Universidad de Carabobo

Facultad Experimental de Ciencias y Tecnología

Departamento de Computación

Sistemas Operativos

Prof. Mirella Herrera

Zaín Rondón – CI: 30.601.248

Asignación 2: *Algoritmos de planificación de procesos*

Un proceso en general, es un conjunto de acciones que se ejecutan secuencialmente para llevar a cabo cierto objetivo.

En el contexto de sistemas, el proceso es considerada la unidad mínima de trabajo de dicho sistema.

En un ***sistema operativo*** se ejecutan cantidad de procesos en concurrencia, siendo el mismo el encargado de manipular y gestionar los recursos para que dichos procesos se ejecuten de manera exitosa. Por esta razón, se desarrollan mecanismos, protocolos y algoritmos que permiten al sistema operativo gestionar los procesos de manera eficiente y segura.

Un programa está compuesto de múltiples instrucciones, y dentro de un sistema operativo se pueden ejecutar una gran cantidad de programas al mismo tiempo de manera concurrente.

Un programa en sí mismo no es un proceso, pero una ***instancia*** de él lo es, esto se conoce como entidad pasiva y activa, donde la instancia de un programa en cuestión es la entidad activa, y el programa en si la entidad pasiva, dentro de un programa existe la posibilidad de crear subprocesos que se ejecuten en un solo hilo (o en varios). La cantidad de instancias (procesos) de programas que corren simultáneamente en un computador (y sus subprocesos) dependen de la capacidad del hardware en términos de memoria, potencia de la CPU y otros recursos como el almacenamiento secundario o externo.

Es tarea del sistema operativo gestionar estos recursos efectivamente, y de esta necesidad se comienzan a desarrollar los llamados ***algoritmos de planificación de procesos***, una serie de algoritmos creados para este propósito. En el presente trabajo exploraremos el concepto y funcionamiento de varios de estos algoritmos, así como cuales serían sus ventajas y desventajas, y requerimientos para su funcionamiento.

Algoritmos de planificación de procesos

Tiempo de espera y tiempo de respuesta

Dado que cada proceso tiene asociado un tiempo de ejecución, es lógico pensar en un ***tiempo de espera,*** que será el tiempo que un proceso pasa en la cola de procesos hasta su ejecución.

Otro concepto importante es el ***tiempo de respuesta***, que puede tender a confundirse con el tiempo de espera. El tiempo de respuesta indica cuánto tarda un proceso desde que llega al sistema hasta que empieza a ejecutarse.

Se calculan de la siguiente manera:

- ***Tiempo de respuesta:*** Momento de inicio de ejecución - Momento de llegada.
- ***Tiempo de espera:*** Tiempo de respuesta - Tiempo de ejecución.

Ambas métricas serán recurrentes en todos los algoritmos que se presentarán a continuación, pudiendo diferir en su comportamiento de algoritmo en algoritmo

Tipos de algoritmos de planificación

Podemos distinguir entre dos tipos de algoritmos de planificación:

- **Apropiativo:** Un algoritmo de planificación es apropiativo cuando permite que el sistema interrumpa la ejecución de un proceso en un determinado momento para ejecutar otro proceso y resumirlo más adelante o cancelarlo.

- **No apropiativo:** Por el contrario, un algoritmo es no apropiativo cuando no permite la interrupción del proceso actual y solo se podrá ejecutar un nuevo proceso cuando este termine.

Estos conceptos también serán referenciados más adelante, donde ciertos algoritmos pueden ser adaptados para funcionar con ambos principios mientras que otros trabajan exclusivamente con alguna de las dos modalidades.

1) **First Come, First Served** (FCFS)

Se trata de uno de los algoritmos de planificación de procesos más simple dentro de su categoría. En este algoritmo los procesos se ejecutan en el orden en el que llegan, de ahí su nombre en inglés; **primero *en llegar, primer atendido***. Un ejemplo real para ilustrar este algoritmo pueden ser las colas en los supermercados, donde el primer cliente de la fila es el primero en ser atendido.

Funcionamiento

1. **Llegada:** El proceso es invocado en el sistema y es colocado en una cola de procesos, donde se colocan en el orden en el que van entrando los procesos.

2. **Ejecución:** Se toma el proceso en el frente de la cola, se ejecuta y es removido de la cola de procesos.

3. **Repetición:** El sistema toma el siguiente proceso en la cola y lo ejecuta.

Este algoritmo no toma en cuenta ningún tipo de prioridad, y entre sus desventajas podemos encontrar tiempos de respuesta muy altos debido a procesos extensos o complejos.

