Sistemas Operativos

Unidad 6: Planificación.

"La planificación a largo plazo no se ocupa de las decisiones futuras sino del futuro con las decisiones actuales."

~ Peter Drucker

Planificación del procesador

El objetivo de la planificación es asignar los procesos al procesador para ser ejecutados. La planificación afecta el rendimiento general del sistema porque decide qué proceso ejecutará y qué procesos deberán esperar. Existen 3 tipos de planificación, largo, medio y corto plazo.

Tipos de planificación

Planificación a largo plazo

El planificador a largo plazo determina qué programas son admitidos en el sistema para su procesamiento. Es quien controla el grado de multiprogramación del sistema. Cuando un programa es admitido, se convierte en un proceso y el mismo es añadido a la cola de procesos del planificador a corto plazo.

En algunos sistemas, un proceso recién creado permanece en disco, en cuyo caso se añade a la cola de procesos del planificador a medio plazo.

El planificador a largo plazo toma 2 decisiones, la primera cuando agregar un nuevo proceso al sistema y la segunda qué proceso agregar.

La primer decisión, cuando agregar un nuevo proceso al sistema, se realiza en base al grado de multiprogramación deseado. Tener en cuenta que a mayor cantidad de procesos creados, menor tiempo de procesador tendrá cada uno. Cada vez que un proceso finaliza, el planificador a largo plazo debe decidir si agrega uno o más procesos al sistema. También puede suceder que el tiempo ocioso del procesador supere un determinado valor y se deba invocar al planificador a largo plazo para que incremente la cantidad de procesos en el sistema.

La segunda decisión es la de qué proceso admitir en el sistema, la misma puede ser tan simple como utilizar una política FIFO o ser un algoritmo más complejo que involucre la prioridad, el tiempo estimado de ejecución, requisitos de E/S.

Los estados de los procesos relacionados con el planificador a largo plazo son los estados *nuevo* y *saliente*.

Planificación a medio plazo

La planificación a medio plazo es parte de la función de intercambio (swap), por ende los estados involucrados con este tipo de planificador son los estados: bloqueado, bloqueado/suspendido y listo/suspendido. Este planificador también participa en la decisión de qué grado de

multiprogramación se desea. Para agregar (traer desde disco) un proceso a memoria se deberá contemplar los requerimientos de memoria de los procesos.

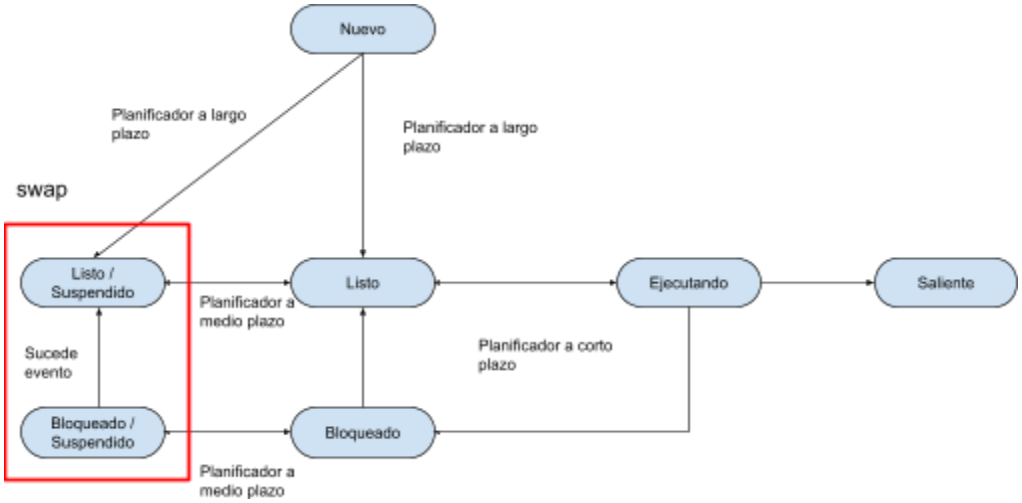
Planificación a corto plazo

El planificador a corto plazo, también llamado activador (dispatcher), es el que toma las decisiones de grano fino sobre qué proceso deberá pasar a utilizar el procesador.

Su objetivo principal es el de optimizar el comportamiento del sistema.

El planificador a corto plazo, es un pequeño programa que intercambia los procesos en el procesador. Se ejecuta siempre que ocurre un evento que puede conducir al bloqueo del proceso actual y que puede proporcionar la oportunidad de expulsar al proceso actual (del procesador) en favor de otro. Estos eventos pueden ser: interrupciones, llamadas al sistema, señales.

