Planificación

Desde hace unos años, se están optando por ordenadores multiprocesador y multithread, esto significa que los hilos y procesos compiten por el CPU.

Cuando un proceso no está en ejecución y quiere que la CPU le de un **quantum**, se pone en estado **listo**. Imaginemos que sólo hay una única CPU. Ésta tiene que decidir lo siguiente que se va a ejecutar. Esta parte del sistema se llama **planificador de procesos** y usa un **algoritmo de planificación**.

Las siguientes cuestiones van a referirse a tanto planificación de hilos o procesos (ya que los hilos se planifican sin importar el proceso al que corresponden) aunque se harán más adelante puntualizaciones.

Introducción a la planificación

En las computadoras actuales normalmente hay un sólo proceso activo (una persona suele trabajar con un programa o dos al mismo tiempo), de tal forma que los demás procesos están en segundo plano y apenas ocupan tiempo de CPU.

Otra cosa que hay que tener en cuenta es que la mayor parte del uso de la CPU es en Ráfagas Largas donde se ejecuta un proceos propiamente y otras Ráfagas Cortas entre ellas donde se acceder a Entrada/Salida.

Para hacer comnutación de procesos se siguien los siguientes pasos:

1. Se cambia de modo **usuario** a **kernel**.
2. Se **guarda el estado** del proceso actual (y todos los registros que usa para retomarlos más adelante).
3. Se elige **otro proceso** mediante un algoritmo de planificación.
4. Se carga en MMU el **mapa de memoria** del nuevo proceos.
5. Se **inicia** el nuevo proceso.

   *NOTA: Como se puede comprobar, no es un proceso barato computacionalmente, por lo que hay que hacerlo óptimamente.

Tipos y categorias para planificar procesos

Los algoritmos se dividen en:

- **No apropiativos**: Selecciona un proceso y deja que se ejecuta hasta que se bloquea el mismo.
- **Apropiativos**: Selecciona un proceso y deja que se ejecute durante un tiempo, después el SO lo bloquea y selecciona otro.

Por otro lado, existen distintos tipos de sistema, en los que algunos funciona mejor un apropiativo y otros un no apropiativo:

1. **Procesamiento por Lotes**: Se utilizan para hacer tareas periódicas como realizar nóminas, inventarios, etc. En estos sistemas no hay usuarios que esperen una respuesta rápida (por lo que pueden ser útiles los **no apropiativos**).

```
Importa el RENDIMIENTO, es decir el número de trabajos por hora.
```

2. **Interactivo**: Al necesitar respuestas rápidas para interactuar con el usuario, se deben evitar procesos que acaparen la CPU y puedan mantener al usuario esperando con un proceso que no se bloquea el mismo mientras hace una tarea no prioritaria (obviamente necesitamos siempre algoritmos **asociativos**).

```
Importa minimizar el TIEMPO DE RESPUESTA, es decir, el tiempo que transcurre entre
emitir un comando y obtener el resultado.
```

3. **De tiempo real**: El ejemplo más claro sería un robot; mientras que en el interactivo, interactua la computadora con el usuario sin saber exactamente cuando va a tener que responderle o escucharle, en un robot está pensado para que cada tarea se ejecute de forma más controlada. De tal forma que los procesos suelen hacer su trabajo y bloquearse con rapidez. (por lo que **no sería indispensale un algoritmo apropiativo**).

```
Se caracteriza por tener TIEMPOS LÍMITE que deben cumplirse
```

Algoritmos POR LOTES

FIFO

No Apropiativo

Es un algoritmo fácil de entender y de implementar. Sin embargo, tiene la contra de que si da la casualidad de que el proceso que se ejecuta en un momento tarda varios segundos, mientras que los siguientes son procesos que requieren de microsegundos (como E/S). Estos procesos que pueden acabarse en menos de lo que tarda en acabar el que está actualmente van a tener que esperar mucho tiempo.

Más Corto Primero

No Apropiativo

Este algoritmo resuelve el problema que plantea el anterior: si hay varios procesos rápidos y uno más largo, el total de tiempo perdido por cada uno va a ser menor.

