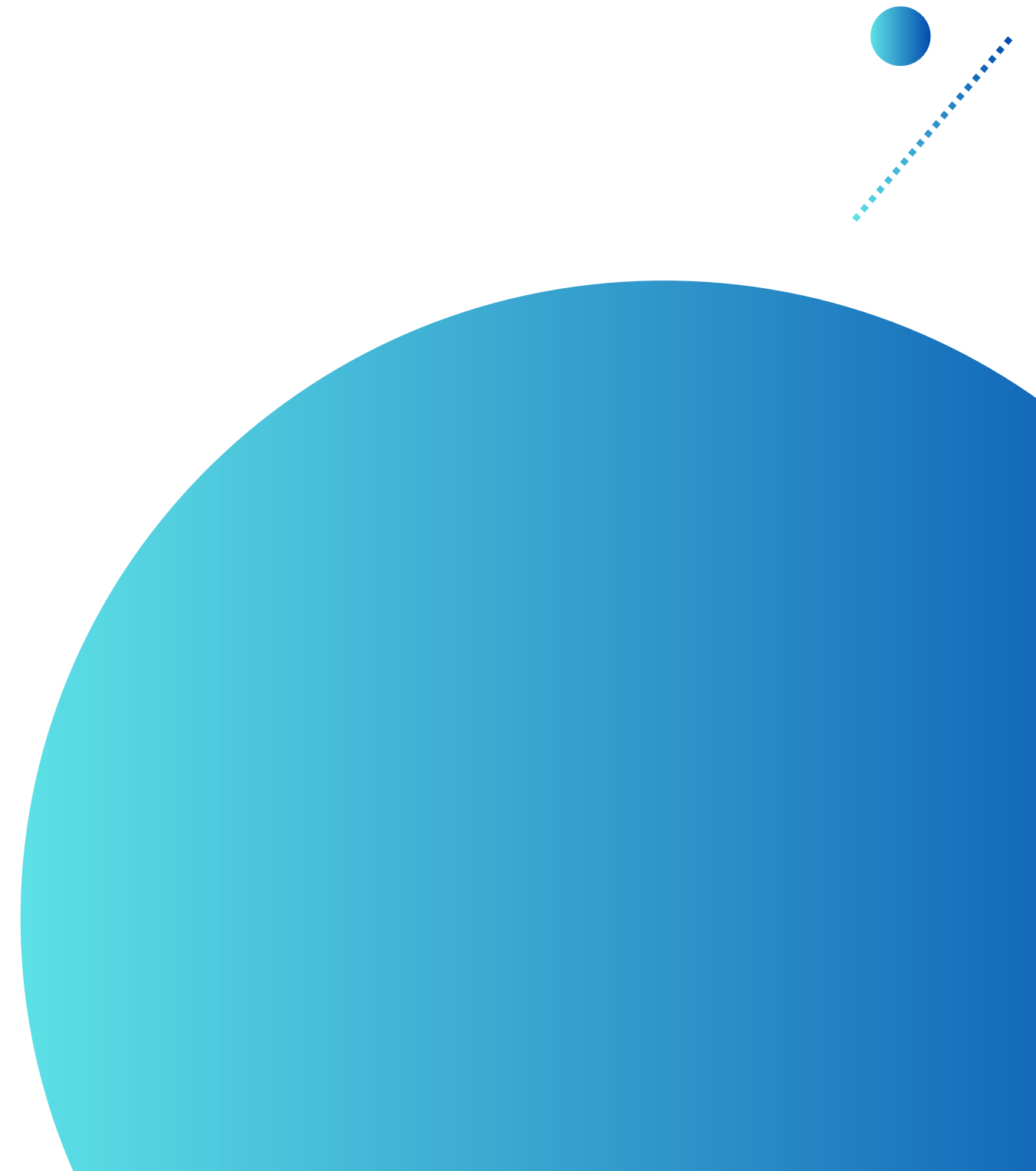


Algoritmos de planificación de procesos

Sistemas Operativos – Zain Rondón



Conceptos básicos

Métricas temporales

Arrival Time (AT): Tiempo de llegada del proceso.

Burst Time (BT): Tiempo de ráfaga (tiempo que el proceso usa la CPU).

Start Time (ST): Tiempo de llegada del proceso.

Finish Time (FT): Tiempo de finalización del proceso.

