FACULTAD DE INGENIERÍA

Sistemas Operativos

Planificación

Lic. R. Alejandro Mansilla

Licenciatura en Ciencias de la Computación
Facultad de Ingeniería
Universidad Nacional de Cuyo

Conceptos iniciales

- Escenario más simple, computadora con un único proceso que ejecuta hasta el final. *(consola de videojuegos)*
- Como evolución podemos encontrar el procesamiento por lotes
- Luego se evolucionó hacia la multiprogramación
- Con la aparición del tiempo compartido, se necesito conmutar rápidamente entre procesos, teniendo el SO que decidir cuándo y a qué proceso conmutar.

El código en el sistema operativo que está a cargo de esta tarea es el **planificador.**

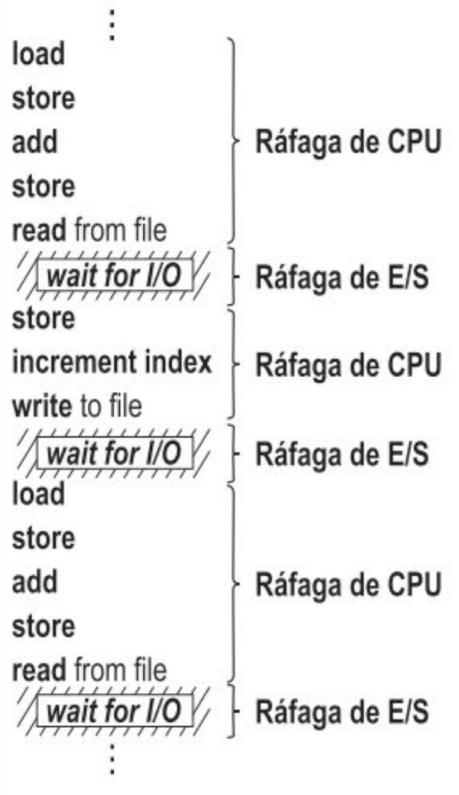
Supuestos subyacentes

Los supuestos básicos detrás de la mayoría de los algoritmos de planificación son:

- Hay un grupo de procesos ejecutables que compiten por la CPU
- Los procesos son independientes y compiten por los recursos
- El trabajo del planificador es el de distribuir el preciado recurso de la CPU a los diferentes procesos de manera "justa" y de manera de optimizar algún criterio de rendimiento.

Características

- Si bien los procesos o hilos no son todos iguales, cumplen en su mayoría con el mismo patrón: es típico que ejecuten durante un cierto tiempo, luego realicen alguna operación de entrada/salida, luego ejecuten por un tiempo más y así hasta finalizar.
- Los procesos se caracterizan por tener ráfagas de CPU intercaladas con rafagas de E/S *(Bursts)*
- Los procesos que realizan muchos cálculos y poca E/S diremos que son orientados a la CPU *(cpu bound)*
- Mientras que los que operan a la inversa diremos que son orientados a E/S o *(I/O bound)*
- Es esperable de un planificador que aproveche los momentos de E/S de un proceso para permutar la CPU a otro ya que las operaciones de E/S suelen tomar mucho tiempo y la CPU estaría ociosa
- Desde el punto de vista de los procesos, estos corren como si tuvieran toda la CPU para ellos. El usuario, que interactúa con el proceso, tiene la misma sensación.



Contexto

Cuando el núcleo decide ejecutar un proceso o hilo, ordena al hardware almacenar el **contexto** del proceso en ejecución. *(Estado de registros de la CPU, en el área de almacenamiento del contexto del proceso)*

Luego se carga el contexto del otro proceso desde su propia área de almacenamiento, se limpian las caches, se cambia el mapa de memoria virtual y se reanuda la ejecución de ese proceso o hilo recién cargado. A este procedimiento se le denomina **cambio de contexto.** *(conlleva un overhead extra)*

Desalojo

- El OS mueve un proceso de estado ejecutando a Listo sin que el proceso o hilo lo haya solicitado. Para luego retomar su ejecución.
- Sin desalojo, el SO implementa la política de "ejecutar hasta finalizar"
- Un proceso puede ceder voluntariamente la CPU porque ha finalizado o porque está esperando un evento externo de E/S *(disco, teclado etc)*. En este caso, para implementar el desalojo el SO necesita una interrupción de reloj o equivalente.
- Los SO poseen un "reloj de interrupción" o "temporizador de intervalos" para generar estas interrupciones.
- El desalojo está presente en todos los sistemas operativos modernos de propósito general
- Un planificador sin desalojo requiere de la buena voluntad de los procesos *(multitarea cooperativa)*