2) **Shortest Job First** (SJF)

Prioriza los procesos con el menor tiempo de ejecución estimado lo que significa que el sistema ejecutará primero los procesos más cortos sobre los más extensos. Es mayormente utilizado en contextos donde la duración de los procesos es conocida o se puede estimar fácilmente, optimizando así el flujo de procesos.

Funcionamiento

1. **Llegada:** Los procesos ingresan a la cola de espera en cualquier orden.

2. **Selección:** Se elige el proceso con el menor tiempo de ejecución entre los disponibles.

3. **Ejecución:** Se ejecuta el proceso seleccionado hasta que termine.

4. **Repetición:** El sistema selecciona nuevamente el proceso más corto disponible y lo ejecuta.

Presenta dos desventajas notables cuando analizamos un poco el flujo de funcionamiento: En primer lugar, el ingreso constante de procesos “cortos” a la cola de procesos puede hacer que eventualmente los procesos largos nunca se ejecuten o tarden mucho en hacerlo, pudiendo renegar procesos vitales o generar tiempos de respuesta muy altos para dichos procesos, este problema puede ser corregido introduciendo el concepto de ***ageing***, o envejecimiento de procesos.

Y el segundo problema es que su aplicación en un sistema en tiempo real como lo puede ser un sistema operativo en sus operaciones fundamentales puede ser limitada debido a que este algoritmo requiere conocer o estimar eficientemente la duración de los procesos de antemano.

3) **Shortest Remaining Time First** (SRTF)

Se trata de una versión ***apropiativa*** del algoritmo ***Shortest Job First*** *(SJF)* donde el orden de los procesos a ejecutar se determina en función del menor tiempo restante para cada proceso, a diferencia de su versión ***no apropiativa*** aquí se puede interrumpir el proceso que se está ejecutando actualmente si llega un proceso con un menor tiempo restante de ejecución.

Funcionamiento:

1. **Llegada:** Los procesos ingresan a la cola de espera en cualquier orden.

2. **Selección:** Se elige el proceso con el menor tiempo de ejecución restante entre los disponibles.

3. **Ejecución:** Si un nuevo proceso con menor tiempo restante llega, el proceso en ejecución se interrumpe y se reprograma.

4. **Repetición:** Se repite el flujo anterior.

El objetivo de este algoritmo es mejorar los tiempos de respuesta al encargarse primero de los procesos más cortos, agregando un costo asociado a constante cambio de contexto, al permitir la ejecución de otros procesos interrumpiendo el proceso actual, lo que puede generar una sobrecarga al sistema con lotes intensivos de procesamiento.

En este algoritmo el sistema necesita llevar un registro de los tiempos de los procesos, así como aquellos que ya han terminado, por lo que a nivel de implementación tiene más requerimientos.

Similar al anterior algoritmo, procesos largos corren el riesgo de ser renegados con la constante introducción de procesos cuyo tiempo (restante) de ejecución sea más corto.

4) **Round Robin, Selfish RR**

Su nombre hace referencia a una metodología de planificación donde los procesos o puntos a tratar se hacen de manera cíclica y continua, en el Round Robin, todos los procesos tienen la misma prioridad.

Es un algoritmo apropiativo, pues constantemente saltará de proceso en proceso atendiéndolos a todos de manera concurrente con la misma prioridad y de manera circular, una vez que llega al último proceso, saltará al primero hasta que todos los procesos eventualmente sean finalizados.

Esta filosofía de equidad se logra a través de la introducción de un concepto llamado "Quantum" o "Time Slice" que es un intervalo de entre 10ms y 100ms, en el cual un proceso tendrá tiempo de ejecutarse antes de pasar al siguiente

Funcionamiento

1. **Llegada:** Los procesos ingresan a la cola en orden de llegada.

2. **Asignación de quantum:** Se establece un intervalo de tiempo fijo (ej. 4 ms).

3. **Ejecución:** Se ejecuta el primer proceso por un máximo de quantum.

    - Si finaliza, se elimina de la cola.
    - Si no finaliza, se pausa y pasa al final de la cola.

4. **Repetición:** Se toma el siguiente proceso en la cola y se ejecuta con las mismas reglas.

La ventaja principal del Round Robin es que se reducen significativamente los tiempos de respuesta para todos los procesos dentro de la cola, siempre y cuando el tamaño del Quantum sea adecuado; si el quantum es muy corto, el sistema tendría muchos cambios de contexto, impactando en el rendimiento, pero si el quantum es muy largo acabaría comportándose más como un FCFS.

Existe una variante del algoritmo RR llamada Selfish RR, que permite que un proceso tome más de un Quantum (o Quantum más largo) según el estado del proceso y su prioridad.