Conceptos básicos

Tiempos de espera, respuesta y retorno

Response time (RT)	Turn around time (TAT)	Waiting Time (WT)
Intervalo de tiempo transcurrido desde que se emite una solicitud hasta que se comienza a recibir la respuesta	Total de tiempo que toma un proceso en completarse.	Tiempo total que un proceso pasa en la cola de procesos listos antes de ser ejecutado.
$RT = ST - AT$	$TAT = FT - AT$	$WT = TAT - BT$

Conceptos básicos



Algoritmos apropiativos y no apropiativos

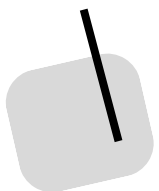
Apropiativos

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 apropiativos

Un algoritmo es no apropiativo cuando no permite la interrupción del proceso actual y solo se podrá ejecutar un nuevo proceso cuando este termine.

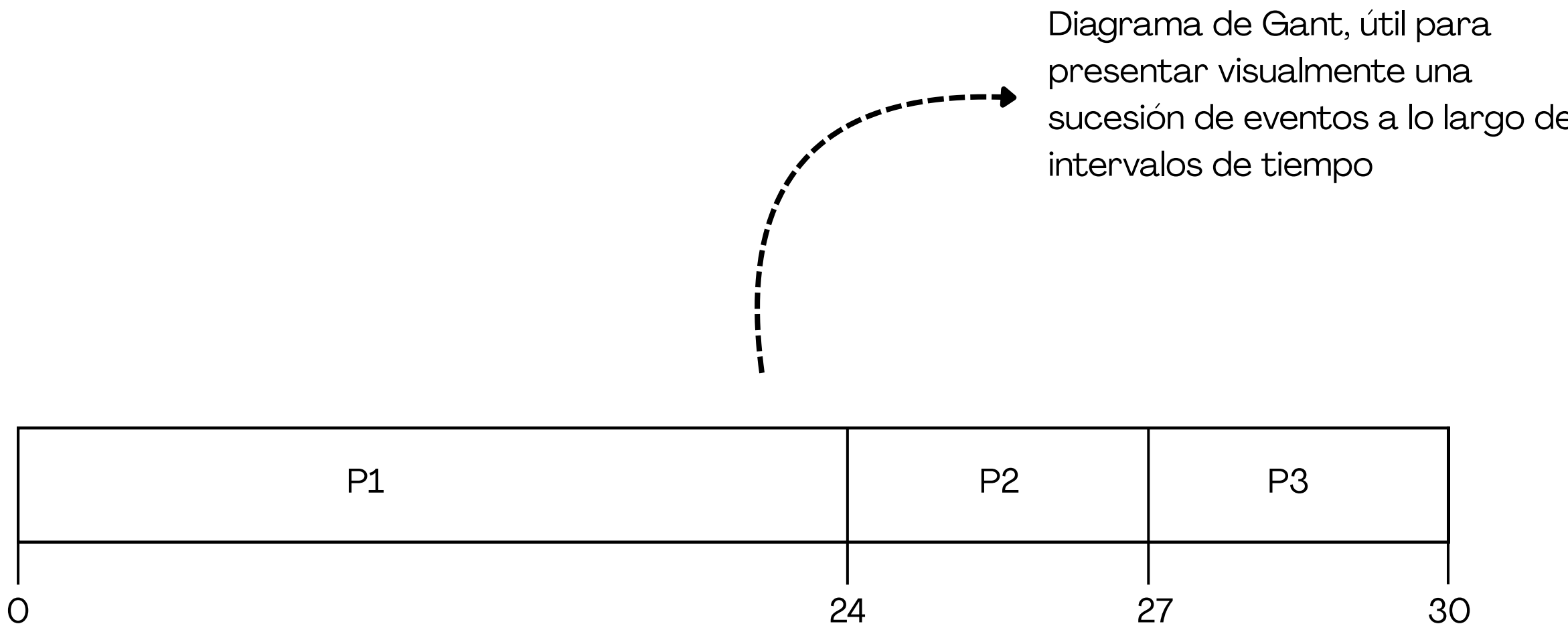
Los sistemas operativos usan ambos tipos de algoritmos según la necesidad o requerimiento que se quiera cubrir.