Relación entre planificador y los estados de un proceso



Algoritmos de planificación

Criterios

Antes de describir los algoritmos utilizados por el *dispatcher* para planificar la ejecución de procesos, cabe aclarar que existen algunos criterios en los que basarse para evaluar los

algoritmos de planificación. Estos criterios pueden clasificarse en 2 dimensiones, los orientados al usuario y los orientados al sistema.

Los criterios orientados al usuario, se relacionan con el comportamiento del sistema tal y como lo percibe el usuario, mientras que los criterios orientados al sistema se centran en el uso eficiente del procesador.

	Orientado al usuario	Orientado al sistema
Cuantitativo	• **Tiempo de estancia**: tiempo transcurrido desde que se lanza hasta que finaliza el proceso. Es la suma del tiempo de ejecución, tiempo de espera por recursos. • **Tiempo de respuesta**: para un proceso interactivo es el tiempo que transcurre desde que se lanza la petición hasta que se empieza a recibir la respuesta. • **Tiempo límite**: cuando un proceso puede especificar el tiempo límite el planificador debe darle prioridad a dicho proceso y maximizar el número de procesos con tiempo límite terminados.	• **Rendimiento**: apunta a maximizar el número de procesos completados por unidad de tiempo. Es una medida de cuánto trabajo está siendo realizado. • **Utilización del procesador**: es el porcentaje de tiempo que el procesador está ocupado
Cualitativo	• **Previsibilidad**: apunta a que un proceso tarde el mismo tiempo y consuma los mismos recursos independientemente de la carga del sistema.	• **Equidad**: en ausencia de cualquier directiva que oriente la planificación, todos los procesos deben ser tratados de la misma manera, evitando la inanición. • **Imposición de prioridades:** el planificador debe favorecer a los procesos con prioridades más altas. • **Balanceo de recursos**: los recursos del sistema deben permanecer ocupados. Los procesos que utilicen poco los recursos que están sobreutilizados, deberían ser favorecidos.

Los criterios expuestos en la tabla anterior deben ser evaluados a la hora de diseñar un algoritmo de selección de procesos.

Cabe aclarar que estos criterios son independientes y es imposible optimizar todos a la vez. Por ejemplo, disminuir el tiempo de respuesta puede requerir un algoritmo que intercambie procesos con mayor frecuencia, incrementando la sobrecarga del sistema, reduciendo el rendimiento.

Políticas

Cada algoritmo, además de considerar los criterios antes mencionados, implementa una **función de selección** que es la que permite elegir el proceso que debe ser ejecutado. Esta función está basada en alguna política como puede ser la *prioridad*, *requisitos sobre los recursos* o las *características de ejecución del proceso*.

La política de selección también puede contemplar la expulsión de los procesos del procesador.

Políticas sin expulsión: en este tipo de políticas, el proceso que está ejecutando, lo hace hasta que termine o bien hasta que se bloquee esperando por una operación de E/S.

Políticas con expulsión: en este tipo de políticas, el proceso que está ejecutando puede ser interrumpido y pasado al estado *listo* por el sistema operativo.

Las políticas con expulsión, tienen mayor sobrecarga que las no expulsivas, pero pueden proporcionar mejor servicio al sistema en general ya que previenen la monopolización del procesador.

Respecto de las políticas basadas en prioridades de los procesos, cabe aclarar que pueden conducir a una situación de inanición si se aceptan procesos con prioridad más alta que los procesos que están esperando ser ejecutados. Por lo general y para evitar esta situación, los algoritmos que implementan dichas políticas poseen un esquema de modificación de la prioridad del proceso conforme avanza el tiempo.

Primero en llegar, primero en servirse (FCFS)

Este algoritmo implementa un esquema FIFO sobre la cola de procesos. Cuando el proceso que está utilizando el procesador deja de ejecutar (causa de una interrupción, bloqueo, etc) se selecciona para ejecutar el proceso que estuvo más tiempo en la cola de listos.

Este algoritmo proporciona mejor rendimiento en la planificación de procesos largos. Una de las desventajas de este algoritmo es que favorece a los procesos limitados por el procesador sobre los procesos limitados por la E/S.

Definición	Se dice que un proceso está limitado por el procesador si realiza mucho trabajo computacional, es decir, hace un uso intensivo del procesador por sobre el resto de las operaciones. Se dice que un proceso está limitado por la E/S si pasa más tiempo realizando operaciones de E/S que utilizando el procesador.

El hecho de que favorezca los procesos limitados por el procesador se debe en parte a que estos procesos ejecutarán hasta terminar o hasta que se bloqueen por una operación de E/S, mientras que los procesos limitados por la E/S por lo general pasarán más tiempo bloqueados esperando la operación de E/S que usando el procesador. De este análisis también se desprende que tanto el procesador como los dispositivos de E/S estarán por momentos ociosos, desperdiciando tiempo valioso.

