2º de Grado en Ingeniería Informática

Sistemas Operativos

Sistemas operativos, apuntes de la asignatura.
Profesor: Juan Carlos Fernández Caballero
email: jfcaballero@uco.es

TEMA 4 – PLANIFICACIÓN

1.Índice de contenidos

1 Objetivos de la planificación

Existen diferentes objetivos que se tratan de conseguir con la planificación. Muchos de ellos son contradictorios entre sí, y hay que llegar a soluciones de compromiso.

La naturaleza o finalidad de cada sistema determinará qué objetivos son los más importantes. Algunos de los más relevantes son:

- **Justicia:** El algoritmo de planificación debe dar una **porción de tiempo de CPU justa a todos los procesos**, ningún proceso debe de acaparar la CPU.
- **Aplazamiento indefinido:** Se debe de **evitar la inanición de los procesos**. Esto ocurre cuando un proceso no recibe nunca servicio por parte del procesador o se posterga durante mucho tiempo.
- **Productividad:** La productividad es la **cantidad de trabajo finalizada por unidad de tiempo**. Es mejor terminar 50 trabajos por unidad de tiempo que terminar 40. Esta medida depende mucho de la "**longitud" de los procesos** o trabajos.

  Un buen planificador debería maximizar la productividad, aunque ello puede conllevar a una sobrecarga por **Costo extra.**
- **Costo extra:** El **tiempo utilizado en la planificación** debe ser el mínimo posible, ya que es tiempo inútil para el sistema.

  Recordemos algunos conceptos, el *dispatcher* es un módulo del SO que da el control de la CPU al proceso seleccionado por el planificador de corto plazo, que se estudiará en las siguientes secciones. El tiempo empleado por el *planificador a corto plazo* y por el *dispatcher* en los salvados/restauración de contexto debe ser el menor posible.
- **Tiempo de respuesta aceptable:** Es importante que los usuarios de un sistema tengan un tiempo de respuesta aceptable ante procesos que lancen, ya sea multiusuario en tiempo real o monousuario. Podría definirse como el tiempo que transcurre desde que se crea o lanza un proceso (estado Nuevo) hasta que se obtienen sus primeros resultados, es decir, hasta que su **primera interacción en estado Ejecutando** se produzca.
- **Degradación aceptable:** Ante una subida de carga en relación a los procesos (muchas peticiones de nuevos procesos o mucho uso de CPU por parte de los mismos), el algoritmo o algoritmos de planificación de un sistema debe/n procurar una degradación del rendimiento de forma suave y sin que ocurra una caída global. Por ejemplo, el **planificador a largo plazo** puede limitar el número de procesos a cargar en RAM, o el **planificador a medio plazo** puede realizar tareas de *swapping*.

2 Criterios para la planificación

Los administradores de los grandes centros de cómputo se basan en una serie de métricas para verificar el desempeño de sus sistemas, las cuales también sirven para comparar el rendimiento de los diversos algoritmos de planificación:

- **Tiempo de estancia, de residencia, o de retorno (*turnaround time*):** Es el tiempo que transcurre desde el momento en que se crea un proceso, estado Nuevo, hasta el momento en que se completa, estado Saliente. **Incluye:** el tiempo de ejecución efectiva o uso de CPU + el tiempo de espera + tiempos de E/S, etc.

- **Tiempo de espera (*waiting time*):** El problema del criterio del tiempo de estancia es que incluye muchos factores que no dependen del algoritmo de planificación, como por ejemplo que se produzcan interrupciones u operaciones de entrada/salida mientras un proceso está ejecutando.

  Por ello definimos el tiempo de espera, como la suma de tiempos que el proceso está esperando a ser servido, o lo que es lo mismo, la suma de tiempos de permanencia en estados distintos al de Ejecución. El objetivo es minimizar el tiempo medio de espera.

- **Utilización de la CPU, eficacia, o tiempo de servicio (*CPU utilization*):** Porcentaje de tiempo que está ocupada la CPU por un proceso.

  Desde el punto de vista de un proceso es la **suma de los tiempos que éste está en la CPU**, o el tiempo que necesita para ejecutarse por completo si no hubiera ningún proceso más en el sistema.

3 Modos de decisión

Los algoritmos de planificación se pueden dividir en dos categorías con respecto a la forma en que manejan los instantes de tiempo en que se ejecuta la selección de un proceso:

1. **No apropiativos o sin expulsión:**

   El planificador selecciona un proceso para ejecutarlo y después deja que se ejecute en CPU hasta que el mismo proceso **se bloquea** (alguna llamada bloqueante o de E/S)**, termina** (normal o de manera abrupta – p.e división por cero -) o se produce una **interrupción** que no tenga que ver con la planificación (p. e. se debe parar para tratar una interrupción que tenga que ver con un operación de E/S anterior de otro proceso). Con este modelo no existe rodaja de tiempo.

2. **Apropiativos o con expulsión:**

   El planificador selecciona un proceso para ejecutarlo y **la decisión de expulsarlo de la CPU se toma si sucede algún tipo de evento debido a la política de planificación** que haga que el proceso actual tenga que ser expulsado, como por ejemplo:

   - El proceso se ha ejecutado por un máximo de tiempo fijo (***quantum*** o rodaja de tiempo). Si sigue en ejecución hasta el final del intervalo o rodaja de tiempo, el proceso se pasa a la lista de Listos y el planificador selecciona otro proceso para ejecutar.
   - Un nuevo **proceso con mayor prioridad llega a la lista de Listos**, ya sea desde estado Nuevo o desde Bloqueado.

   Como **ventaja** pueden proporcionar mejor servicio a la población total de procesos, ya que **previenen** que cualquier proceso pueda monopolizar el procesador durante mucho tiempo y produzca **inanición de procesos**.

   Como **desventaja**, las políticas expulsivas tienen **mayor sobrecarga** que las no expulsivas, ya que hay muchos **más cambios de contexto.**

Independientemente del modo de decisión, a continuación se reorganizan y se muestran algunos **ejemplos que darían lugar a la expulsión de un proceso de la CPU**, y por tanto, a la invocación del planificador:

- Se produce una **interrupción** (por parte del controlador de E/S) proveniente de una E/S anterior, de forma que se interrumpe la ejecución del proceso actual para tratar la interrupción.
- Por la razón que sea el proceso actual pasa a **estado Bloqueado**, por ejemplo:
  - En determinadas llamadas al sistema que provoquen una E/S, como por ejemplo *open()*, *read()*, *etc,* de forma que el proceso llamante pasa a estado Bloqueado.
  - Ante llamadas bloqueantes, como cuando un proceso se bloquea a espera de que termine un hijo o un hilo mediante la invocación de *waipid() o pthread_join()*, o incluso ante mecanismos de exclusión mutua como los semáforos (*semWait()*), los cuales pueden bloquear el proceso actual.

