Algoritmos de planificación de procesos

En sistemas operativos multitarea los procesos aparentemente se ejecutan a la vez, pero si solo hay un único procesador, este se tiene que distribuir entre todos los procesos que se estén ejecutando.

A la forma en que la CPU se distribuye para ejecutar los procesos se le llama planificación. Existen distintos algoritmos de planificación.

El planificador a corto plazo (dispatcher) decide qué proceso de los que están preparados para ejecutarse pasa a utilizar el procesador. El planificador debe intentar minimizar el tiempo de respuesta, maximizar la cantidad de procesos que se ejecuten y evitar que un proceso quede postergado indefinidamente. Debe intentar tratar a los procesos por igual salvo en caso de que haya procesos que tengan mayor prioridad, aunque también debe evitar que procesos muy prioritarios pospongan indefinidamente proceso menos prioritarios, mediante técnicas como el envejecimiento (aging).

Los estados en los que se puede encontrar un proceso son:

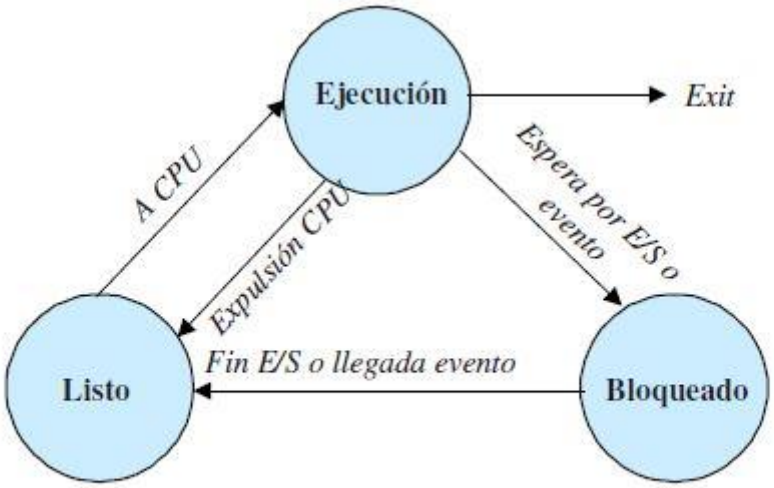
- Listos, en espera o preparados: el proceso está preparado para ejecutarse, es decir, está en espera de que el proceso que se está ejecutando deje libre la CPU. El proceso está a la espera de que se le asigne la CPU.
- Bloqueados: Está esperando a un evento para que se complete su ejecución. Por ejemplo el proceso está esperando unos recursos que está siendo utilizado por otro proceso en ese momento.
- En ejecución o activo: el proceso está ejecutando sus instrucciones en ese momento, es decir, está ocupando la CPU.

Transiciones de estado entre procesos

Cuando un proceso pasa de un estado a otro, se produce una transición de estado. Podemos tener cuatro transacciones:

1. De ejecución a bloqueado: por ejemplo al iniciar una operación de E/S.
2. De ejecución a listo: por ejemplo, en un sistema que utilice el tiempo compartido, cuando el proceso que ocupa la CPU lleva demasiado tiempo ejecutándose continuamente, el sistema operativo decide que otro proceso ocupe la CPU, pasando el proceso que ocupaba la CPU a estado listo.
3. De listo a en ejecución: cuando lo requiere el planificador de la CPU.
4. De bloqueado a listo: se dispone del recurso por el que se había bloqueado el proceso. Por ejemplo, termina la operación de E/S.

De las cuatro transiciones de estado posibles, la única iniciada por el proceso de usuario es el bloqueo, las otras tres son iniciadas por entidades externas al proceso.



Entre las distintas planificaciones existen dos tipos principales:

- Expulsiva (preemptive): un proceso puede ser desalojado de la CPU sin que haya finalizado, y se queda en la cola de proceso en espera.
- No expulsiva (non-preemptive): un proceso no puede ser desalojado de la CPU hasta que termina o se bloquea porque necesita un recurso no disponible.

A la operación de desalojar un proceso de la CPU para que otro empiece a ejecutarse se le llama **cambio de contexto**. Esta operación debe realizarse con la mayor rapidez posible y en ella se debe almacenar la información sobre el estado de la ejecución del proceso, para que cuando vuelva a estar en ejecución, se reinicie desde el mismo punto y con los mismos valores en que se desalojó.

Cuando un proceso está en estado activo es posible que monopolice la CPU (de forma intencionada o no). Para evitar esto el sistema operativo hace uso de un reloj de interrupciones hardware. Al pasar a estado activo un proceso el sistema inicializa el reloj. Cuando pasa un tiempo predeterminado (tiempo máximo que un proceso puede hacer uso de la CPU, y se llama quantum) lo que hace el reloj es generar una interrupción. Cuando el sistema operativo atiende esta interrupción le quita la CPU al proceso (o no si es el único que está listo).