Es decir, hay 3 procesos de 1, 2 y 10 segundos respectivamente. Si se ejecuta en este orden (10, 2, 1) los dos últimos procesos van a tardar en ejecutarse 10+2 y 10+2+1, en total 25 segundos (más los 10 del primero 35 segundos en total).

Sin embargo si se ejecutan el el orden (1, 2, 10) los dos últimos procesos van a tardar en ejecutarse 2+1 y 10+2+1, en total 16 segundos (más los 1 segundos del primero 17 en total).

Menor Tiempo Restante a Continuación

Apropiativo

Es la versión apropiativa del algoritmo **Más corto primero**. La diferencia es que el planificador puede comparar los tiempos que necesita cada proceso.

En un ejemplo, si tenemos un proceso que tarda 10 segundos y llevamos transcurridos 3 (es decir, quedan 7 segundos) y se pone en listo un proceso que tarda 5 segundos, el planificador tiene libertad para bloquear el proceso actual y darle prioridad al de 5 segundos (ya que 5 < 7).

Algoritmos para INTERACTIVOS

Planificación por Turno Circular

A cada proceso se le asigna un intervalo de tiempo límite para ejecutarse llamado **quantum**. Si el proceso sigue ejecutandose la CPU es apropiada para darsela a otro proceso y el actual se bloquea.

Con respecto al quantum, hay que recordar que el costo para conmutar de un proceso a otro es bastante costoso, asi que si se establece un quantum demasiado corto se producen demasiadas conmutaciones de procesos y se reduce la eficiencia de la CPU. Sin embargo, si se establece demasiado largo puede producir una mala respuesta a las peticiones interactivas.

Planificación por Prioridad

A cada proceso se le asigna una prioridad, el proceso con prioridad más alta es el que se puede ejecutar.

Para evitar que un proceso con alta prioridad se ejecute de manera indefinida (ya que siempre es el que tiene más prioridad, por ejemplo una película), el planificador puede **reducir la prioridad** en cada **interrupción de reloj**. Además, también se pueden utilizar otros métodos como asignarle un quantum.

Múltiples Colas

Este algoritmo cuenta con varias listas FIFO, cada una con una prioridad. En este caso, los procesos con más prioridad se ejecutarían cada 1 quantum, los de la siguiente cola, cada 2 quantums y así sucesivamente. Este algoritmo simplemente explica las diferentes colas, los ejemplos de los quantums estuvieron trabajando de esa forma durante un tiempo en el CTSS, pero no tiene por que ser exactamente así, con quantums. De hecho en ese ejemplo cada vez que un proceso utilizaba todos los quantums que tenía asignados, se movía una clase hacia abajo en la jerarquía de colas, lo cual realmente en el algoritmo no está especificado.

Proceso Más Corto a Continuación

En las computadoras interactivas, se prioriza el tiempo que tarda en volver a atender al usuario, por lo que el proceso más corto a continuación tiene mucho sentido. La complicación que tiene este método es averiguar cuál de los procesos actuales ejecutables es el más corto.

Planificación Garantizada

Este algoritmo se basa en hacer promesas reales a los usuarios acerca del rendimiento y después cumplirlas.

Si hay *n* usuarios conectados, un único usuario recibirá 1/*n* de los ciclos de la CPU.

Planificación por Sorteo

El resultado es parecido al anterior algoritmo. La idea es dar a los usuarios *boletos de lotería* y realizar un sorteo entre todos los boletos. En este caso, los procesos prioritarios tendrían más boletos, pero puede ser que aún teniendo más boletos no salgan premiados.

Planificación por Partes Equitativas

Este algoritmo tiene en cuenta los usuarios que ejecutan los procesos a diferencia de los demás, que simplemente tienen en cuenta cosas como la prioridad, el tiempo estimado o el momento en el que se coloca en la cola.

Cuando hay múltiples usuarios usando el sistema puede ser buena idea.

Algoritmos para SISTEMAS TIEMPO REAL

En el caso de los Sistemas de Tiempo Real, es conveniente tener siempre la respuesta correcta pero en un tiempo límite fijo. Por ejemplo, un robot debe dar respuestas a estímulos que suelen tener un tiempo determinado, y estas respuestas queremos que sean lo más precisas posibles, como también ocurriría en el caso de una máquina que monitoriza pacientes en un hospital (a diferencia de los Interactivos que prácticamente lo único que importa es una respuesta rápida para que el usuario no espere nada).

