PLANIFICACION

Cuando una computadora se multiprograma, con frecuencia tiene varios procesos o hilos que compiten por la CPU al mismo tiempo. Esta situación ocurre cada vez que dos o más de estos procesos se encuentran al mismo tiempo en el estado listo. Si sólo hay una CPU disponible, hay que decidir cuál proceso se va a ejecutar a continuación. La parte del sistema operativo que realiza esa decisión se conoce como **planificador de procesos** y el algoritmo que utiliza se conoce como **algoritmo de planificación**.

¿Cuándo correr la planificación?

- Cuando se crea un proceso (cuando esta en listo y hay que pasarlo a ejecución)
- Cuando se termina un proceso
- Luego que un proceso se bloquea (E/S)
- Cuando llega una interrupción de un periférico (timer)

Necesidad

Poder realizar multitarea debido a que aprovecho los tiempos muertos de la CPU y no la dejo ociosa. La función que cumple el planificador es la siguiente:

Cuando un proceso se está ejecutando y se produce una interrupción externa **no se ejecuta el planificador,** si no que lo manda a bloqueado, luego de que se atiende mi interrupción de E/S, ahí recién **se ejecuta mi planificador** que realiza el cambio de contexto de tareas, guarda el contenido actual de mi proceso en el TSS y recién ahí pasa de bloqueado a listo el proceso que estaba esperando una interrupción de E/S, que luego si le toca en la lista pasa a ejecución por el planificador.

Tipos de algoritmos de planificación en función de interrupciones de clock

CPU Bonded: que mi CPU se queda haciendo muchos cálculos.

No expropiativo: selecciona un proceso para ejecutarlo y después sólo deja que se ejecute hasta que el mismo se bloquea (ya sea en espera de una operación de E/S o de algún otro proceso) o hasta que libera la CPU en forma voluntaria. Incluso aunque se ejecute durante horas, no se suspenderá de manera forzosa. En efecto, no se toman decisiones de planificación durante las interrupciones de reloj. Una vez que se haya completado el procesamiento de la interrupción de reloj, se reanuda la ejecución del proceso que estaba antes de la interrupción, a menos que un proceso de mayor prioridad esté esperando por un tiempo libre que se acabe de cumplir.

Expropiativo: selecciona un proceso y deja que se ejecute por un máximo de tiempo fijo. Si sigue en ejecución al final del intervalo de tiempo, se suspende y el planificador selecciona otro proceso para ejecutarlo (si hay uno disponible). Para llevar a cabo la planificación expropiativa, es necesario que ocurra una interrupción de reloj al final del intervalo de tiempo para que la CPU regrese el control al planificador.

Sistemas por lotes (Batch)

Se utilizan para hacer tareas periódicas en las que no interactúan con usuarios por lo tanto se usan algoritmos no expropiativo.

Algoritmos de planificación

- ***Primero en entrar primero en ser atendido (First-come First-Served):*** A medida que van llegando las tareas la voy poniendo en memoria y las voy ejecutando recién cuando la que se está ejecutando se bloquea por E/S. Si un proceso se bloquea por E/S y le llega la E/S, se pasa al final de la cola de espera. La desventaja es que no es equitativo con los procesos I/O bonded y CPU bonded.

- ***El trabajo mas corto primero (Shortest Job First):*** Se usa cuando uno sabia cuanto demoraba un programa. Selecciona el trabajo más corto primero. La desventaja es que no puedo saber cuánto demoran mis tareas de antemano a menos que sea periódica, debo tener mis tareas en el tiempo 0 para poder acomodarlas y las tareas más largas tardan más tiempo en ejecutarse debido a que si entran tareas más cortas seguirán ejecutándose estas. La ventaja es que disminuyen el turnaround.

- ***El menor tiempo restante a continuación (Shortest Remaining Time Next):*** Es un sistema expropiativo, necesito saber de antemano los tiempos de ejecución. El planificador siempre selecciona el proceso cuyo tiempo restante de ejecución sea el más corto. Cuando llega un nuevo trabajo, su tiempo total se compara con el tiempo restante del proceso actual. Si el nuevo trabajo necesita menos tiempo para terminar que el proceso actual, éste se suspende y el nuevo trabajo se inicia (por eso expropiativo). La ventaja es que no necesito tener todas mis tareas en el tiempo cero debido a que mi planificador es expropiativo dado que si llega tarde un proceso y tiene el tiempo restante de ejecución más corto este se va a ejecutar. La desventaja es que las tareas más largas tienen inanición, esto quiere decir que, si llegan muchos procesos de corto tiempo, las tareas largas no se ejecutan nunca.

