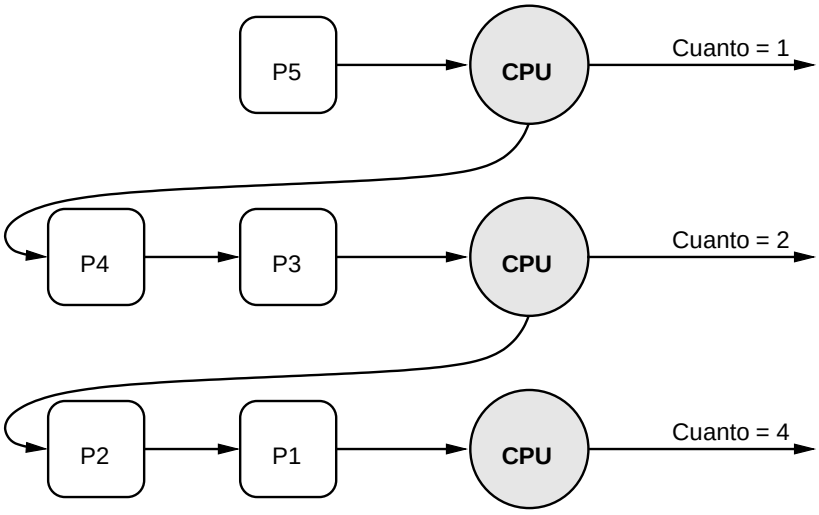

Figura 4.15: Instante $t = 8$ para el ejemplo de planificación en varios niveles con realimentación para q=2^n

El algoritmo de planificación de varios niveles con realimentación es el más general que se puede emplear, permitiendo adaptarlo a las necesidades de nuestro sistema mediante la configuración de diversos parámetros. Entre éstos se encuentran:

- El número de colas.
- El algoritmo de planificación de cada cola.
- El método utilizado para determinar cuando se pasa un proceso a una cola de menor prioridad, por ejemplo, los procesos podrían dar más de una vuelta en la misma cola.
- El método utilizado para determinar cuando se pasa un proceso a una cola de mayor prioridad, para esto se podría fijar un tiempo máximo de espera de los procesos en cada cola.
- El método utilizado para determinar en qué cola entrará un proceso cuando se desbloquea.

4.4. Evaluación de algoritmos de planificación

En el apartado 4.3 hemos visto muchos algoritmos de planificación cada uno con sus propias características, la cuestión es ¿qué algoritmo de planificación elegiremos para un sistema particular? La selección de uno puede ser una tarea difícil.

El primer problema que nos encontramos es definir el criterio que vamos a usar para seleccionar el algoritmo. A continuación veremos algunos de los que podemos emplear para el planificador a corto plazo.

4.4.1. Criterios del planificador a corto plazo

Se puede establecer un conjunto de criterios para la elección del algoritmo de planificación. Éstos se pueden clasificar en dos grupos: los orientados al usuario y los orientados al sistema. Los primeros están relacionados con el comportamiento de un sistema tal como lo percibe el propietario de un proceso individual. Los segundos tienen como objetivo principal es el uso eficiente del procesador.

También se pueden clasificar los criterios de otra forma: aquellos que están relacionados con el rendimiento del sistema y los que no lo están. Normalmente los primeros son cuantitativos, mientras que los segundos son cualitativos.

En las tablas 4.12 y 4.13 se hace un resumen de todos estos criterios de planificación. Los criterios son interdependientes, y es imposible optimizarlos todos simultáneamente, de forma que el diseño de una política de planificación implica un compromiso entre requisitos opuestos. Por ejemplo, para proporcionar un buen tiempo de respuesta se requerirá un algoritmo de planificación que cambie los procesos frecuentemente, incrementando así la carga del sistema y reduciendo el rendimiento.

La elección del criterio dependerá de la naturaleza y uso del sistema. Así, en la mayor parte de los sistemas interactivos se emplea el tiempo de respuesta. Por contra, en sistemas por lotes se busca mejorar el tiempo de retorno.

Una vez decidido el criterio que se va a utilizar habrá que evaluar diferentes algoritmos frente a él. Existen diversos métodos de evaluación, tales como la analítica, la simulación y la implementación.

	Criterios relacionados con el rendimiento, orientados al usuario
Tiempo de respuesta	Es el intervalo de tiempo desde que se envía una petición hasta que se empieza a atender.
Tiempo de retorno	Es el intervalo de tiempo entre el envío de un proceso y su terminación. Incluye el tiempo de ejecución más el tiempo de espera de los recursos, incluido el procesador.
Plazos	Para los sistemas de tiempo real su objetivo es terminar los procesos en un cierto plazo.
	Otros criterios orientados al usuario
Previsibilidad	Un proceso debería ejecutarse en la misma cantidad de tiempo, aproximadamente, independientemente de la carga del sistema.