La prioridad generalmente se asigna según parámetros como el tipo de tarea, su estado, uso del CPU. Procesos con un requerimiento de recursos más alto recibirían prioridad bajo este esquema, mientras que procesos menos exigentes seguirían un comportamiento mas cercano al Round Robin tradicional.

5) **Prioridades**

Un algoritmo de planificación de procesos ***con prioridad*** es un algoritmo que tiene la capacidad de atender procesos según una prioridad establecida con anterioridad según varios parámetros particulares.

Algoritmos explorados anteriormente, como el SJF, son algoritmos con prioridades. El algoritmo FCFS es un algoritmo donde todos los procesos tienen la misma prioridad.

En un algoritmo de planificación de procesos con prioridad, cada proceso tiene una prioridad asociada, y determinará en que orden será atendido, si puede ser interrumpido por otro proceso, o si por el contrario es prioritario y debe ser atendido primero.

La prioridad puede ser establecida interna o externamente según la lógica interna del sistema o factores.

Las prioridades externas en la planificación de procesos se establecen con base en factores ajenos al sistema operativo, lo que significa que no dependen de características técnicas como el uso de CPU o memoria, sino de decisiones administrativas o políticas.

Las prioridades internas en la planificación de procesos se basan en métricas medibles dentro del sistema operativo para determinar la prioridad de ejecución, como el requerimiento de memoria, CPU, límites de tiempo y/o llamadas al sistema.

Un problema recurrente en los algoritmos con prioridad es la ***iniciación*** de un proceso, donde el ingreso constante de nuevos procesos con una prioridad mas alta puede renegar a procesos con una prioridad menor, provocando que su tiempo de respuesta sea muy elevado, o bien nunca se ejecuten.

Para contrarrestar esto, se introduce el concepto de “aging” o “envejecimiento” de procesos, que va ajustando su prioridad según cuanto tiempo lleva esperando su ejecución, eventualmente un proceso “antiguo” tendrá la prioridad más alta y será ejecutado.

6) **Colas Multinivel y colas Multinivel con Retroalimentación**

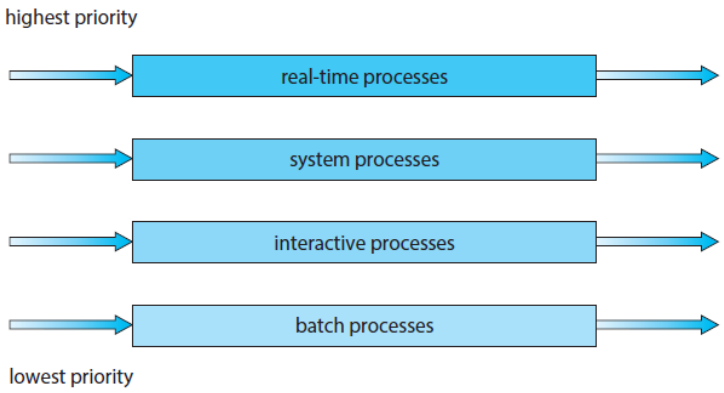
La planificación con colas multinivel es una técnica utilizada en sistemas operativos para gestionar procesos según diferentes niveles de prioridad o características, separándolos en múltiples colas con reglas de planificación específicas.

1. **Clasificación de procesos:** Se asignan a distintas colas según su prioridad o tipo. Cada cola tiene su propio algoritmo: Puede usar FCFS, RR, SJF, etc.

2. **Orden de ejecución:**
   - Las colas de alta prioridad se ejecutan primero.
   - Procesos de colas inferiores solo se atienden si la superior está vacía.

3. **Reglas de transferencia:**
   Algunos sistemas permiten que los procesos cambien de cola con base en su tiempo de ejecución o consumo de recursos (retroalimentación)

Se trata de un sistema de planificación mas complejo que requiere un planteamiento bien pensado sobre la prioridad intrínseca de cada proceso, así como la asignación de dicha prioridad para cada proceso. Es un sistema flexible que permite definir la lógica requerida en cada nivel de acuerdo a las necesidades y requerimientos de cada uno.



Un diagrama conceptual de una Cola de prioridad multinivel, la prioridad más alta la tendrían los procesos interactivos, o procesos en tiempo real.

En la forma más generalizada de este enfoque, se asigna una prioridad estáticamente a cada proceso y este permanece en la misma cola mientras dura su tiempo de ejecución.

Cola multinivel con Retroalimentación

Generalmente cuando se utiliza un algoritmo de planificación de cola multinivel, el proceso esta destinado a permanecer siempre en la cola a la que fue asignado durante su ciclo de vida.