4 Tipos de planificación

El objetivo de la planificación de procesos es asignar procesos a ser ejecutados por el procesador a lo largo del tiempo, de forma que se cumplan los objetivos del sistema tales como el tiempo de respuesta, la productividad, etc.

En muchos sistemas, esta actividad de planificación se divide en tres funciones independientes: **planificación a largo, medio, y corto plazo:**

- La planificación a **largo plazo** se realiza cuando se crea un nuevo proceso, de forma que hay que decidir si se añade un nuevo proceso al conjunto de los que están activos actualmente. Por tanto, el planificador a largo plazo determina qué programas se admiten en el sistema para su procesamiento, controlando el grado de multiprogramación.

  Cuanto mayor sea el número de procesos creados y cargados en RAM, menor será el porcentaje de tiempo en que cada proceso se pueda ejecutar, es decir, más procesos compiten por la CPU. Así, el planificador a largo plazo puede limitar el grado de multiprogramación a fin de proporcionar un servicio satisfactorio al actual conjunto de procesos cargados en RAM, de forma que cada vez que termine un proceso, el planificador puede decidir si añadir o no, uno o más trabajos Nuevos a la lista de Listos.

  Por otro lado, si la fracción de tiempo que el procesador está ocioso excede un determinado valor, se puede invocar al planificador a largo plazo para traer procesos desde estado Nuevo a estado Listo (siempre que haya procesos en estado Nuevo, claro está).
- La planificación a **medio plazo** es parte de la función de intercambio o memoria virtual (*swapping*). En ocasiones hay que decidir si se añade un proceso a la memoria principal o por el contrario, por razones de rendimiento, se añade a memoria de disco o *swapp*.
- La planificación a **corto plazo** conlleva decidir cuál de los procesos Listos para ejecutar es el siguiente a introducir en la CPU. Éste planificador se invoca siempre que ocurre un evento que puede conllevar su retirada de la CPU. En este caso hay que expulsar al proceso actualmente en ejecución en favor de otro, que dependerá de la política de planificación.

En términos de **frecuencia de ejecución:**

- El **planificador a largo plazo** ejecuta con relativamente **poca frecuencia** y toma la decisión de grano grueso de admitir o no un nuevo proceso y qué proceso admitir.
- El **planificador a medio plazo**, en caso de que haya razones de rendimiento (poco frecuente) se utiliza para tomar decisiones de **intercambio**. Suponga que el sistema dispone de poca memoria principal, se encuentre muy llena, o se necesite pasar de estado Nuevo a estado Listo un proceso que ocupe mucha RAM. En este tipo de casos se pasarían otros procesos de la lista de Listos o Bloqueados a *Swap* (estado Suspendido).

  En los sistemas actuales a nivel de usuario el *swapping* se usa cada vez menos, ya que se dispone de gran cantidad de memoria RAM para alojar a procesos, por lo que hay **muy poca frecuencia de uso**. Por otro lado, el que un sistema tenga mucha RAM no implica necesariamente mayor velocidad, ya que si el planificador a largo plazo pone muchos procesos en el sistema, estos tendrán cada vez menos tiempo de CPU asignado, y el tiempo de respuesta será mayor.
- El **planificador a corto plazo** se ejecuta mucho más **frecuentemente** y toma las decisiones de grano fino sobre qué proceso ejecutar el siguiente.

5 Tipos de procesos

De forma genérica, la decisión de cuál es el siguiente trabajo a admitir para la CPU puede basarse en encontrar un compromiso entre **procesos limitados por el procesador** y **procesos limitados por la E/S:**

- Se dice que un **proceso está limitado por el procesador** si realiza mucho trabajo computacional y solo muy ocasionalmente o incluso nunca, usa los dispositivos de E/S.
- Se dice que un **proceso está limitado por la E/S** si se pasa más tiempo utilizando los dispositivos de E/S que el procesador.

La relación de compromiso entre estos dos tipos genéricos de procesos puede ser tomada dependiendo de los recursos de E/S que vayan a ser utilizados, de forma que se intente equilibrar el uso de la E/S.

6 Primero en llegar, primero en servirse (first-come-first-served, FCFS)

La política de planificación más sencilla se llama primero en servirse (FCFS) o primero llega, primero sale (FIFO). En el momento en que un proceso pasa al estado de Listo, se une a una única cola de Listos. Cuando el proceso actual en ejecución deja de ejecutar, se selecciona para ejecutar el proceso siguiente que ha estado más tiempo en la cola de Listos.

Tenga en cuenta que en este algoritmo de planificación **no existen las rodajas de tiempo**, es un **modelo de decisión no expulsivo**, por lo que **un proceso se puede ejecutar entero** (termina de manera normal o abrupta), hasta que realice un operación que lo lleve a estado Bloqueado, o hasta que se produzca una interrupción que no tenga que ver con la planificación y que haya que tratar.

En caso de que un proceso se bloquee, pasaría a la lista de Bloqueados en espera de un evento, de forma que cuando ocurra el evento pasará a la lista de Listos.

Considérese el siguiente **ejemplo de procesos limitados por el procesador**, donde un proceso corto va tras uno largo, pero antes recuerde las siguientes definiciones:

- *Tiempo de Estancia = Tiempo Finalización - Tiempo Llegada.*
- *Tiempo de Espera = Tiempo de Estancia – Tiempo de CPU o de servicio.*
- *Tiempo de Estancia Normalizado = Tiempo de estancia / Tiempo de CPU o de servicio.*

  Se podría definir el ***tiempo de Estancia Normalizado*** como el valor que indica la proporción de tiempo de más que un proceso está en el sistema hasta su finalización comparado con el tiempo que estaría si estuviera el solo.