Cuadro 4.12: Criterios de planificación orientados al usuario

	Criterios relacionados con el rendimiento orientados al sistema
Rendimiento	Es el número de procesos terminados por unidad de tiempo. Este valor depende de la longitud promedio de los procesos, pero también se ve afectado por la política de planificación.
Utilización del procesador	Es el porcentaje de tiempo que el procesador está ocupado. Es un criterio importante para los sistemas de tiempo compartido; en sistemas monousuario y de tiempo real este criterio tiene menos importancia que otros.
	Otros criterios orientados al sistema
Equidad	Los procesos deberían tratarse de la misma forma, y ninguno debería quedar relegado.
Prioridades	Se favorece a los procesos de prioridad más alta.
Ocupación de recursos	Se deben mantener los recursos del sistema ocupados.

Cuadro 4.13: Criterios de planificación orientados al sistema

4.4.2. Evaluación analítica

Este tipo de evaluación utiliza el algoritmo y la carga del sistema para producir una fórmula o número que evalúa el rendimiento del algoritmo para esa carga.

4.4.2.1. Modelo determinista

Partiendo de una carga particular determina el rendimiento que obtendría con cada algoritmo. Es un método simple pero requiere conocer con exactitud la carga que tendrá el sistema.

El ejemplo que hemos ido desarrollando durante el estudio de los diferentes algoritmos de planificación es una muestra de este tipo de evaluación. En la tabla 4.14 podemos ver una comparativa de los resultados obtenidos para todos los algoritmos, donde t_r es el tiempo de respuesta, t_e es el tiempo de espera, t_R es el tiempo de retorno y t_{RN} el tiempo de retorno normalizado. Así, el algoritmo SRT obtiene mejores resultados que los demás excepto en el tiempo de respuesta, donde el realimentado con un cuanto de 1 en todos los niveles obtiene mejor resultado.

Algoritmo	t_r	t_e	t_R	t_{RN}	Media
FIFO	6,4	6,4	10,8	3,12	6,68
SPN	4,4	4,4	8,8	2,83	4,91
SRT	3,4	**3,8**	**8,2**	**1,60**	**4,25**
Prioridades no apr.	4,4	4,4	8,8	2,03	4,91
Prioridades apr.	3,0	5,0	9,4	2,07	4,87
RR (q=2)	2,6	7,6	12,0	2,81	6,25
RR (q=4)	4,2	7,0	11,4	2,74	6,34
HRRN	5,6	5,6	10,0	2,59	5,95
Real. (q=1)	**0**	8,4	12,8	2,75	5,99
Real. (q=2^n)	0,2	8,4	12,8	2,96	6,09

Cuadro 4.14: Comparación de resultados de distintos algoritmos de planificación

4.4.2.2. Modelos de colas

Los procesos que se ejecutan en un sistema varían de un día para otro por lo que no suele ser posible el empleo del modelo determinista. Sin embargo, es más fácil determinar la distribución de tiempo de CPU y de operaciones de E/S que necesitan los procesos. A partir de estas distribuciones es posible calcular el rendimiento promedio, la utilización de la CPU, el tiempo de espera, etc, para la mayoría de los algoritmos.

En este caso, el sistema de computación se describe como una red de servidores, teniendo asociada una cola de espera cada uno de ellos. La CPU es un servidor con una cola de procesos listos, así como el sistema de E/S

con sus colas de dispositivos. Conociendo las velocidades de llegada y las de servicio, podemos calcular la utilización, la longitud promedio de la cola, el tiempo promedio de espera, etc. Esta área de estudio se conoce como **análisis de colas**, que queda fuera del ámbito de este libro.

4.4.3. Simulación

Se simula mediante software un sistema de computación, donde los principales componentes del sistema se representan mediante estructuras de datos. El simulador tiene una variable que representa al reloj, a medida que el valor de esta variable aumenta, el simulador modifica el estado del sistema para reflejar las actividades de los dispositivos, los procesos y el planificador. Durante la simulación, se generan estadísticas mediante las cuales se realiza la evaluación del algoritmo de planificación.

4.4.4. Implementación

Los métodos anteriores tienen una exactitud limitada. La única forma completamente exacta de evaluar un algoritmo de planificación es codificarlo y ponerlo en el sistema para ver cómo trabaja.

La principal dificultad de este método de evaluación es su costo. El gasto no sólo es el de codificación del algoritmo y la modificación del sistema operativo para soportarlo, sino también la reacción de los usuarios a los continuos cambios en el sistema. Un sistema de máquina virtual podría facilitar este tipo de evaluación.

4.5. Planificación en Linux

Linux dispone de dos algoritmos de planificación de procesos separados. Uno es para los procesos de tiempo real donde prima la prioridad, y otro para el resto de procesos en tiempo compartido.