En contraste, una cola multinivel con retroalimentación permite que el sistema operativo mueva procesos de una cola inferior a una superior y viceversa.

El funcionamiento en alto nivel puede verse de la siguiente manera:

1. **Clasificación inicial:** Los procesos comienzan en una cola específica, generalmente en la de menor prioridad.

2. **Evaluación durante la ejecución:** A medida que un proceso usa CPU o realiza operaciones de entrada/salida, su prioridad puede ajustarse.

3. **Cambio de cola:**
    - Si un proceso consume mucho CPU, puede bajar de nivel, relegándose a una cola con menor prioridad.
    - Si un proceso espera demasiado tiempo, puede subir de nivel para evitar inanición.

4. **Ejecución progresiva:** Los procesos avanzan por distintas colas hasta que terminan o alcanzan una prioridad establecida.

Tabla comparativa de los algoritmos de planificación de procesos

Algoritmo	**Ventajas**	**Desventajas**	**Eficiencia en tiempo de ejecución**	**Recursos necesarios**
FCFS	Sencillo y fácil de implementar.	Procesos largos bloquean a los cortos.	Baja si hay mucha variabilidad.	Cola FIFO
SJF	Menor tiempo promedio de espera.	Requiere conocer duración; riesgo de inanición.	Alta si se conoce la duración.	Lista ordenada o heap.
SRTF	Excelente tiempo de respuesta.	Mucho cambio de contexto; complejidad alta.	Muy alta si se administra bien.	Cola de prioridad, control continuo de tiempos.
Round Robin	Equitativo; ideal para sistemas interactivos.	Quantum mal ajustado reduce eficiencia.	Media; depende del quantum.	Cola circular, control de quantum.
Selfish Round Robin	Optimiza según prioridad y carga.	Puede romper equidad; requiere mayor control.	Alta con buena lógica de prioridad.	Igual que RR + lógica de prioridad.
Prioridades	Procesos críticos tienen preferencia.	Inanición sin *aging*.	Alta si se aplica *aging*.	Cola de prioridad, sistema de *aging*.
Colas Multinivel	Flexible; asigna algoritmos según el tipo de proceso.	Procesos quedan fijos en su cola.	Alta si se configura correctamente.	Varias colas, reglas fijas por nivel.
Multinivel con Retroalimentación	Dinámico; evita inanición; se adapta al comportamiento del proceso.	Muy complejo de implementar y ajustar.	Muy alta y eficiente.	Varias colas + lógica de promoción/degradación.

En la práctica

1) **Sistemas de Tiempo Real**

Un sistema de tiempo real es aquel donde la corrección del sistema depende no solo de la lógica del resultado, sino también del tiempo en el que se entrega. Se utilizan en aplicaciones críticas como control industrial, dispositivos médicos, automóviles, telecomunicaciones, entre otros.

Clasificación:

- **Sistemas de Tiempo Real Duros:** No toleran ningún incumplimiento en el plazo. Ejemplo: airbags en un automóvil.
- **Sistemas de Tiempo Real Blandos:** Toleran retrasos ocasionales, aunque afectan el rendimiento. Ejemplo: reproducción de video.

Algoritmos comunes:

- **Rate Monotonic Scheduling (RMS):** Algoritmo estático basado en períodos; menor período, mayor prioridad.
- **Earliest Deadline First (EDF):** Dinámico, planifica con base en el proceso con la fecha límite más cercana.
- **Least Laxity First (LLF):** Prioriza procesos con menor tiempo libre entre la ejecución restante y su deadline.

Estos sistemas requieren algoritmos altamente deterministas y predictibles, donde el overhead del cambio de contexto y las interrupciones debe ser mínimo.

2) **Planificación para Linux**

El núcleo de Linux ofrece múltiples políticas de planificación, adaptándose tanto a sistemas de propósito general como a sistemas en tiempo real.

Políticas de planificación en Linux:

- **SCHED_NORMAL o SCHED_OTHER:** Utiliza el Completely Fair Scheduler (CFS), que intenta repartir el tiempo de CPU de forma justa entre todos los procesos.
- **SCHED_FIFO:** Planificación en tiempo real, tipo cola FIFO sin interrupciones hasta que termine o sea bloqueado.
- **SCHED_RR:** Similar a FIFO pero con quantum fijo (Round Robin).

- **SCHED_DEADLINE:** Basado en EDF, permite planificación por deadline, runtime y period.

3) **Planificación para Windows**

El planificador del sistema operativo Windows es híbrido, basado en prioridades dinámicas y estáticas, combinando rendimiento y capacidad de respuesta para distintos tipos de tareas.