Proceso	Tiempo de Llegada	Tiempo de Servicio (T_s)	Tiempo de Comienzo	Tiempo de Finalización	Tiempo de Estancia (T_r)	T_r/T_s
W	0	1	0	1	1	1
X	1	100	1	101	100	1
Y	2	1	101	102	100	100
Z	3	100	102	202	199	1,99
Media					100	26

Analizada la tabla anterior, se dan las siguientes características en FCFS:

1. El tiempo de estancia normalizado para el proceso **Y** es excesivamente grande en comparación con los otros procesos. El valor de 100 significa que el tiempo total que el proceso está en el sistema es 100 veces mayor que el tiempo requerido para su procesamiento. Este **inconveniente** sucederá siempre que llegue un **proceso corto a continuación de un proceso largo** (**Y** llega en el instante 2 y **X** en el instante 1, pero **X** es un proceso largo que va a retrasar a **Y**, que es corto).
2. Otro **inconveniente** de FCFS es que **tiende a favorecer procesos limitados por el procesador** sobre los procesos limitados por la E/S:
   2.1. Considere que hay una colección de procesos, uno de los cuales está limitado por el procesador y un número de procesos limitados por la E/S. Si un proceso limitado por el procesador llega primero y está ejecutando, el resto de los procesos debe esperar. En este punto, la mayor parte de los dispositivos de E/S pueden estar ociosos, incluso aunque exista trabajo potencial que pueden hacer, ya que hasta que no termine de ejecutarse el proceso actual no se servirá el resto.
   2.2. Por otro lado, un proceso limitado por la E/S que pase a estado Bloqueado al principio de su ejecución, no volverá a entrar en Ejecución hasta que se desbloquee y pase a la cola de Listos, habiendo aprovechado poco tiempo la CPU que se le otorgó.
3. Sin embargo, los **procesos largos no van mal**, por ejemplo el proceso **Z** tiene un tiempo de estancia de casi el doble que **Y** (1,99), pero su tiempo de estancia normalizado está por debajo de 2.0. Esto sucede cuando hay **procesos largos tras procesos cortos.**

FCFS **no es una alternativa atractiva por sí misma para un sistema,** sin embargo a menudo se combina con otras políticas para proporcionar una planificación eficaz.

6.1 Ejemplo 1

A continuación se muestra un ejemplo del algoritmo de planificación FCFS:

Tabla de información de los procesos:

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	3	0	3	3-0=3	3-3=0
B	1	5	3	8	8-1=7	7-5=2
C	3	2	8	10	10-3=7	7-2=5
D	9	5	10	15	15-9=6	6-5=1
E	12	5	15	20	20-12=8	8-5=3

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	x	x	x																	
B				x	x	x	x	x												
C									x	x										
D											x	x	x	x	x					
E																x	x	x	x	x

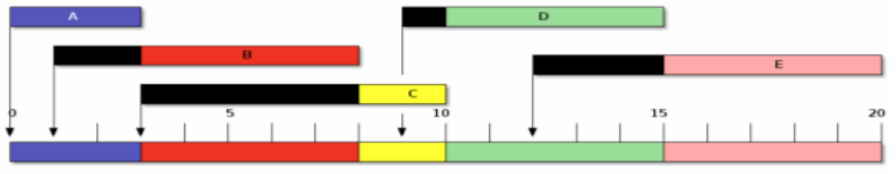

Figura 4: Primero llegado, primero servido (FCFS)

En color negro los procesos que están en el sistema en un estado distinto al de Ejecución.

7 Turno rotatorio (*Round Robin*)

Una forma directa de reducir el castigo que tienen los trabajos cortos después de los largos con FCFS es la utilización de expulsión basándose en el reloj. La política más sencilla es la del turno rotatorio, también denominada **planificación cíclica**.

Se genera una i**nterrupción de reloj cada cierto intervalo de tiempo.** Cuando sucede la interrupción, el proceso actual en ejecución se sitúa en la cola de Listos, y se selecciona el siguiente trabajo según la política FCFS. Esta técnica es también conocida como cortar el tiempo (*time slicing*), porque a cada proceso se le da una rodaja de tiempo antes de ser expulsado. Si antes de finalizar su rodaja de tiempo un proceso invoca a una operación de E/S que requiere sacarlo de la CPU o invoca algún tipo de rutina que lo Bloquee, el proceso **no cumpliría la rodaja entera.**

Con la planificación en turno rotatorio, el tema clave de diseño es la longitud del ***quantum*** de tiempo o rodaja a ser utilizada:

- Si el *quantum* es muy pequeño, el proceso se moverá por el sistema relativamente rápido. Por otra parte, existe una sobrecarga de procesamiento debido al manejo de la interrupción de reloj y por las funciones de planificación y activación (*dispatcher*). De esta forma, **se deben evitar los *quantums* de tiempo muy pequeños.**
- Nótese que en el caso extremo de un ***quantum* de tiempo mayor que el proceso más largo** en el sistema, la planificación en turno rotatorio degenera en FCFS, pudiendo provocar **inanición** (en el sentido de que el último proceso tardaría mucho en entrar en CPU), favoreciendo a los procesos largos y perjudicando a los cortos.

Otra desventaja de la planificación en turno rotatorio es que **trata de forma desigua**l a los procesos limitados por el procesador y a los procesos limitados por la E/S (estos últimos no aprovechan entera su rodaja de tiempo). Generalmente, un proceso limitado por la E/S tiene ráfagas de procesador más cortas que los procesos limitados por el procesador, ya que antes de que se acabe su *quantum*, posiblemente invocarán una operación de E/S que los lleve al estado Bloqueado, quedando a la espera de que termine la operación de E/S que hayan invocado. Por el contrario, un proceso limitado por el procesador generalmente utiliza la rodaja de tiempo completa mientras ejecuta e inmediatamente vuelve a la cola de Listos.

En esta política, a no ser que se diga lo contrario, **hay que tener en cuenta la siguiente suposición**: Si en un mismo instante de tiempo o ciclo de reloj un proceso sale de CPU y otro llega a Listos, el proceso que llega se pone delante del proceso que sale de la CPU.

7.1 Ejemplo 1

A continuación se muestra una simulación Round Robind para *q=4* y *q=1* (termine de completar la tabla con los tiempos de **Inicio**, **Fin**, **Estancia** y **Espera**).