First Come, First Served

- Los procesos se atienden en el orden en el que llegan usando la metodología FIFO (First In, First Out)
- Es un algoritmo no apropiativo, por lo que una vez que inicia, ningún otro proceso podrá ser ejecutado hasta que el proceso actual termine
- Generalmente presenta tiempos de espera altos
- Apropiado para procesamiento por lotes (Batch)

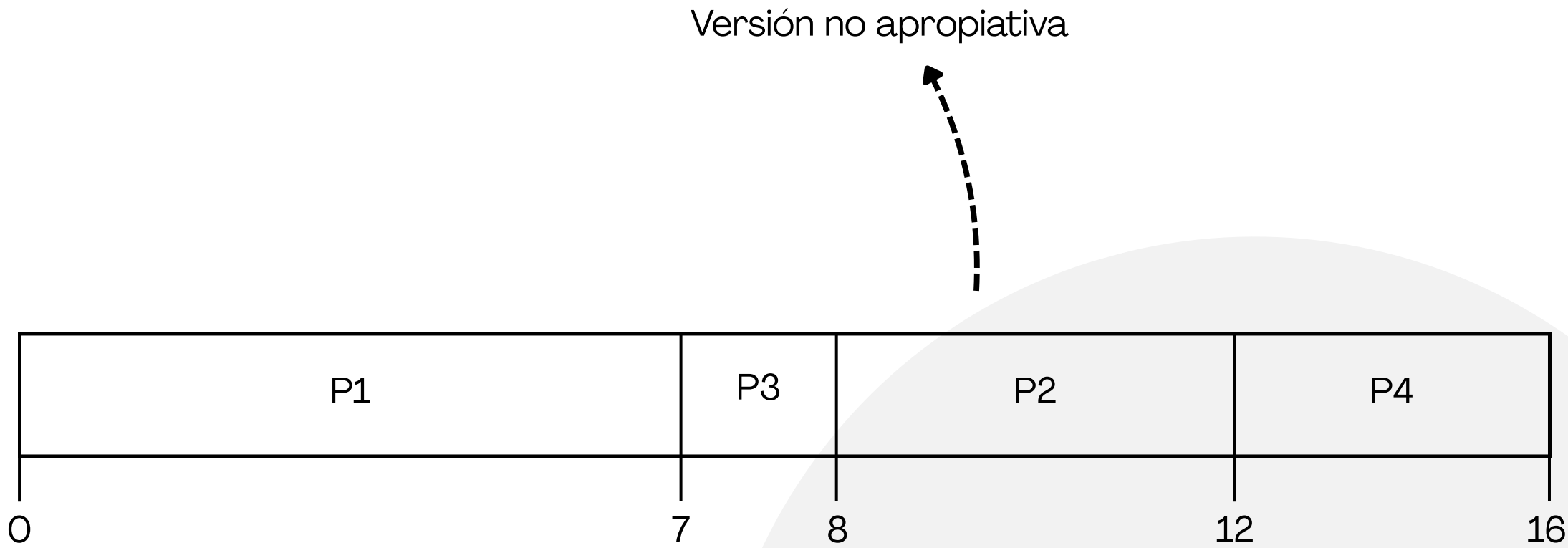
Proceso	Burt Time
P1	24
P2	3
P3	3



Shortest Job First

- Prioriza los procesos con el menor tiempo de ejecución estimado
- Tiene dos enfoques, apropiativo y no apropiativo
- No suele ser muy útil en sistemas en tiempo real, necesita saberse cuanto durara el proceso de antemano, o en su defecto tener una buena estimación
- Presenta tiempos de respuesta más cortos
- Procesos más largos corren el riesgo de inianición

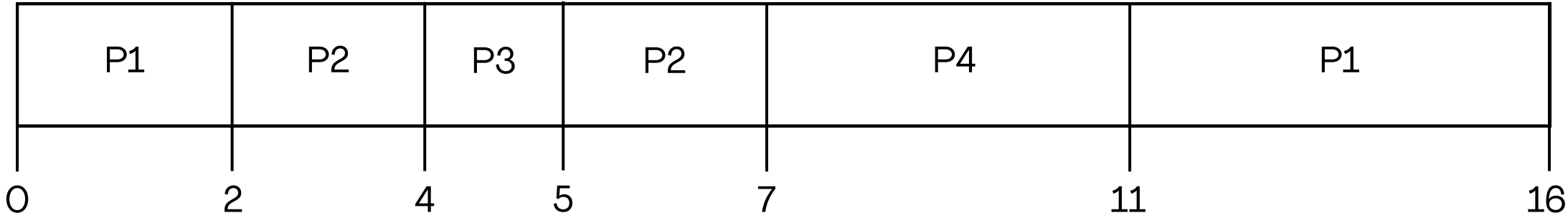
Proceso	Burt Time
P1	7
P2	4
P3	1
P4	4



Shortest Remaining Time First

- Es la versión apropiativa del algoritmo anterior
- Si un nuevo proceso aparece en la lista de procesos listos con menor CPU-burst, se le quita la CPU para asignarla al nuevo proceso.
- Similar al anterior algoritmo, procesos largos corren el riesgo de sufrir inanición con la constante introducción de procesos cuyo CPU Burst sea más corto.