Colas

- El SO organiza colas para administrar las solicitudes a los recursos compartidos
- Hay colas para la planificación del uso de CPU y también colas para el uso de dispositivos. Usadas para ordenar el acceso.
- Los procesos son colocados en la **cola de procesos** *(task queue)* al ser creados.
- Esta cola está formada por los procesos que están listos para ejecución pero que aún no residen en memoria.
- La residencia en memoria es imprescindible para la ejecución del proceso
- La cola de listos *(ready queue)* contiene aquellos procesos que están listos para ejecución en memoria principal. Compiten directamente por la CPU.
- Cualquiera de estas colas no contiene el proceso en sí, sino su PCB.

Criterios

- Maximizar el uso de CPU. Cpu siempre ocupada.
- Maximizar el rendimiento. Número de procesos que se ejecutan por unidad de tiempo.
- Minimizar el tiempo de retorno. Tiempo transcurrido desde que el proceso ingresa hasta que termina. Sumando todos los overheads.
- Minimizar el tiempo de espera. Tiempo que el proceso está en cola de listos sin ejecutar.
- Minimizar el tiempo de respuesta. Tiempo que tarda en comenzar a responder.

Otros criterios:

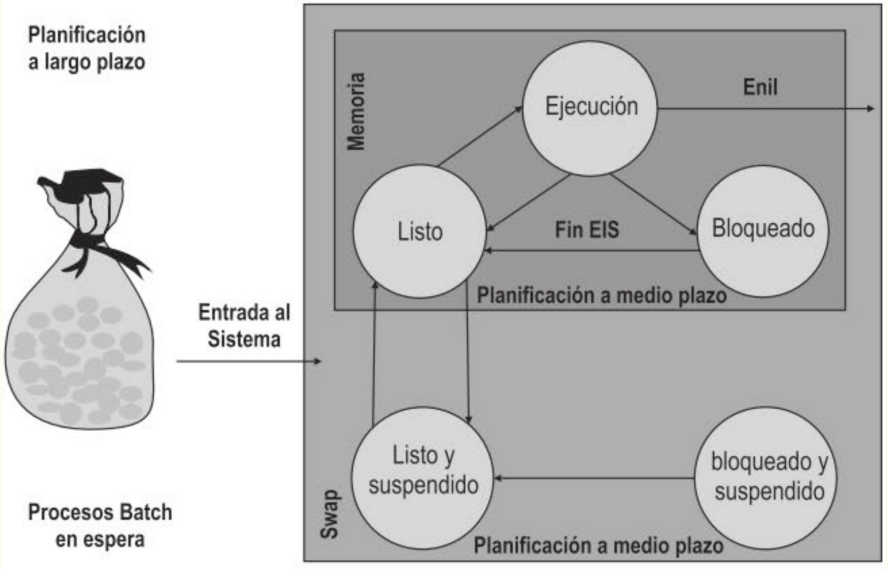
- Equidad. Todos los procesos tratados de igual forma.
- Recursos equilibrados. Evitar abusos en el uso de recursos.

Muchos de estos criterios tienen conflictos entre sí.

Tipos de planificación

El planificador está muy relacionado con los distintos estados de un proceso

- El **planificador a largo plazo** determina qué proceso o hilos se admiten en la cola de listos y cuales pueden competir por los recursos. Al efectuar la llamada al sistema para ejecutar un determinado programa, este planificador autoriza o no la admisión al conjunto de procesos que se están ejecutando en ese momento.
- El **planificador de mediano plazo** es el que se ocupa de la suspensión de procesos y decide qué procesos en estado "bloqueado" dejan de estar en memoria principal para pasar a estar en memoria secundaria.
- El **planificador de corto plazo** es el que decide, de entre los procesos en estado "listo", cual es el que ejecutará a continuación y durante cuánto tiempo. También llamado planificador de la CPU, es el que interviene más frecuentemente.