Sistemas Interactivos

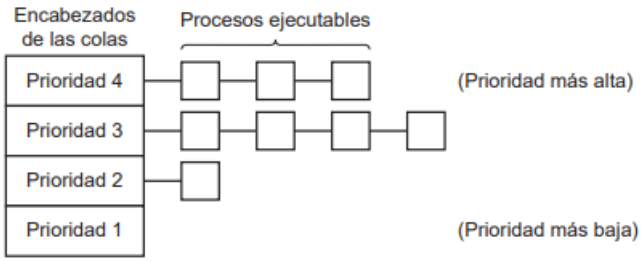
En los sistemas interactivos lo que necesito es un feedback rápido para el usuario, entonces lo más importante es minimizar el tiempo de respuesta es decir el tiempo que transcurre un comando y obtener el resultado (proceso E/S).

Algoritmos de planificación

- ***Planificación por turno circular (Round Robin):*** Sistema expropiativo. A cada proceso se le asigna un intervalo de tiempo, conocido como quantum, durante el cual se le permite ejecutarse, este quantum no tiene que ser muy chico (se desperdicia el tiempo que se da la CPU) ni muy grande (se tarda en dar la CPU). A medida que los procesos van llegando estos se encolan y se va seleccionando el primero en la cola para correr. Si terminado quantum sigue corriendo el proceso, se pone al final de la cola y se ejecuta el segundo.

El cambio de contexto es una desventaja, si la conmutación entre procesos tarda 1ms y el quantum es 4ms, se estaría desperdiciando el 20% de la CPU

- ***Planificación por Prioridad:*** Sistema expropiativo. La idea básica es simple: a cada proceso se le asigna una prioridad y el proceso ejecutable con la prioridad más alta es el que se puede ejecutar. Para evitar que los procesos con alta prioridad se ejecuten de manera indefinida (inanición), el planificador puede reducir la prioridad del proceso actual en ejecución en cada pulso del reloj. Si esta acción hace que su prioridad se reduzca a un valor menor que la del proceso con la siguiente prioridad más alta, ocurre una conmutación de procesos. De manera alternativa, a cada proceso se le puede asignar un quantum de tiempo máximo que tiene permitido ejecutarse. Cuando este quantum se utiliza, el siguiente proceso con la prioridad más alta recibe la oportunidad de ejecutarse.



- ***El proceso más corto a continuación (Shortest process next):*** Si consideramos la ejecución de cada comando como un "trabajo" separado, entonces podríamos minimizar el tiempo de respuesta total mediante la ejecución del más corto primero. El único problema es averiguar cuál de los procesos actuales ejecutables es el más corto. Un método se basa en realizar estimaciones del comportamiento anterior a través de un algoritmo de envejecimiento que realiza sumas ponderadas, si la tarea nunca se realizó antes no tengo forma de saberlo. El promedio ponderado que se obtiene le da más peso a lo que se demoró la última vez que a lo que se demoró primero **contrario** a SFJ.

- ***Planificación Garantizada:*** Sistema expropiativo. Se da una proporción de la CPU, 1/n, donde n son los usuarios, y el proceso que se ejecuta primero es el que tiene la menor ratio que se calcula:

$$ratio = \frac{Tiempo\ consumido}{Tiempo\ total\ (en\ funcion\ de\ la\ CPU)}; Tiempo\ consumido = \frac{Tiempo\ de\ inicio}{n}$$

  El problema es que tengo que tener siempre la misma cantidad de usuarios.

- ***Planificación por Lotería:*** Sistema expropiativo. Se selecciona al azar ya que el planificador sortea un boleto y el proceso que lo tenga se ejecuta, los procesos más importantes pueden obtener boletos adicionales para incrementar su probabilidad de obtener la CPU más tiempo.

- ***Planificación por partes equitativas***: Se asigna la CPU en partes equitativas en función de los usuarios, pero no en función de los procesos ya que si un usuario tiene más procesos no importa, se ejecuta un proceso en un usuario y después otro proceso en el otro usuario.

NOTA: cuando hacemos una llamada a sistema hacemos un cambio de contexto y ejecutamos una rutina del SO que es por ejemplo programar el controlador de un disco, pero este no es el algoritmo de planificación después que programe el controlador el SO pone al proceso en estado bloqueado porque sabe que va obtener ese dato en un futuro lejano y recién ahí ***llamo a la rutina para elegir otra nueva tarea que es el planificador.***