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	3				
B	2	6				
C	4	4				
D	6	5				
E	8	2				

Esquema de ejecución de los procesos en el sistema con *q=4*:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	x	x	x																	
B				x	x	x	x									x	x			
C								x	x	x	x									
D												x	x	x	x					x
E																		x	x	

Esquema de ejecución de los procesos en el sistema con *q=1*:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	x	x		x																
B			x		x		x			x				x				x		
C						x			x				x				x			
D								x				x				x			x	x
E											x				x					

7.2 Ejercicio 1

A continuación se muestra otro ejercicio a resolver del algoritmo de planificación de turno rotatorio *q=1*:

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	3				
B	1	5				
C	3	2				
D	9	5				
E	12	5				

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A																				
B																				
C																				
D																				
E																				

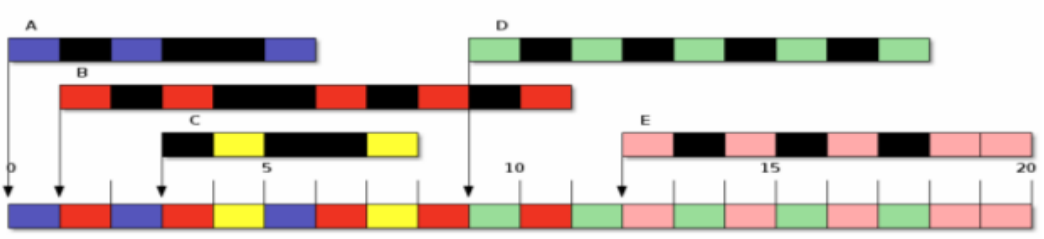

Figura 5: Ronda (*Round Robin*)

7.3 Ejercicio 2

Usando la siguiente tabla de procesos, realice una planificación *Round Robin* con las siguientes suposiciones:

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	3				
B	2	5				
C	4	2				
D	6	5				
E	8	5				

- *q=4.*
- El proceso C hace una operación de E/S en el ciclo 7 (por tanto la finaliza en el 8) y vuelve a estar disponible en Listos en el ciclo 15.

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A																				
B																				
C																				
D																				
E																				

7.4 Turno rotatorio virtual (Virtual Round Robin)

Existe un refinamiento de la planificación en turno rotatorio que denomina **turno rotatorio virtual** (V*irtual Round Robin — VRR —*) y que evita que los procesos limitados por E/S no se vean tan perjudicados.

Los nuevos procesos que llegan se unen a la cola de Listos, que es gestionada con FCFS. Cuando expira el tiempo de ejecución de un proceso, es decir, termina su rodaja de tiempo (decisión expulsiva), vuelve a la cola de Listos.

Cuando se bloquea un proceso por E/S, se une a la cola de Bloqueados por ese tipo de E/S. Hasta aquí, todo como siempre.

La nueva característica es una cola de Listos auxiliar FCFS a la que se mueven los procesos Bloqueados en una E/S. Cuando se va a tomar un proceso por parte del planificador a corto plazo, los procesos en la cola auxiliar tienen preferencia sobre los de la cola de Listos, y cuando se activa un proceso desde la cola auxiliar, éste ejecuta por **un tiempo no superior a lo que resta de su rodaja de tiempo:**

- **Si lo que le resta es 0**, es obvio que ha cumplido anteriormente su **rodaja entera**, por lo que volverá a tener una nueva rodaja. Usaremos esta opción a no ser que se indique lo contrario.
- **Otro esquema** podría ser que si al proceso que sale de Bloqueado le resta 0, que **pase a Listos en vez de a la cola auxiliar.**

La siguiente figura muestra un esquema del turno rotatorio virtual:

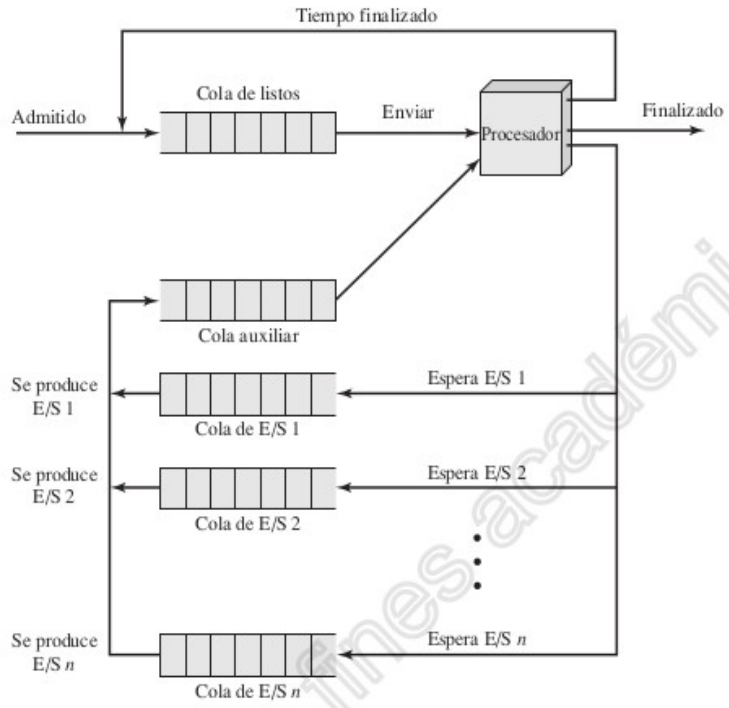

Figura 9.7. Diagrama de encolamiento para el planificador en turno rotatorio virtual.

7.4.1 Ejemplo 1

Dada la política de planificación de Turno Rotatorio Virtual (*Virtual Round Robind o VRR*) con quantum de tiempo q=3, considere los siguientes procesos.

Además tenga en cuenta que en el instante o ciclo 4, el proceso **B** realiza una operación de Entrada/Salida que dura hasta el 5 (1 ciclo, el 4-5), y vuelve a estar disponible para ejecución en el instante o ciclo 9. Por otro lado, en el instante o ciclo 15, el proceso D realiza una operación de Entrada/Salida que dura hasta el 16 (1 ciclo, el 15-16), y vuelve a estar disponible para ejecución en el instante o ciclo 17.

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	3	0	3	3-0=3	3-3=0
B	2	6	3	19	19-2=17	17-6=11
C	4	4	5	15	15-4=11	11-4=7
D	6	5	8	20	20-6=14	14-5=9
E	8	2	12	14	14-8=6	6-2=4

Esquema de ejecución de los procesos en el sistema:

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	x	x	x																	
B				x	x							x					x	x	x	
C						x	x	x							x					
D									x	x	x					x				x
E													x	x						

7.4.2 Ejemplo 2

Dada la política de planificación de Turno Rotatorio Virtual (*Virtual Round Robind o VRR*) con *quantum* de tiempo *q=3*, considere los siguientes procesos.

Además tenga en cuenta que en el instante o ciclo 1, el proceso **A** realiza una operación de Entrada/Salida que dura hasta el 2 (1 ciclo, el 1-2), y vuelve a estar disponible para ejecución en el instante o ciclo 4. Por otro lado, en el instante o ciclo 13, el proceso **D** realiza una operación de Entrada/Salida que dura hasta el 14 (1 ciclo, el 13-14), y vuelve a estar disponible para ejecución en el instante o ciclo 16.