Tipos de planificación

Cuando se toman las decisiones de planificación?, cuando el planificador elige a qué proceso ejecutar?:

- Cuando un proceso cambia de estado "ejecutando" a "esperando", por un pedido de E/S, porque espera que termine un hijo o porque espera que se complete una operación de sincronización.
- Cuando un proceso cambia de estado "ejecutando" a "listo", por ejemplo al completar el manejo de una interrupción como puede ser temporizador del SO
- Cuando un proceso cambia de estado de "esperando" a "listo", por ej, al completar una E/S
- Cuando el proceso finaliza.

Algoritmos de planificación

- FCFS (first-Come-First-Serve)
- SJF (Shortest-Job-First)
  - SRPT (Shortest-Remaining-Processing-Time)
- Por prioridad
- RR (Round-Robin)
- Colas multiniveles
  - MLQ (multi level queues)
  - MLFQ (multi level feedback queues)

Algoritmos: FCFS

First-Come-First-Serve. Primero en llegar, primero en ser atendido.

- Simple de implementar, se le asigna CPU al primero que lo solicita a medida que los proceso o hilos van arribando.
- Se organiza mediante una cola FIFO, colocando el PCB del proceso entrante al final de la cola
- Se elige el primero de esa cola para asignar CPU, y se lo ejecuta hasta que finalice. Algoritmo sin expulsión
- Poco eficiente, largos tiempos de espera.
- Procesos cortos pueden quedar atascados atrás de procesos más largos.

Algoritmos: SJF

- **Shortest-Job-first.** Algoritmo no expulsivo
- Elige de entre los proceso listo, aquel que tenga el tiempo de procesamiento más corto. En caso de empate, usa FCFS
- Es necesario conocer el tiempo de procesamiento de los procesos, algo que antiguamente se podía estimar bien, pero no con los procesos interactivos de la actualidad.
- Hay una versión mejorada para uso en planificación a corto plazo en sistemas interactivos basado en el cálculo del tiempo promedio de rafaga de CPU previos

Algoritmos: SRPT

- **Shortest-Remaining-Processing-Time**
- SJF mejorado, transformado en expropiativo
- Si se está ejecutando un proceso y llega a la cola otro cuya siguiente ráfaga de CPU es más corta que lo que falta por ejecutar del proceso actual.
- En ese caso se desaloja el proceso actual para dar lugar al que acaba de llegar a la cola de listos que cumple con esa premisa.
- Conocida por ser óptima en minimizar el tiempo medio de respuesta
- Poco usado
- Penaliza injustamente a los proceso largos

Algoritmos: Por prioridad

- Asocia un valor a cada proceso que representará su prioridad y se le asigna la CPU al proceso de la cola de listos que tenga el mayor valor.
- Según el SO el valor puede estar en un rango fijo de 0 a n
- Cada sistema también determina cual es el número con mayor o menor prioridad
- La prioridad puede ser fija o variable. Si es variable, puede cambiar durante la vida del proceso.
- Prioridad Interna: el sistema decide de acuerdo a factores medibles y en función del uso de los recursos como memoria, archivos abiertos y tiempos de E/S
- Prioridad Externa: interviene el operador. O factor económico, donde se paga por el procesamiento.

Algoritmos: Round Robin

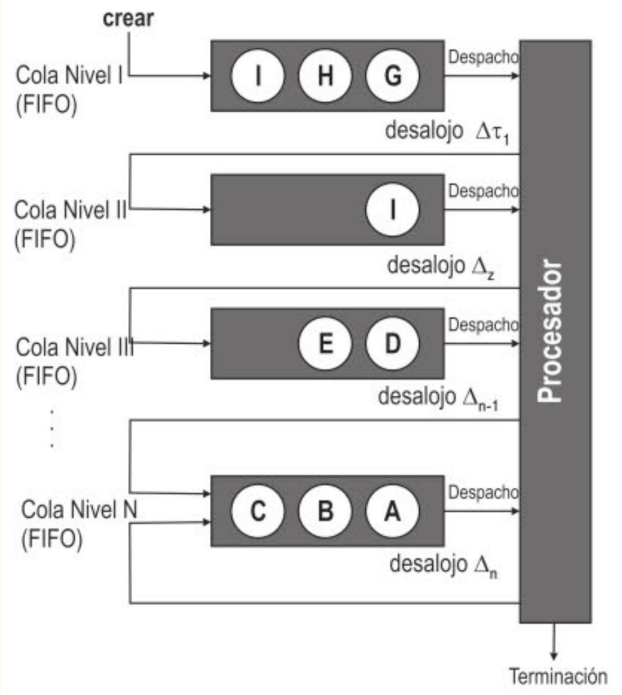
- Muy usado en sistemas de tiempo compartido
- Se asigna a cada proceso de la cola de listos un intervalo de tiempo de CPU llamado *time-slice* o ***quantum***
- Los procesos van tomando la CPU por turnos y por el tiempo que indique el quantum
- Al cumplirse el tiempo, el proceso es desalojado mediante una interrupción y retornado a la cola de listos.
- También puede suceder que el proceso haya terminado o que entre en una operación de entrada/salida. En ese caso el planificador selecciona otro de la cola de listos.
- Equitativo y ordenado
- Tiempo de espera promedio largo