Existen una serie de algoritmos de planificación que le indican al planificador qué proceso se debe elegir para pasar a ejecutarse de los que están en espera. Estos algoritmos deben maximizar la utilización de la CPU, evitando que esté libre en algún momento, y minimizar el tiempo de respuesta de cada proceso.

En cada algoritmo sabremos para cada proceso:

- Tiempo de entrada o de llegada al sistema (T_I): es el momento en que el proceso llega al sistema.
- Tiempo de ejecución (T_X): es el tiempo que el proceso necesita para su ejecución total.

Nos interesa obtener para cada proceso los siguientes datos:

- Tiempo de respuesta o de retorno (T_R): es el tiempo que pasa desde que el proceso llega al sistema hasta que se obtienen los resultados.
- Tiempo de espera (T_E): es el tiempo que el proceso pasa dentro del sistema en espera.
- Además también querremos saber los tiempos medios tanto de respuesta como de espera, para poder saber si para ciertos datos de entrada el algoritmo es más o menos óptimo, en general.

Entre los distintos algoritmos de planificación podemos destacar:

Algoritmo primero en entrar primero en salir: FIFO (First Inpunt First Output) o FCFS (First Come First Served)

Utilizando este algoritmo, los procesos se ejecutan en el orden de llegada. El primero que llega se empieza a ejecutar y los siguientes deberán esperar su turno para poder empezar a ejecutarse.

Proceso	Tiempo de llegada	Tiempo de ejecución
A	0	5
B	1	3
C	2	1

0	1	2	3	4	5	6	7	8
A	A	A	A	A				
					B	B	B	
								C

Desventaja: beneficia a procesos largos.

Algoritmo el más corto primero: SJF (Short Job First)

Este algoritmo coge el proceso más corto de los que están esperando para usar la CPU. En caso de igual tiempo, se aplica el FIFO. Favorece a los procesos que tarden menos tiempo en ejecutarse, pudiendo darse el caso de que los procesos largos no se ejecuten nunca.

Proceso	Tiempo de llegada	Tiempo de ejecución
A	0	5
B	1	3
C	2	1

0	1	2	3	4	5	6	7	8
A	A	A	A	A				
						B	B	B
					C			

Algoritmo el tiempo restante más corto primero: SRTF (Short Remaining Time First)

Este algoritmo va seleccionando de los procesos que están en espera al que le quede menos tiempo estimado para terminar. En caso de empate, se utiliza FIFO. Este algoritmo es expulsivo, ya que si mientras se está ejecutando un proceso llega otro que le quede menos tiempo para acabar que el que está utilizando la CPU lo desplaza, es decir, se produce un cambio de contexto. Favorece a los procesos con menor tiempo de ejecución y mejora los tiempos medios en general. Sin embargo, perjudica a los que necesitan más tiempo.

Proceso	Tiempo de llegada	Tiempo de ejecución
A	0	5
B	1	3
C	2	1

0	1	2	3	4	5	6	7	8
A					A	A	A	A
	B		B	B				
		C						

Algoritmo por prioridades

Este algoritmo consiste en asociar a cada proceso una prioridad. El orden de entrada en la CPU será según la prioridad, y en caso de empate se aplica el FIFO. Existen variantes de este algoritmo, que puede ser expulsivo o no expulsivo, y también se le puede ir cambiando la prioridad a un proceso, es decir, hacerlo más prioritario a medida que lleve más tiempo esperando la CPU

para evitar que procesos poco prioritarios se posterguen indefinidamente si llegan procesos más prioritarios.

La prioridad se suele indicar por un número entero, que cuanto más bajo sea, el proceso será más prioritario. Puede haber más de un proceso con la misma prioridad.

Este algoritmo no favorece a los procesos que necesiten más tiempo de CPU o menos, sino que optimiza el tiempo de respuesta de los más prioritarios.

Proceso	Tiempo de llegada	Prioridad	Tiempo de ejecución
A	0	2	5
B	1	3	3
C	2	1	1

0	1	2	3	4	5	6	7	8
A	A		A	A	A			
						B	B	B
		C						

Algoritmo de operación por rondas: RR (Round Robin)

Este algoritmo va dando tiempo de ejecución a cada proceso que esté en espera. Se debe establecer un tiempo o quantum (q), tras el cual el proceso abandona la CPU y da paso al siguiente. La cola de procesos se estructura como una cola circular siguiendo el orden FIFO. Para este algoritmo es necesario que el cambio de contexto sea muy rápido. Si no hubiese procesos en espera el que está utilizando la CPU seguiría hasta que llegara alguno. Es un algoritmo expulsivo.

Proceso	Tiempo de llegada	Tiempo de ejecución
A	0	5
B	1	3
C	2	1

0	1	2	3	4	5	6	7	8
A	A				A	A		A
		B	B				B	
				C				

Ejecución con quantum = 2.