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia (fin-llegada)	Tº Espera (estancia-cpu)
A	0	3	0	6	6-0=6	6-3=3
B	2	6	2	12	12-2=10	10-6=4
C	4	4	6	18	18-4=14	14-4=10
D	6	5	12	20	20-6=14	14-5=9
E	8	2	14	16	16-8=8	8-2=6

Esquema de ejecución de los procesos en el sistema:

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	x	x				x														
B			x	x	x					x	x	x								
C							x	x	x									x		
D													x	x			x		x	x
E															x	x				

7.4.3 Ejercicio 1

Dada la política de planificación de Turno Rotatorio Virtual (Virtual Round Robind o VRR) con *quantum* de tiempo *q=4*, considere los siguientes procesos.

Además tenga en cuenta que en el instante o ciclo 6, el proceso **B** realiza una operación de Entrada/Salida que dura hasta el 7 (1 ciclo, el 6-7), y vuelve a estar disponible para ejecución en el instante o ciclo 9. Por otro lado, en el instante o ciclo 14, el proceso **D** realiza una operación de Entrada/Salida que dura hasta el 15 (1 ciclo, el 14-15), y vuelve a estar disponible para ejecución en el instante o ciclo 17.

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	3				
B	2	6				
C	4	4				
D	6	5				
E	8	2				

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A																				
B																				
C																				
D																				
E																				

7.4.4 Ejercicio 2

Dada la política de planificación de Turno Rotatorio Virtual (Virtual Round Robind o VRR) con *quantum* de tiempo *q=4*, considere los siguientes procesos.

Además tenga en cuenta que en el instante o ciclo 4, el proceso **B** realiza una operación de Entrada/Salida que dura hasta el 5 (1 ciclo, el 4-5), y vuelve a estar disponible para ejecución en el instante o ciclo 7.

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	3				
B	2	6				
C	4	4				
D	6	5				
E	8	2				

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A																				
B																				
C																				
D																				
E																				

8 Planificación mediante colas de prioridades multinivel

En muchos sistemas, a cada proceso se le asigna una prioridad y se le asigna en una cola multinivel, de forma que el planificador siempre elegirá un proceso de prioridad mayor sobre un proceso de prioridad menor.

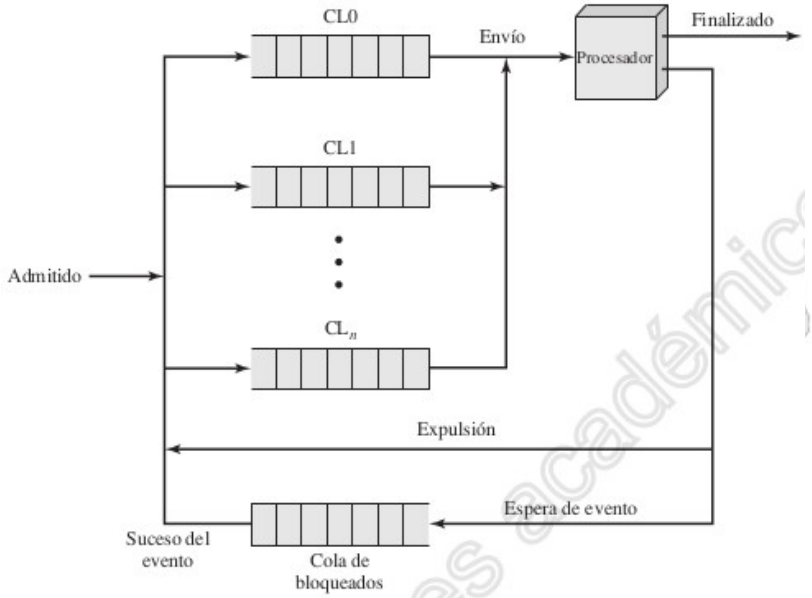

Figura 9.4. Encolamiento con prioridades.

La Figura 9.4 muestra el uso de las prioridades. Por claridad, el diagrama de colas está simplificado, ignorando la existencia de múltiples colas bloqueadas. En lugar de una sola cola de procesos listos para ejecutar, se proporcionan un conjunto de colas en orden descendente de prioridad: *CL0*, *CL1*, …, *CLn*, con la *prioridad[CLi]>prioridad[CLj],* para *i<j*[1].

La prioridad de cada proceso la establece el sistema inicialmente en función parámetros como el propietario del proceso, el tipo del proceso, su tamaño, los recursos que requiere, etc.

Cuando se va a realizar una selección en la planificación, el planificador comenzará en la cola de listos con la prioridad más alta, *CL0*. Si hay uno o más procesos en la cola, se selecciona un proceso utilizando alguna política de planificación, por ejemplo una política FIFO. Si *CL0* está vacía, entonces se examina *CL1*, y así sucesivamente.

Esta política es **expulsiva por rodaja de tiempo o *quantum***, de forma que un proceso se expulsa de la CPU si se cumple su rodaja de tiempo asignada, se produce una interrupción, o se invoca a una operación que lo lleve a estado Bloqueado. Cuando se produce su expulsión sale de la cola *CLi*, de forma que cuando vuelve al estado de Listo, se sitúa en *CLi* nuevamente.

Con esta política, habrá tantas colas en el sistema como prioridades, a no ser que se agrupen rangos de prioridades en colas. Como la prioridad de un proceso no cambia durante su ejecución y ésta se usa para planificar a corto plazo, se le llama una política de **prioridades estáticas**.

1 En UNIX y otros muchos sistemas, los valores mayores de prioridad representan procesos de prioridad más baja; a menos que se especifique, seguiremos esta convención. Algunos sistemas, tales como Windows, utilizan la convención opuesta: un número mayor significa una mayor prioridad

Un problema de las prioridades estáticas es que algunos parámetros, fundamentalmente los referidos al consumo de recursos y en particular al tiempo de CPU no se conocen *a priori*, por lo que la prioridad puede no ser indicativa de comportamiento del programa. Esto es así porque la prioridad se asigna al crearse el proceso, según del tipo que sea.

Otro problema de las colas multinivel con prioridades estáticas es que un proceso con **baja prioridad puede llegar a inanición** si hay continuamente procesos con mayor prioridad listos para ejecutarse.