Algoritmos: MLQ

- Multi level queue. Colas multiniveles
- Se divide la cola de procesos listos en varias colas, una por cada tipo de trabajo
- El criterio de asignación es en función de alguna propiedad del proceso, por ejemplo el tamaño de la memoria, la prioridad o el tipo de proceso.
- Colas distintas para procesos interactivos o en background porque se sabe que requieren distintos tiempos de respuesta.
- Los interactivos pueden ser planificados usando RR y mientras que para los otros usar FCFS.
- Debe haber un criterio de planificación entre las colas. Suele implementarse prioridad fija con desalojo. Esto es, no se puede ejecutar un proceso si hay otro listo en una cola con mayor prioridad.

Algoritmo: MLFQ

- **Multi-level-Feedback-queues**
- Permite que un proceso se pueda mover de una cola a otra dependiendo de su comportamiento durante su ejecución.
- Un planificador MLFQ estará definido por los siguientes parámetros:
  - Número de colas
  - Algoritmo de planificación de cada cola
  - Algoritmo de planificación entre distintas colas
  - Método para determinar cuándo pasar un proceso a otra cola de mayor o menor prioridad
  - Método para determinar en qué cola introducir el proceso para darle servicio
- Este algoritmo es considerado el más general, pero a la vez el más complejo.



Planificación en Posix

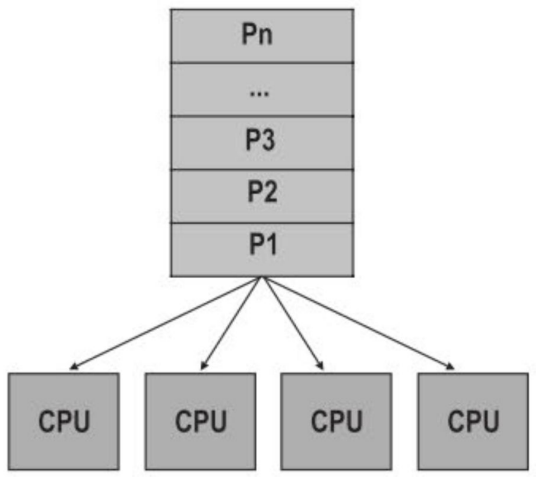
- Norma publicada en 1996 por la IEEE establece una serie de políticas de planificación aplicables a proceso e hilos
- Cada proceso e hilo lleva asociada una política de planificación y prioridad
- Cada política lleva asociado un rango de prioridades al menos de 32 niveles
- El planificador elegirá siempre el proceso o hilo con la prioridad más alta
- Posix establece las siguientes políticas:
  - FIFO *(FCFS pero con desalojo)*
  - RR
  - OTHER *(dependerá de la implementación)*
- Un proceso puede cambiar su prioridad o su política usando el servicio adecuado, esto implica una re planificación
- Las políticas conviven en el planificador ya que se definen por proceso

Planificación en Linux

- Constante evolución, y cambiante de acuerdo a la versión del núcleo
- Linux almacena toda la información referida al proceso, incluyendo su planificación en su PCB. Estructura `task_struct`
- Características según la versión:
  - 0.96: planificador bastante simple, buena respuesta a mucha E/S. Basado en prioridades dinámicas, con rodajas de tiempo y desalojo pero sin colas, el proceso se elegía a partir de un grupo de procesos listos.
  - 1.2: Cada proceso almacena la política de planificación en su PCB. Si bien Linux no es un SO en tiempo real, a partir de esta versión se podía planificar un proceso como de tiempo real.
  - 2.2: Clases de planificación y permite políticas de planificación para tareas de tiempo real y tareas no desalojables. También incluyó soporte para multiprocesamiento simétrico.