Proceso	Burt Time
P1	7
P2	4
P3	1
P4	4



Round Robin

- Es un algoritmo apropiativo
- Cada proceso tiene un “Quantum”, que es un intervalo temporal de X duración (10–100ms) para ejecutarse
- Al acabarse este intervalo de tiempo, el proceso es interrumpido y llevado al final de la cola para dar paso a otro proceso
- El tiempo del Quantum debe escogerse adecuadamente
 - Si el Quantum es muy corto, ocurrirán muchos cambios de contexto que sobrecargan el sistema
 - Si es muy largo puede comportarse como un FCFS y generaría tiempos de respuesta muy altos
- Todos los procesos tienen la misma prioridad
- Requiere una implementación más elaborada

Round Robin

Existe una variación del algoritmo de Round Robin llamado **Selfish Round Robin**.

Esta versión del algoritmo de Round Robin destaca por lo siguiente:

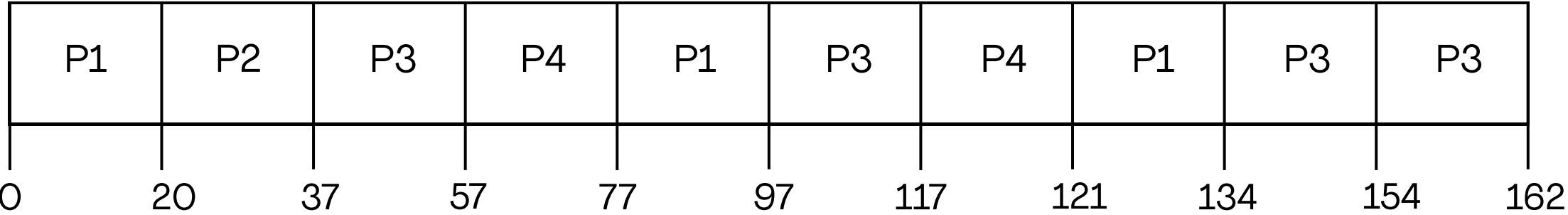
- Algunos procesos pueden tener prioridad sobre otros
 - Su Quantum puede incrementar o se le puede asignar más de un Quantum.
 - Esta prioridad viene dada por parametros como el uso de CPU o el tiempo de ejecución.

Resulta útil para procesos pesados, sin embargo, introduce la necesidad de prevenir la inianición de procesos con menos prioridad.

Round Robin

Proceso	Burt Time
P1	53
P2	17
P3	68
P4	24

Quantum = 20ms



Planificación basada en prioridades

Ya hemos explorado la posibilidad de que ciertos procesos tengan prioridad sobre otros.

En la planificación de procesos con prioridad se distinguen dos tipos principales:

- **Prioridad estática:** La manera más común, es asignar un valor de prioridad fijo para cada proceso y procesarlos según su orden en la Ready queue. Si bien resulta positivo que procesos de mayor prioridad, como eventos de E/S se ejecuten primero, es posible que procesos con poca prioridad nunca se ejecuten (iniciación).
 - Para solucionar este problema se utiliza un mecanismo denominado “aging” o “envejecimiento”, donde a procesos con más tiempo en la cola se les asigna la prioridad más alta.
- **Prioridad dinámica:** Algoritmos como el Selfish RR o el SRTF y el mecanismo de “aging” mencionado anteriormente, son ejemplos de planificación con prioridad dinámica, donde la prioridad se asigna según parámetros que van cambiando a medida que pasa el tiempo.

Planificación basada en prioridades



En general, en una planificación basada en prioridades se utiliza un esquema **apropiativo**, de manera que procesos con mayor prioridad (estática o dinámica) entrantes puedan satisfacer dicha prioridad en un tiempo relativamente corto.