En esta política hay que tener en cuenta lo siguiente:

- Si la **expulsión** de un proceso se realizase **antes de cumplir su rodaja de tiempo**, el proceso no cumpliría la rodaja entera y si volviese a Listos no ejecutaría el tiempo restante **sino que comenzaría la rodaja de nuevo.**
- En el caso de que un proceso en ejecución **salga de la CPU a la misma vez que llega un nuevo proceso**, entonces el nuevo proceso se añade a la lista de Listos antes que el proceso que termina.
- Además, si un proceso termina y no hay **ningún proceso con mayor prioridad ni tampoco hay un proceso delante de el con la misma prioridad**, **el proceso que termina sigue en CPU**.

8.1 Ejemplo 1

Si se considerase ***q=2,*** la tabla de tiempos y la tabla que muestra la ejecución de cada proceso serían las siguientes (se sobreentiende que hay 3 colas, una por prioridad):

- Suponga que un valor de **1 da mayor prioridad que un valor de 2**, y así sucesivamente.

Proceso	Prioridad	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	2	0	3	0	7	7-0=7	7-3=4
B	2	1	5	2	10	10-1=9	9-5=4
C	1	3	2	4	6	6-3=3	3-2=1
D	3	9	5	10	20	20-9=11	11-5=6
E	1	12	5	12	17	17-12=5	5-5=0

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	x	x					x													
B			x	x				x	x	x										
C					x	x														
D											x	x						x	x	x
E													x	x	x	x	x			

8.2 Ejercicio 1

A continuación se muestra un ejemplo de colas multinivel estáticas con ***q=1*** (se sobreentiende que hay 3 colas, una por prioridad)***:***

- Suponga que un valor de **1 da mayor prioridad que un valor de 2**, y así sucesivamente.

Tabla de información sobre los procesos:

Proceso	Prioridad	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	2	0	3				
B	2	1	5				
C	1	3	2				
D	3	9	5				
E	1	12	5				

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A																				
B																				
C																				
D																				
E																				

9 Colas de prioridades multinivel retroalimentada (feedback)

Una variación de las colas con prioridades multinivel es la **retroalimentación multinivel o *feedback*.** En esta política de planificación hay fijado un número máximo de colas con prioridad. Cuando un proceso llega al sistema se sitúa en la primera cola, CL0.

Cuando se va a realizar una selección en la planificación, el planificador comenzará en la cola de listos con la prioridad más alta, CL0. Si hay uno o más procesos en la cola, se selecciona un proceso utilizando alguna política de planificación, normalmente una política FIFO. Si CL0 está vacía, entonces se examina CL1, y así sucesivamente.

Un proceso se expulsa de la CPU si se cumple su rodaja de tiempo asignada, si invoca a una operación que lo lleve a estado Bloqueado o si se produce una interrupción. A diferencia de las colas con prioridades básica, **si un proceso se expulsa de una cola *Cli* (aunque sea por una E/S), automáticamente pasará a la siguiente cola con menor prioridad. Si no ha cumplido su rodaja de tiempo no ejecutará lo que le reste, sino que comienza con una rodaja nueva.**

Cuando un proceso cambia su nivel de prioridad a lo largo de su estancia en el sistema se dice que se tienen **prioridades dinámicas.**

Cuando un proceso llega a la cola máxima o de menor prioridad no puede descender más, por lo que es devuelto a esta cola repetidas veces hasta que se consigue completar. De esta forma, la cola con menor prioridad se trata con una política de turno rotatorio o *Round Robin*.

Esquema de colas de prioridades multinivel retroalimentada:

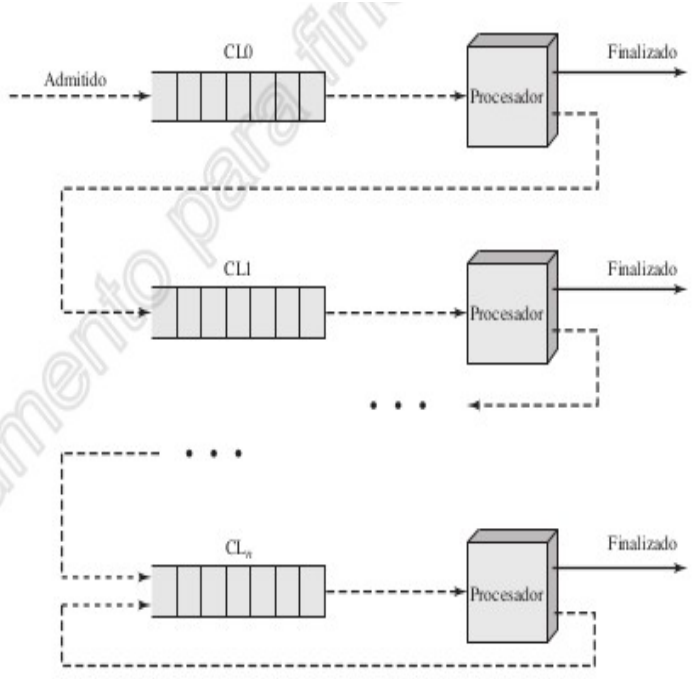

Figura 9.10. Planificación retroalimentada.

Problemas que presenta esta política de planificación:

Un esquema de colas multinivel con valores de "*q*" pequeños (por ejemplo *q=1)* puede provocar que:

- El **tiempo de estancia** para procesos más **largos** se puede **alargar de forma alarmante.**
- Puede ocurrir **inanición** para procesos más **largos si están entrando nuevos trabajos frecuentemente** en el sistema, ya que el planificador siempre empieza la asignación de procesos de CPU por la cola CL0.

9.1.1 Ejemplo 1

Este ejemplo muestra la resolución de un problema con *q=1* y 3 colas, la última con feedback:

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	3	0	11	11-0=11	11-3=8
B	2	6	2	20	20-2=18	18-6=12
C	4	4	4	16	16-4=12	12-4=8
D	6	5	6	19	19-6=13	13-5=8
E	8	2	8	10	10-8=2	2-2=0

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	x	x									x									
B			x	x								x			x			x		x
C					x	x							x			x				
D							x	x						x			x		x	
E									x	x										

9.1.2 Ejercicio 1

Use la misma tabla de procesos anterior con **3 colas**, la ultima *feedback* y con ***q=2.***

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	3				
B	2	6				
C	4	4				
D	6	5				
E	8	2				

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A																				
B																				
C																				
D																				
E																				

9.2 Versión con $q=2^i$

Para compensar el problema del quantum $q = 1$, se podrían variar los tiempos de expulsión de cada cola, de manera que un proceso de la cola *CL0* tiene una rodaja de una unidad de tiempo; un proceso de la cola *CL1* tiene una rodaja de dos unidades de tiempo, y así sucesivamente.