Planificación en Linux: version 2.4

- El planificador dividía el tiempo en períodos *(epochs)*. Dentro de ese período, un proceso podía ejecutar hasta agotar su rodaja de tiempo.
- Si el número de procesos era grande, el planificador podía demandar una notable cantidad de tiempo de procesador.
- Carecía de prestaciones para aprovechar el multiprocesamiento simétrico



Planificación en Linux: version 2.6.8.1

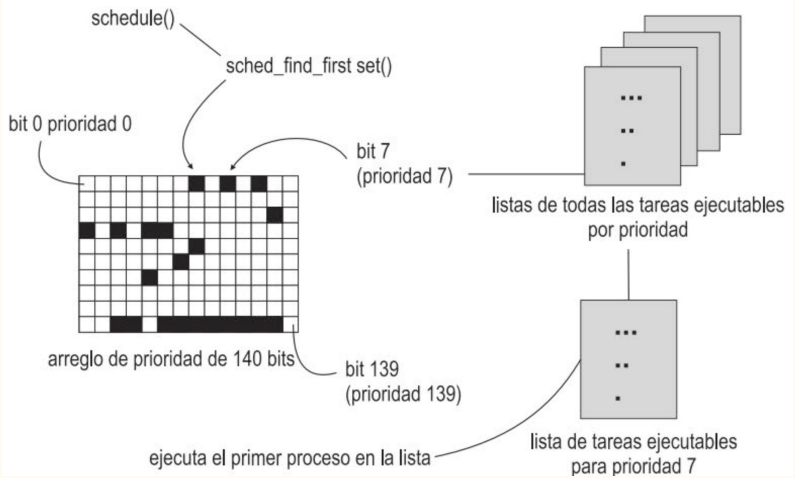
- Uno de los cambios más significativos a partir de la serie 2.5 fue el cambio del planificador
- Uno de los grandes desafío, disminuir el tiempo de cálculo que se incrementaba con la cantidad de proceso
- Uno de los disparadores de este cambio fue el uso de Java, que a través de su máquina virtual, los programas crean múltiples hilos de ejecución
- Base funcional sustentada en 2 estructura:
  - Cola de ejecución
  - Arreglos de prioridad

Características y funcionamiento

- Existe una cola de ejecución por cada CPU *(a diferencia del algoritmo previo)*
- Cada cola de ejecución contiene 2 arreglos de prioridad donde cada uno es un arreglo de listas enlazadas, uno para cada nivel de prioridad. Hay 140 niveles.
- Las tareas que son planificadas para ejecutar, se agregan al final de un arreglo *(al de su nivel de prioridad)* y en ese arreglo los elementos son atendidos por orden FIFO
- El planificador elige para ejecutar a la tarea que esté primera en el arreglo de prioridad activo.
- A medida que los procesos van acabando sus rodajas de tiempo se los va moviendo a otro arreglo de prioridad, el "expirado"
- Las primeras 100 posiciones de la cola de ejecución son reservadas para las tareas de "tiempo real" las otras 40 para las de menor prioridad, llamadas "tareas de usuario"

Características y funcionamiento *(cont.)*

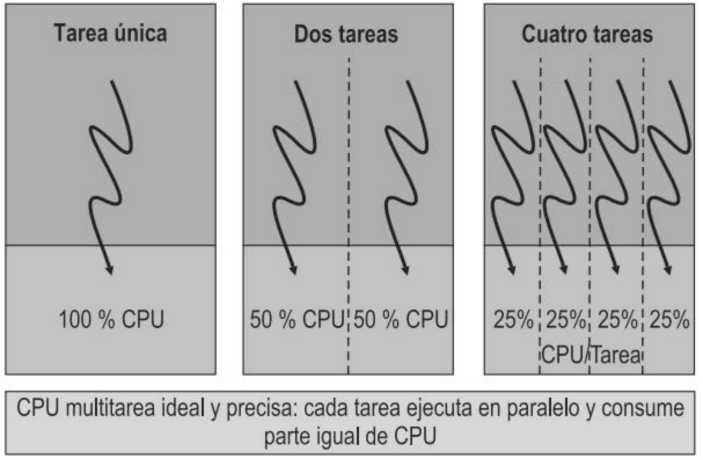
- Durante el movimiento de una cola a otra, a los procesos se les vuelve a calcular una rodaja de tiempo y una nueva prioridad
- Cuando no hay más tareas ejecutables en el arreglo de prioridad activo, se intercambian con el arreglo de prioridad expirado.
- La clave de optimización de tiempos es usar la instrucción de procesador `find-first-bit-set` para encontrar el bit de mayor prioridad habilitado