Para los procesos de tiempo compartido se utiliza un algoritmo de asignación por turnos combinado con prioridades dinámicas, con un cuanto de tiempo de 200 milisegundos. Para evitar que algún proceso no obtenga la CPU durante períodos muy largos, utiliza un contador que se actualiza cada

vez que un proceso abandona el estado de ejecución. Éste es función de la prioridad del proceso y del tiempo que lleva en la cola de listos.

El planificador de procesos para su ejecución el proceso con un contador más alto. Cuando un proceso llega a la cola de listos se comprueba si su contador es mayor que el del proceso en ejecución, en cuyo caso se solicita una reordenación de los procesos listos, para adjudicar de nuevo el control de la CPU.

Cada vez que se agote el cuanto de tiempo, al proceso que acaba de terminar se le decrementa en una unidad su contador. Si los procesos que están listos tiene un contador a 0 se recalcula el valor de éste, según la siguiente regla:

$$contador = contador - anterior/2 + prioridad$$

De este modo, se tiene en cuenta la prioridad y la historia del proceso. Ésta consiste en tomar la mitad del valor del contador la última vez que se recalculó los contadores. Con la influencia de la prioridad, los procesos por lotes que suelen tener menor prioridad que los interactivos, recibirán menos tiempo el uso de la CPU para favorecer la ejecución de los procesos con más prioridad.

Las prioridades van desde -20 a 20, siendo -20 la mayor prioridad. Se establece la prioridad 0 como la asignada a un proceso por omisión. Los usuarios pueden disminuir la prioridad de sus procesos, mientras que el administrador puede aumentarla. El rango de prioridades es:

- -20 a -1 para los procesos que se ejecutan en modo supervisor.
- 0 a 20 para los procesos en modo usuario.

El algoritmo de planificación para los procesos de tiempo real implementado consta de dos algoritmos (de acuerdo a la normativa POSIX.1b): asignación por turnos y FIFO. En ambos casos, el proceso que tenga más prioridad será siempre el que se ejecute. Entre procesos de igual prioridad, se ejecutará el proceso que lleve más tiempo esperando. A estos procesos se les da un valor al contador de 1000 al que se añade su propia prioridad.

En relación a la planificación de tiempo real, el planificador ofrece garantías sobre las prioridades relativas, pero no sobre la rapidez de la ejecución del proceso. Esto hace que sólo sea válido para sistemas de tiempo real blando.

Un pseudocódigo del planificador de procesos aparece en el algoritmo 4.1.

Algoritmo 4.1 Planificador de procesos

```
para todo proceso
hacer
  si el proceso tiene un intervalo de tiempo de ejecución
  entonces
    si el intervalo de tiempo ha expirado
    entonces
      Enviar una señal de alarma (SIGALRM) al proceso
      Si se requiere, restaurar el intervalo de tiempo
    fin si
  fin si
  si el proceso está bloqueado (TASK_INTERRUPTIBLE)
  entonces
    si el proceso ha recibido su señal o ha expirado su tiempo de bloqueado
    entonces
      Cambiar el proceso a estado de listo (TASK_RUNNING)
    fin si
  fin si
fin para
para todo proceso
hacer
  si el proceso está listo (TASK_RUNNING)
  entonces
    Comprobar si es el proceso que tiene el contador más alto
  fin si
fin para
si todos los procesos listos tienen el contador con valor cero
entonces
  para todo proceso
  hacer
    Recalcular el contador de acuerdo a su prioridad
  fin para
fin si
Otorgar el control de la CPU al proceso con el contador más alto
```

4.6. Resumen

La evolución dinámica de los procesos en el sistema consiste en una serie de transiciones entre los diferentes estados por los que puede pasar: nuevo, listo, bloqueado, terminado, ejecución, etc. Los planificadores son los elementos del sistema operativo encargados de realizar estas transiciones.

Existen tres planificadores en el sistema: el de largo plazo, medio plazo y corto plazo. Cada uno tiene un objetivo distinto. El primero es el responsable

de la entrada y salida al sistema, es decir, de la admisión de nuevos procesos.

El planificador a medio plazo tiene la función de suspender o activar procesos en función de las necesidades del sistema. De este modo, estos dos planificadores son los encargados de controlar el grado de multiprogramación.

Finalmente, el planificador a corto plazo se encarga de otorgar el control del procesador a uno de los procesos que se encuentran en estado listo. Este planificador es uno de los elementos que más se ejecutan dentro del sistema operativo, y tiene especial importancia en su rendimiento, debido a que si un proceso pasa a estado bloqueado, sobre él recae la responsabilidad de que el sistema pueda continuar ejecutando otro proceso.

La existencia de numerosos algoritmos para el planificador a corto plazo (FIFO, SPN, SRT, RR, HRRN, etc.) hace necesario establecer qué criterios nos van a permitir evaluarlos para decidir cuál es el más adecuado para nuestro sistema.