La prioridad puede venir dada según dos enfoques:

- **Factores internos:** Parámetros medibles y cuantificables del sistema, uso de CPU, memoria y almacenamiento, tiempo de ejecución y límites de tiempo pueden ser algunos de estos factores.
- **Factores externos:** Factores que responden a decisiones políticas, administrativas o que responden a la lógica del negocio en el que se encuentre.

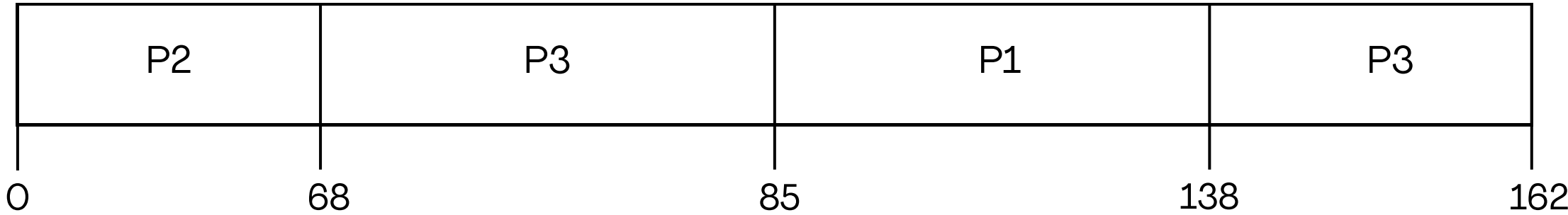
A través de una planificación por prioridad se pueden optimizar los recursos físicos del sistema y la logística requerida por cada negocio u organización.



Planificación basada en prioridades

Proceso	Burt Time	Prioridad
P1	53	3
P2	68	1
P3	17	2
P4	24	4

Ejemplo con prioridades estáticas (tiempo de llgada = 0)



Colas multinivel

La planificación con colas multinivel es una técnica utilizada en sistemas operativos para gestionar procesos según diferentes niveles de prioridad o características:

- En el sistema operativo existen procesos con mayor prioridad que otros
- Se pueden clasificar estos procesos según sus cualidades y naturaleza
- Se utilizan “colas multinivel” para describir conceptualmente estas prioridades:
 - Cada cola puede tener su propio algoritmo de planificación (FCFS, RR, SRTF, etc)
 - Los procesos en colas inferiores esperan a que la cola superior este vacía para comenzar su ejecución
 - Permite tener un control flexible y extensible según que procesos tengan mayor prioridad
- Existe la posibilidad de intercambiar procesos entre colas según varios parametros, esto se conoce como retroalimentación