Este algoritmo utiliza una política sin expulsión.

Round Robin

Para evitar la penalización que sufren los procesos cortos con FCFS, se puede utilizar un algoritmo como Round Robin basado en las interrupciones de reloj. En este algoritmo cada vez que se produce una interrupción de reloj, el proceso actual es reemplazado por el siguiente en la cola de listos. El proceso que es expulsado vuelve a la cola de listos. La cola de listos se maneja con una política FIFO.

En este tipo de planificación, la clave está en el *quantum* (intervalo de tiempo) a utilizar. Si el quantum es muy chico, habrá una sobrecarga muy grande en el sistema debido a las operaciones de intercambio de procesos. Por otra parte, si el quantum es muy grande, este algoritmo termina pareciéndose a FCFS.

Este algoritmo presenta una desventaja debido a que el tratamiento asignado a los procesos limitados por la E/S no es equitativo con respecto a los procesos limitados por el procesador, ya que estos últimos utilizarán el procesador más veces que un proceso limitado por la E/S. Esto se debe a que los procesos limitados por la E/S estarán bloqueados esperando la operación de E/S mientras que los procesos limitados por el procesador pasarán de *ejecutando* a *listo* y nuevamente a *ejecutando*.

Una alternativa que mejora esta inequidad es el algoritmo Round Robin Virtual en el que se define una cola extra, además de la de *listos*, en la que se colocan los procesos que vuelven del estado *bloqueado*. Al momento de elegir un proceso a ejecutar, los procesos que se encuentren en la nueva cola tendrán prioridad.

Primero el proceso más corto (SPN)

Este algoritmo implementa una política sin expulsión, al igual que FCFS, en el que se selecciona el proceso con el tiempo de procesamiento estimado más corto.

Una complejidad que presenta este algoritmo es conocer el tiempo estimado de procesamiento de un proceso. Este tiempo podría ser indicado por el programador en un sistema por lotes. Este enfoque presenta el problema de que si el tiempo indicado por el programador es muy corto, el sistema expulsará el proceso antes de su ejecución.

En un sistema interactivo puede ser calculado a partir de estadísticas del sistema operativo.

Con este algoritmo, en caso de que exista un ingreso al sistema de procesos cortos, se corre el riesgo de inanición para los procesos más largos y al no posee expulsión, no es un algoritmo adecuado para un entorno de tiempo compartido

Menor tiempo restante (SRT)

Este algoritmo es la versión de SPN pero con una política de expulsión. En este caso el dispatcher elige al proceso con el menor tiempo restante. Cuando un proceso se agrega a la cola de listos podría tener un tiempo restante menor que el proceso que se encuentra actualmente en ejecución. Por lo tanto se producirá la expulsión del mismo, posibilitando la ejecución del proceso nuevo.

A pesar de la política con expulsión, las desventajas de SPN siguen siendo las mismas para este algoritmo.

Feedback

Tanto SPN como SRT requieren conocer el tiempo de servicio de los procesos. Si no se puede obtener, el algoritmo Feedback es una buena alternativa ya que se basa en penalizar a los trabajos que estuvieron más tiempo ejecutando. Esto quiere decir que el algoritmo se basa en el tiempo que cada proceso estuvo ejecutando y no en el tiempo restante.

Para implementar este algoritmo, se definen N colas de prioridad, todas con una política FIFO, desde la 0 a la N donde la cola 0 es la de más prioridad y la N la de menor prioridad. Al igual que en *Round Robin* se define un *quantum* de tiempo para que los procesos utilicen el procesador.

Los procesos ingresan inicialmente a la cola 0 y a medida que van ejecutando y son expulsados van siendo ingresados a una cola de menor prioridad hasta llegar a la cola N.

Cada cola de menor nivel posee un *quantum* de tiempo mayor que la cola inmediatamente anterior.

Otra particularidad de este algoritmo es el tratamiento que otorga a los procesos ante operaciones de E/S. Si el proceso se bloquea por una operación de E/S, al volver del estado bloqueado, es insertado nuevamente en la misma cola en la que estaba antes de bloquearse. Por otra parte si un proceso alcanza la cola de menor nivel y se bloquea ante una operación de E/S, al volver es insertado en la cola inmediatamente superior

Bibliografía utilizada

● William Stallings. Sistemas operativos. Pearson Education. S.A., Madrid, 2005. ISBN-84-205-4462-0.

● AndrewS.Tanenbaum. Modern Operating System. Pearson Education Inc., 2009. ISBN-Q-lB-filBMST-L

● Multilevel Feedback queue. Wikipedia, La enciclopedia libre, 2019 [consulta: 21 de marzo del 2019]. Disponible en https://en.wikipedia.org/wiki/Multilevel_feedback_queue