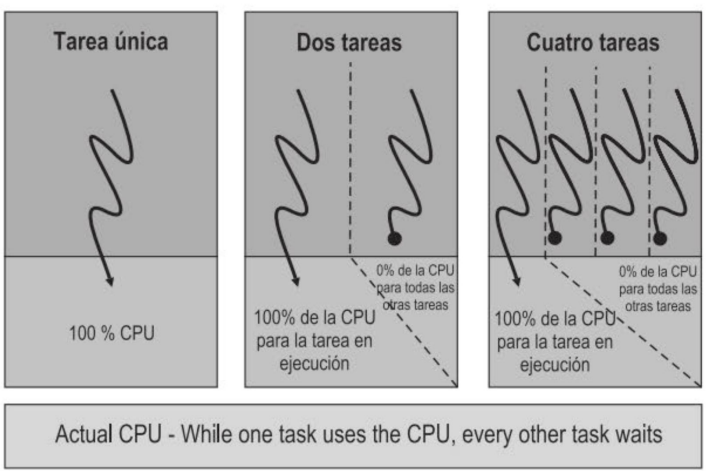


- El tiempo que toma encontrar una tarea para ejecutar no depende del número de tareas activas sino del número de prioridades. El tiempo es siempre el mismo.

Versiones posteriores

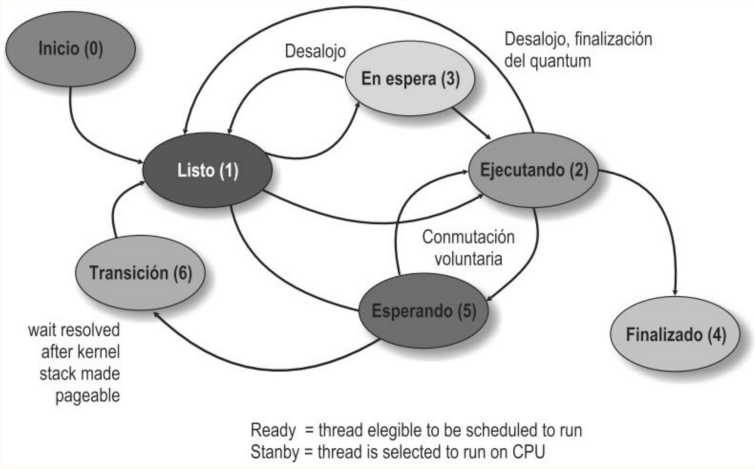
- 2.6.21:
  - Más versatilidad en la configuración de la freq de interrupciones.
  - Mayor interactividad.
  - Más sobrecarga incluso en momentos de inactividad *(muchas interrupciones para atender)*
- 2.6.23
  - CFS *(completely Fair Scheduler)*
  - CFS modela básicamente a una CPU multitarea ideal y precisa sobre hardware real
  - Tiempo de ejecución virtual
  - El tiempo de espera de una tarea es el que se considera que la tarea hubiera merecido en el modelo de equidad.
  - CFS no usa rodajas de tiempo en el sentido convencional, sino que calcula rodajas de tiempo para cada proceso justo antes de ser planificado.