Colas multinivel

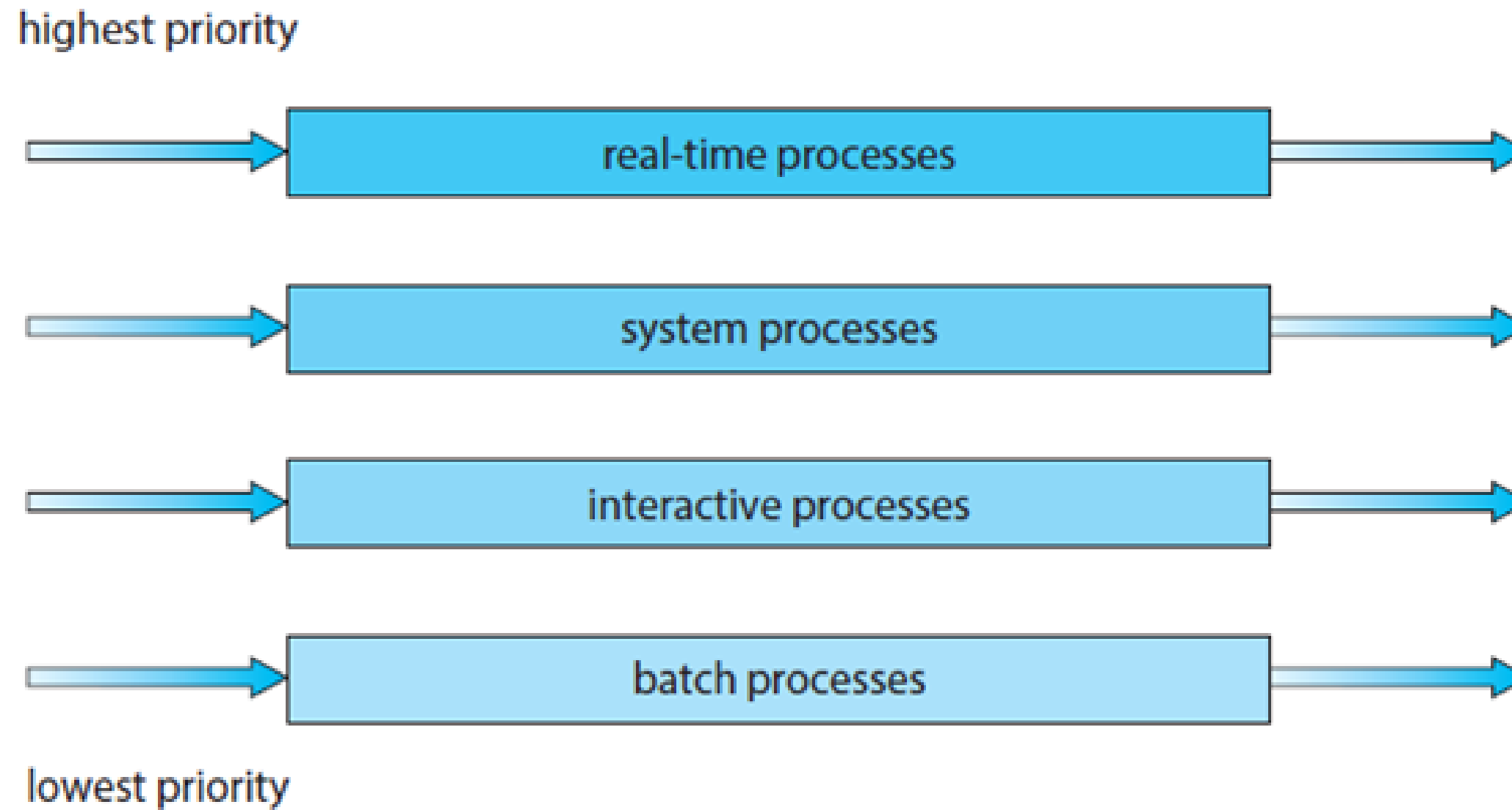


Diagrama conceptual de una Cola de prioridad multinivel, la prioridad más alta la tendrían los procesos en tiempo real o procesos interactivos.

Colas multinivel con retroalimentación

Una cola multinivel con retroalimentación permiten que el sistema operativo mueva procesos de una cola inferior a una superior y viceversa. El proceso de cambio de cola permite reasignar la prioridad de un proceso por varias razones, la más común es para evitar la inanición de procesos con menor prioridad