Precisamente estos sistemas se categorizan en dos:

1. **Tiempo Real Duro**: los tiempos tienen un límite absoluto que se debe cumplir obligatoriamente.
2. **Tiempo Real Suave**: en este caso no es conveniente fallar en el tiempo límite pero es tolerable.

En estos Sistemas, los procesos tienen un tiempo conocido a priori y son tiempos cortos, precisamente diseñados para responder a eventos sin fallar en el tiempo fijo. Teniendo en cuenta esto último, tal vez ni siquiera sea posible manejar todos los eventos (que pueden ser **periódicos** o **aperiódicos**).

Cuando un Sistema de Tiempo Real es capaz de manejar todos los eventos sin fallar en el tiempo de CPU para cada proceso, se dice que es **Planificable**.

Política vs Mecanismo

El algoritmo de planificación de un sistema está implementado de cierta forma con algoritmo de **planificación por prioridad**. Sin embargo, cada proceso puede modificar las prioridades de sus hijos mediante una Llamada al Sistema, de tal forma que aunque el algoritmo global no cambie, cada proceso puede controlar un poco más a sus hijos y darle más libertad dentro de su alcance.

Planificación de Hilos

Hilos a Nivel Usuario

El kernel no está consciente de la existencia de los hilos:

- Selecciona el Proceso A y le otorga un quantum, este proceso tiene su propio **planificador de hilos** y puede ejecutarse todo el tiempo que desee su propio planificador. Cuando acaba el quantum, el planificador de procesos superior seleccionará otro proceso. Y si vuelve otra vez a darle un quantum al Proceso A continuará con el hilo anterior.

Hilos a Nivel Kernel

El kernel selecciona un hilo específico sin importar el proceso al que pertenece (aunque puede saber a qué proceso pertenece si lo desea).

Esto permite que cuando acabe un hilo, puede avisar de que ha acabado al Planificador y puede ejecutar cualquier otro hilo del sistema. A diferencia de los hilos a nivel usuario, que si acaba un hilo, el quantum sigue dandole tiempo a otro hilo de su mismo proceso aunque no sea lo más óptimo.

Sin embargo, no siempre es óptimo cambiar a otro hilo de otro proceso si no tiene mucha prioridad ya que recordemos que es costoso cambiar de proceso.

En esta imagen se ven muy bien las posibilidades que da cada uno:

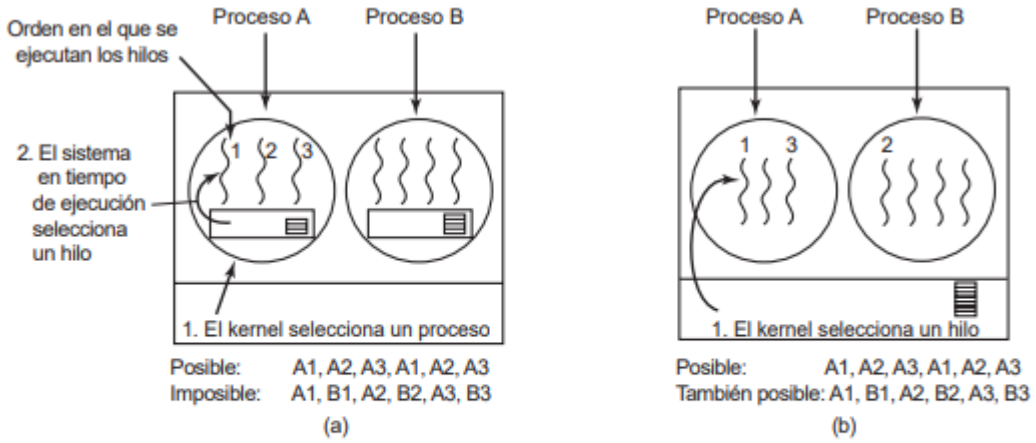

Figura 2-43. (a) Posible planificación de hilos a nivel usuario con un quántum de 50 mseg para cada proceso e hilos que se ejecutan durante 5 mseg por cada ráfaga de la CPU. (b) Posible planificación de hilos a nivel kernel con las mismas características que (a).