Características:

- Prioridades de 0 a 31, donde de 16 a 31 son usadas por procesos en tiempo real.
- Uso de Round Robin dentro del mismo nivel de prioridad.
- La prioridad puede aumentar temporalmente si un proceso interactúa con el usuario (para mejorar la respuesta).
- Windows permite la afinidad de procesos a determinados núcleos.

Tipos de planificación:

- **Tiempo compartido** (time-sharing): Para la mayoría de los procesos de usuario.
- **Tiempo real:** Para procesos críticos que requieren ejecución inmediata.

Windows también soporta hyper-threading, múltiples núcleos y sistemas NUMA.

4) **Planificación en Sistemas Multiprocesadores**

Un sistema multiprocesador tiene varias CPU físicas. La planificación debe decidir no solo qué proceso ejecutar, sino también en qué CPU ejecutarlo, maximizando el uso del hardware.

Estrategias:

- **Planificación global** (global queue): Una sola cola de procesos compartida por todos los CPUs.
- **Planificación por CPU** (per-CPU queue): Cada CPU tiene su propia cola, con planificación local.
- Processor affinity (afinidad de CPU):
    - **Hard affinity**: El proceso solo puede ejecutarse en CPUs específicas.

- **Soft affinity**: El sistema intenta mantener un proceso en la misma CPU para aprovechar la caché.

Balanceo de carga: Importante para distribuir los procesos de manera uniforme.

Linux y Windows implementan mecanismos automáticos para gestionar estas estrategias según la arquitectura y la carga del sistema.

5) **Planificación en Sistemas Multicore**

Un sistema multicore tiene una sola CPU física con múltiples núcleos de procesamiento. Comparte características con sistemas multiprocesador, pero con implicaciones más directas sobre compartición de caché, buses de datos y recursos.

Desafíos adicionales:

- Evitar interferencia en caché compartida entre núcleos.
- Mantener la afinidad del proceso al núcleo, para mejorar la eficiencia.
- Aprovechar el paralelismo fino (multi-threading) mediante bibliotecas como OpenMP, pthreads o TBB.

La planificación puede usar thread pinning, migración dinámica o ajustar la política según el uso del sistema.

6) **Planificación de Hilos**

Un hilo (o thread) es una subunidad de un proceso que comparte el mismo espacio de memoria pero puede ejecutarse de forma concurrente con otros hilos.

Tipos:

- Hilos a nivel de usuario (User-level threads): Gestionados por bibliotecas del usuario; más rápidos, pero no paralelizables en múltiples CPUs sin soporte del sistema operativo.
- Hilos a nivel de núcleo (Kernel-level threads): Gestionados por el SO; más versátiles y capaces de ejecutarse en paralelo real en sistemas multiprocesador o multicore.

Planificación de hilos:

- Similar a la planificación de procesos.
- Se puede aplicar Round Robin, prioridad, CFS, etc.

- Muchos sistemas modernos permiten planificación mixta y balanceo de carga entre hilos activos para maximizar el rendimiento.

Bibliografía

- Silberschatz, A., Galvin, P. B., & Gagne, G. Operating System Concepts - Sección 1.5.1: Process Management.

- Silberschatz, A., Galvin, P. B., & Gagne, G. Operating System Concepts - Sección 5.3.4: Priority Scheduling.

- Silberschatz, A., Galvin, P. B., & Gagne, G. Operating System Concepts - Sección 5.3.5 y 5.3.6: Multilevel Queue Scheduling, Multilevel Feedback Queue Scheduling.

- J. Moral - Planificación de Procesos. Disponible en: http://jmoral.es/blog/planificacion-procesos

- GeeksforGeeks - First Come, First Serve CPU Scheduling (No Preemptivo). Disponible en: https://www.geeksforgeeks.org/first-come-first-serve-cpu-scheduling-non-preemptive/

- GeeksforGeeks - Shortest Job First (SJF) CPU Scheduling (No Preemptivo). Disponible en: https://www.geeksforgeeks.org/program-for-shortest-job-first-or-sjf-cpu-scheduling-set-1-non-preemptive/

- GeeksforGeeks - Introducción al Algoritmo Shortest Remaining Time First (SRTF). Disponible en: https://www.geeksforgeeks.org/introduction-of-shortest-remaining-time-first-srtf-algorithm/

- Quora - Diferencias entre planificación apropiativa y no apropiativa. Disponible en: https://www.quora.com/What-is-the-difference-between-preemptive-scheduling-and-non-preemptive-scheduling