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

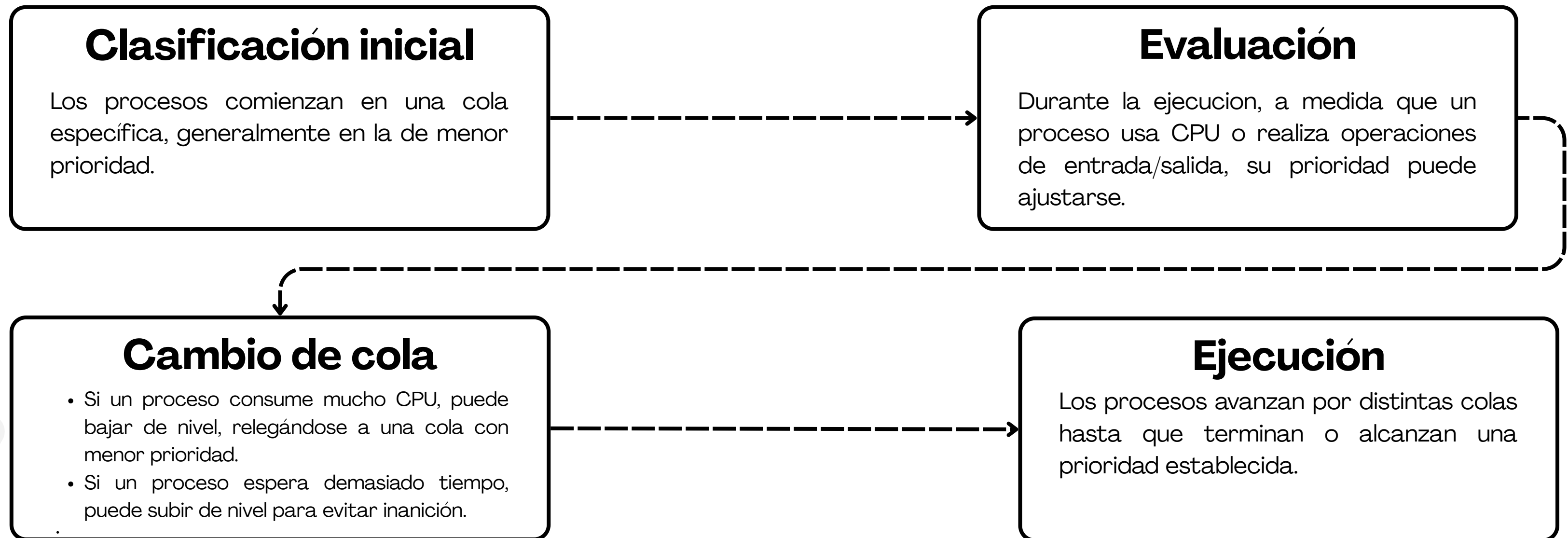


Tabla comparativa

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 <i>aging</i> .	Alta si se aplica <i>aging</i> .	Cola de prioridad, sistema de <i>aging</i> .
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.

Sistemas en tiempo real

En un sistema en tiempo real es crucial que los procesos sean despachados y finalizados en un plazo de tiempo establecido, en este tipo de sistemas no solo es importante el tiempo de respuesta sino la corrección del resultado, que hace referencia a la confiabilidad y exactitud de los resultados entregados.

Para este cometido se utilizan principalmente los siguientes enfoques:

- **Planificación basada en prioridades** (ya expuesta anteriormente)
- **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 o tiempo límite más cercana.
- **Least Laxity First (LLF)**: Prioriza procesos con menor tiempo libre entre la ejecución restante y su deadline.

Estos algoritmos son deterministas y altamente predecibles, y su objetivo principal es garantizar el cumplimiento de los procesos dentro de un plazo determinado

Sistemas en tiempo real

Existen dos tipos de sistemas en tiempo de real:

- **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.

La planificación en sistemas en tiempo real está directamente vinculada con los sistemas operativos, ya que estos deben administrar los recursos de hardware para garantizar que los procesos se ejecuten dentro de los plazos establecidos. En un SO en tiempo real, deben manejarse las siguientes tareas concurrentemente y lo más eficiente en tiempo que se pueda:

- Administración de la CPU
- Manejo de interrupciones
- Gestión de dispositivos y sincronización
- Optimización de kernel


Planificación en Windows y Linux



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

Linux utiliza planificación a corto, medio y largo plazo para decidir qué proceso ocupa la CPU y cuándo.

Linux implementa cinco principales políticas de planificación:

- **SCHED_OTHER:** Para procesos normales en Linux. Utiliza un algoritmo Round-Robin con prioridad dinámica, lo que significa que los procesos compiten por la CPU y su prioridad puede ajustarse en función de su uso.
 - **SCHED_BATCH:** Diseñada para procesos que no requieren una respuesta rápida, como tareas de procesamiento masivo de datos.
 - **SCHED_IDLE:** Se aplica a procesos de muy baja prioridad que solo se ejecutan cuando el sistema está inactivo. Es útil para tareas que pueden esperar indefinidamente sin afectar el rendimiento general.
 - **SCHED_FIFO:** Es una política de tiempo real basada en el principio First-In, First-Out (FIFO). Los procesos con esta política tienen prioridad absoluta sobre los demás y solo ceden la CPU si terminan o son interrumpidos por otro proceso de mayor prioridad.
 - **SCHED_RR:** Similar a SCHED_FIFO, pero con un mecanismo de Round-Robin. Los procesos de tiempo real se ejecutan en turnos de tiempo fijo, lo que permite compartir la CPU entre múltiples procesos de alta prioridad.
- 

Planificación en Windows y Linux

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

- **Prioridades:** Se organizan en un rango de 0 a 31, donde los valores de 16 a 31 están reservados para procesos en tiempo real, asegurando que tareas críticas reciban prioridad sobre procesos estándar.
- **Round Robin:** Dentro de cada nivel de prioridad, Windows utiliza Round Robin, asignando un quantum (unidad de tiempo) para garantizar que los procesos de igual prioridad compartan equitativamente la CPU.
- **Ajuste dinámico de prioridad:** Si un proceso interactúa con el usuario, Windows puede aumentar temporalmente su prioridad para mejorar la capacidad de respuesta y evitar retrasos en la ejecución de aplicaciones interactivas.
- **Afinidad de procesos:** Permite asignar un proceso a núcleos específicos, optimizando el uso de recursos en sistemas multicore.
- **Modos de planificación:** Windows emplea dos enfoques:
 - **Tiempo compartido (Time-sharing):** La mayoría de los procesos de usuario utilizan este esquema, donde los recursos se distribuyen equitativamente.
 - **Tiempo real:** Reservado para procesos críticos que requieren máxima prioridad y mínima latencia.