En general, a **un proceso de la cola *CLi* se le permitiría ejecutar 2^i ($q=2^i$) unidades de tiempo** antes de ser expulsado.

9.2.1 Ejercicio 2

Como ejercicio pruebe a usar la misma tabla de procesos anterior con esquema $q=2^i$, para un sistema de 2 colas (la ultima *feedback*), siendo $i=0$ la cola inicial con mayor prioridad.

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	3				
B	2	6				
C	4	4				
D	6	5				
E	8	2				

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A																				
B																				
C																				
D																				
E																				

Pruebe este mismo ejercicio suponiendo que B hace una E/S en el ciclo 11 y vuelve a estar disponible en el ciclo 15.

9.2.2 Ejercicio 3

Como ejercicio pruebe a usar la misma tabla de procesos anterior con esquema $q=2^i$, para un sistema de 3 colas (la última *feedback*), siendo *i=0* la cola inicial con mayor prioridad.

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	3				
B	2	6				
C	4	4				
D	6	5				
E	8	2				

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A																				
B																				
C																				
D																				
E																				

10 Primero el proceso más corto (shortest process next – SPN)

Otro enfoque para reducir el sesgo a favor de los procesos largos en FCFS es la política **primero el proceso más corto (SPN)**, también conocida como ***Shortest job nex*t (SJN)** y ***Shortest Job First*** **(SJF)**.

SPN es una política no expulsiva en la que se selecciona el proceso con el tiempo de procesamiento más corto esperado, es decir, el proceso que necesite menos CPU o tiempo de servicio para ejecutarse por completo y salir del sistema.

De esta forma un proceso corto se situará a la cabeza de la cola de Listos, por delante de los procesos más largos. Una vez dentro de la CPU un proceso se ejecuta entero si no se Bloquea, es decir, aunque lleguen nuevos procesos a la cola de Listos, los tiempos de cada proceso no se recalculan. Esto se hace cuando el proceso actual termine, para determinar cuál es el proceso más corto de los que hay en ese momento en la lista de Listos.

En caso de que un proceso se Bloquee, pasaría a la lista de Bloqueados en espera de un evento, al igual que ocurre con la política FCFS. Cuando ocurra el evento pasará a la lista de Listos con un **tiempo de servicio igual al restante que le quede por cumplir**.

En el caso de que dos o más procesos tengan el **mismo tiempo de servicio** se le da **prioridad al proceso que lleve mas tiempo en el sistema, es un comportamiento FIFO.**

Con la política SPN es necesario contar con **información por anticipado acerca del tiempo de CPU** que requieren los procesos que forman la lista de Listos. **En las politicas anteriores no se tenía en cuenta el Tº de CPU para tomar una decisión por parte del planificador**, de forma que en los ejercicios que ha realizado el lector se usaba el tiempo de CPU para calcular el tiempo de Espera y para saber cuándo un proceso llegaba a estado Saliente. Al planificador no le hacía falta saber ese dato, solo le hacía falta al lector para poder simular los ejercicios planteados.

Ahora bien, es muy difícil contar con esta información antes de ejecutar el proceso, con lo que normalmente se estima mediante:

- Modelos matemáticos que asignen un tiempo de servicio a un proceso dependiendo de su tipo, carga del sistema, recursos que necesite, historia de ejecución anteriores, etc.

Al trabajar con estimaciones en lugar de duraciones reales, el éxito de la planificación dependerá de lo adecuado que sea el modelo matemático de estimación.

De forma general, con SPN se pueden dar los siguientes **problemas**:

- Hay **posibilidad de inanición** para los procesos más largos si hay una llegada constante de procesos más **cortos mientras el largo está encolado.**
- Hay posibilidad de **penalizar fuertemente a un proceso corto tras uno largo** que ya está ejecutando, debido a la **carencia de expulsión.**

10.1 Ejemplo 1

La siguiente Tabla muestra el resultado de un ejemplo con procesos limitados por el procesador:

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	3	0	3	3-0=3	3-3=0
B	2	6	3	9	9-2=7	7-6=1
C	4	4	11	15	15-4=11	11-4=7
D	6	5	15	20	20-6=14	14-5=9
E	8	2	9	11	11-8=3	3-2=1

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	x	x	x																	
B				x	x	x	x	x	x											
C												x	x	x	x					
D																x	x	x	x	x
E										x	x									

10.2 Ejemplo 2

A continuación se muestra otro ejemplo de la política SPN:

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	3	0	3	3	0
B	1	5	5	10	9	4
C	3	2	3	5	2	0
D	9	5	10	15	6	1
E	12	5	15	20	8	3

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	x	x	x																	
B						x	x	x	x	x										
C				x	x															
D											x	x	x	x	x					
E																x	x	x	x	x

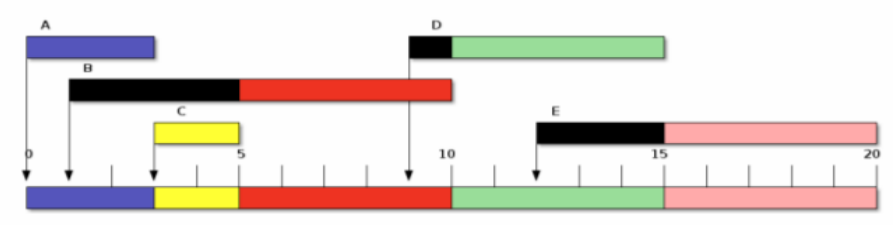

Figura 8: El proceso más corto a continuación (SPN)

10.3 Ejercicio 1

Pruebe a realizar un ejercicio con las siguientes suposiciones:

- El proceso B hace una operación de E/S en el ciclo de reloj 4 (se entiende que la finaliza en el 5).
- B vuelve a estado Listo en el ciclo 9.

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	3				
B	2	6				
C	4	4				
D	6	5				
E	8	2				

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A																				
B																				
C																				
D																				
E																				

10.4 Ejercicio 2

Pruebe a realizar un ejercicio con las siguientes suposiciones:

- El proceso B hace una operación de E/S en el ciclo de reloj 8 (la finaliza en el 9).
- B vuelve a estado Listo en el ciclo 12.