Planificación en Windows

- Sistema con desalojo basado en prioridades
- El hilo en estado de listo con la mayor prioridad es el elegido para ejecutar
- El hilo ejecuta un cierto *Quantum* de tiempo
- Los valores de Quantum pueden variar:
  - Configuración del sistema *(largos o cortos)*
  - Proceso en primer o segundo plano
  - uso de objetos de trabajo para alterar el quantum



- Por mayor prioridad, Windows puede desalojar un proceso antes de que termine su quantum.
- El código del planificador en Windows está diseminado por todo el núcleo, no tiene un módulo específico que lo implemente. A todo este conjunto de rutinas se le denomina el *activador del núcleo*

Planificación en Windows *(cont.)*

Las rutinas de planificación entran en acción cuando ocurre un evento de planificación. Estos eventos pueden ser:

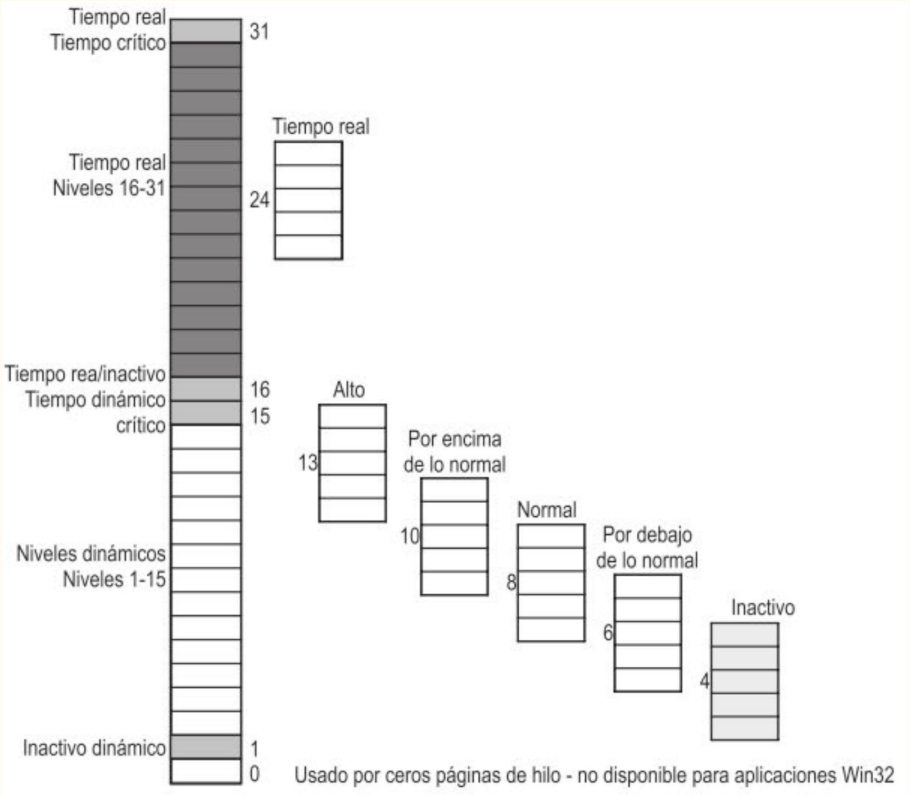
- Un hilo que queda listo para ejecutar. Recién creado o liberado de espera
- Un hilo termina su quantum y deja de ejecutar. Finaliza o entra en espera
- Cambia la prioridad del hilo. Por una llamada al sistema o el mismo OS.
- En sistemas de múltiples procesadores, cambia la afinidad y el proceso se mueve.

Cuando algo de esto sucede, Windows realiza un cambio de contexto y debe determinar qué hilo va a ejecutar a continuación.

Windows planifica a nivel de hilos, los procesos no ejecuta sino que solo proveen recursos y un contexto para que sus hijos ejecuten.

Niveles de prioridad

- 32 niveles de prioridad (16 para tiempo real (16 al 32), 15 niveles variables (1 al 15) y el nivel 0 de sistema, reservado para el hilo de página cero.
- Tanto la API de Windows como el Núcleo pueden asignar prioridades a hilos.
- La prioridad base de un hilo se hereda del proceso que la creó y luego puede ser modificada.



Base de datos del activador

El núcleo usa un conjunto de estructuras de datos para llevar registro de todos los procesos, hilos y sus propiedades. A este conjunto se lo denomina *dispatcher database o base de datos del activador*.

Esta base de datos lleva registro de qué hilos están esperando para ejecutar y qué procesadores están ejecutando hilos.

- El procesador tiene asociadas 32 colas, una por nivel de prioridad, llamadas *colas de listos*.
- Windows tiene un resumen de listos que indica que si hay al menos un hilo en la cola de listos para un determinado nivel
- Adicionalmente también existe un resumen de inactivos.