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	3				
B	1	5				
C	3	2				
D	9	5				
E	12	5				

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A																				
B																				
C																				
D																				
E																				

11 Menor tiempo restante (shortest remaining time – SRT)

SPN puede también implementarse con una decisión **expropiativa**. Dicha política de planificación se conoce como menor tiempo restante (SRT), y **expulsa procesos sin tener en cuenta su rodaja de tiempo.**

De esta forma, si un proceso se une a la lista de procesos Listos con menor CPU restante o estimada, se le quita la CPU al proceso actual para asignarla al nuevo proceso más corto. Así, al igual que con SPN, **el planificador debe tener una estimación del tiempo de servicio o CPU** de los procesos que pasan a la lista de Listos por primera vez.

Si los procesos que van llegando a la lista de Listos no tienen tiempos restantes menores que el proceso que actualmente está en ejecución éste último no se expulsa, ya que no hay asignación de *quantum* o rodaja de tiempo.

La diferencia entonces respecto a SPN es que, mientras que en SPN una vez que termina el proceso que actualmente está ejecutando, es cuando se comprueba la lista de Listos para encontrar el proceso más corto, **en SRT la lista de Listos se comprueba cada vez que llegan procesos a ésta.**

En el caso de que dos o más procesos tengan el **mismo tiempo de servicio restante** se le da **prioridad al proceso que haya llegado antes a la lista de Listos (lleva mas tiempo en el sistema, es un comportamiento FIFO).**

En SRT e**xiste riesgo de inanición para los procesos más largos**.

A diferencia del turno rotatorio, **no se generan interrupciones adicionales por finalización de rodaja de tiempo**, reduciéndose la sobrecarga en ese aspecto **si no llegan procesos** a la lista de Listos con **menor tiempo de servicio**.

Por otra parte, **se deben almacenar los tiempos de servicio transcurridos**, generando **sobrecarga**, además habría que interrumpir el proceso actual en ejecución para calcular el tiempo restante de los nuevos procesos que llegan a la cola de Listos (no se muestra en los ejercicios de simulación por simplificar su entendimiento).

Las políticas SPN y SRT **no se pueden implantar en sistemas en los que no se conoce *a priori* el tiempo de CPU** que requieren los procesos situados en la lista de Listos.

11.1 Ejemplo 1

La siguiente Tabla muestra el resultado de un ejemplo de procesos limitados por el procesador:

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	3	0	3	3-0=3	3-3=0
B	2	6	3	15	15-2=13	13-6=7
C	4	4	4	8	8-4=4	4-4=0
D	6	5	15	20	20-6=14	14-5=9
E	8	2	8	10	10-8=2	2-2=0

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	x	x	x																	
B				x							x	x	x	x	x					
C					x	x	x	x												
D																x	x	x	x	x
E									x	x										

Fíjese que en **el instante 10, tanto el proceso B como D tienen el mismo tiempo restante**, pero como B entró en la lista de Listos antes que D, se le da la prioridad a B.

11.2 Ejemplo 2

A continuación se muestra otro ejemplo más de esta política de planificación:

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	7	0	12	12-0=12	12-7=5
B	2	4	2	7	7-2=5	5-4=1
C	4	1	4	5	5-4=1	1-1=0

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11
A	x	x						x	x	x	x	x
B			x	x		x	x					
C					x							

11.3 Ejercicio 1

Tomando la información de la siguiente tabla, calcule el esquema de ejecución y los tiempos de inicio, Fin, Estancia y Espera, teniendo en cuenta las siguientes suposiciones:

- El proceso C hace una operación de E/S en el ciclo de reloj 5 (se entiende que la finaliza en el 6).
- C vuelve a estado Listo en el ciclo 10.

Proceso	Llegada	Tº CPU	Tº Inicio	Tº Fin	Tº Estancia	Tº Espera
A	0	3				
B	2	6				
C	4	4				
D	6	5				
E	8	2				

Esquema de ejecución de los procesos en el sistema:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A																				
B																				
C																				
D																				
E																				

12 Comparación de rendimiento

Claramente, el rendimiento de las políticas de planificación es un factor crítico en su elección. Sin embargo, es imposible hacer una comparación definitiva, porque el rendimiento relativo dependerá de multitud de factores, que incluyen la distribución de probabilidad de los tiempos de servicio de varios procesos, la eficiencia de la planificación y del mecanismo del cambio de contexto, la naturaleza de las demandas de E/S y el rendimiento de los subsistemas de E/S. Por tanto, dependiendo de estos factores, lo ideal sería que el sistema aplicase una política de planificación u otra.

Por otro lado, no es lo mismo un sistema de propósito general que sistemas de cómputo de propósito específico, donde sabemos qué tipo de procesos va a haber en el sistema y para qué se va a utilizar fundamentalmente. En éste último caso la elección de una política de planificación es algo más sencillo y concreto.

Consulte la bibliografía básica y la Web si está interesado en estudiar la política de planificación utilizada en los diferentes sistemas operativos existentes.

13 Bibliografía

El contenido de este documento se ha elaborado principalmente a partir de las siguientes referencias bibliográficas, y con propósito meramente académico y no lucrativo:

- W. Stallings. *Sistemas operativos, 5ª edición.* Prentice Hall, Madrid, 2005.
- A. S. Tanenbaum. *Sistemas operativos modernos, 3a edición*. Prentice Hall, Madrid, 2009.
- A. Silberschatz, G. Gagne, P. B. Galvin. *Fundamentos de sistemas operativos, séptima edición*. McGraw-Hill, 2005.
- A. Mclver, I. M. Flynn. *Sistemas operativos, 6ª edición*. Cengage Learning, 2011.
- J. A. Alamansa, M. A. Canto Diaz, J. M. de la Cruz García, S. Dormido Bencomo, C. Mañoso Hierro. *Sistemas operativos, teoría y problemas*. Editoral Sanz y Torres, S.L, 2002.
- F. Pérez, J. Carretero, F. García. *Problemas de sistemas operativos: de la base al diseño, 2ª edición*. McGraw-Hill. 2003.
- S. Candela, C. Rubén, A. Quesada, F. J. Santana, J. M. Santos. *Fundamentos de Sistemas Operativos, teoría y ejercicios resueltos*. Paraninfo, 2005.
- J. Aranda, M. A. Canto, J. M. de la Cruz, S. Dormido, C. Mañoso. *Sistemas Operativos: Teoría y problemas*. Sanz y Torres S.L, 2002.
- J. Carretero, F. García, P. de Miguel, F. Pérez, *Sistemas Operativos: Una visión aplicada.* Mc Graw Hill, 2001.