Planificación en Windows y Linux

La planificación de procesos en Linux y Windows tiene varias similitudes, a pesar de sus diferencias en implementación. Aquí algunos puntos en común:

- **Uso de prioridades:** Ambos sistemas asignan prioridades a los procesos para determinar el orden de ejecución.
- **Multiprocesamiento:** Tanto Linux como Windows soportan múltiples núcleos y procesadores para mejorar el rendimiento, con técnicas como Multiprocesamiento simétrico (SMP) o Multiprocesamiento asimétrico (AMP)
- **Planificación basada en tiempos:** Ambos utilizan algoritmos como Round-Robin para distribuir el tiempo de CPU entre procesos.
- **Gestión de hilos:** Ambos sistemas permiten la ejecución de múltiples hilos dentro de un proceso para mejorar la eficiencia.
- **Interrupciones y cambios de contexto:** Ambos manejan interrupciones y cambios de contexto para alternar entre procesos de manera eficiente

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.

La planificación multiprocesador es un aspecto clave en sistemas operativos modernos, ya que permite distribuir la carga de trabajo entre múltiples CPU para mejorar el rendimiento y la eficiencia.

Tipos de multiprocesamiento

- **Multiprocesamiento simétrico (SMP):** Todos los procesadores tienen acceso equitativo a la memoria y pueden ejecutar cualquier tarea.
- **Multiprocesamiento asimétrico (AMP):** Un procesador principal asigna tareas a procesadores secundarios, optimizando la ejecución de ciertas funciones.

Planificación en Sistemas Multiprocesadores

- **Estrategias de planificación**

- **Cola global (Global Queue):** Todos los procesos esperan en una única cola y cualquier CPU disponible los ejecuta.
- **Cola por CPU (Per-CPU Queue):** Cada CPU tiene su propia cola de procesos, reduciendo la competencia entre núcleos.

- **Afinidad de CPU:**

- **Hard Affinity:** Un proceso solo puede ejecutarse en CPUs específicas.
- **Soft Affinity:** Se intenta mantener un proceso en la misma CPU para aprovechar la caché.

- **Consideraciones en la planificación**

- **Balanceo de carga:** Distribuir procesos equitativamente entre CPUs para evitar sobrecarga en un solo núcleo.
- **Migración de procesos:** Permitir que los procesos cambien de CPU si es necesario para mejorar la eficiencia.
- **NUMA (Non-Uniform Memory Access):** Optimiza el acceso a la memoria en sistemas con múltiples procesadores.

Sistemas Multicore y Multiprocesador

La planificación de procesos en sistemas multicore y multiprocesador es fundamental para optimizar el uso del hardware y garantizar un rendimiento eficiente. Aunque ambos modelos permiten la ejecución simultánea de múltiples procesos, su infraestructura y gestión presentan diferencias clave.

Aspecto	Multicore	Multiprocesador
Definición	Una CPU con múltiples núcleos.	Varias CPU físicas trabajando juntas.
Acceso a memoria	Los núcleos comparten la misma memoria y caché.	Cada CPU puede tener su propia memoria o compartirla con otras CPUs.
Planificación	Decide qué núcleo ejecutará cada proceso dentro de la misma CPU.	Debe asignar procesos a CPU específicas, además de decidir cuándo ejecutarlos.
Estrategias usadas	Afinidad de CPU, balanceo de carga, planificación local.	Afinidad de CPU, planificación global o por CPU, NUMA.
Ventajas	Mayor eficiencia energética, reducción de latencia en comunicación interna.	Escalabilidad y capacidad para manejar múltiples tareas pesadas.

Planificación de Hilos



La planificación de hilos es un componente esencial en los sistemas operativos modernos, ya que permite la ejecución eficiente de múltiples tareas simultáneamente. Los hilos son unidades de ejecución dentro de un proceso, y su planificación determina cómo se asignan los recursos del sistema para garantizar un rendimiento óptimo.

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. Existen dos tipos de hilos:

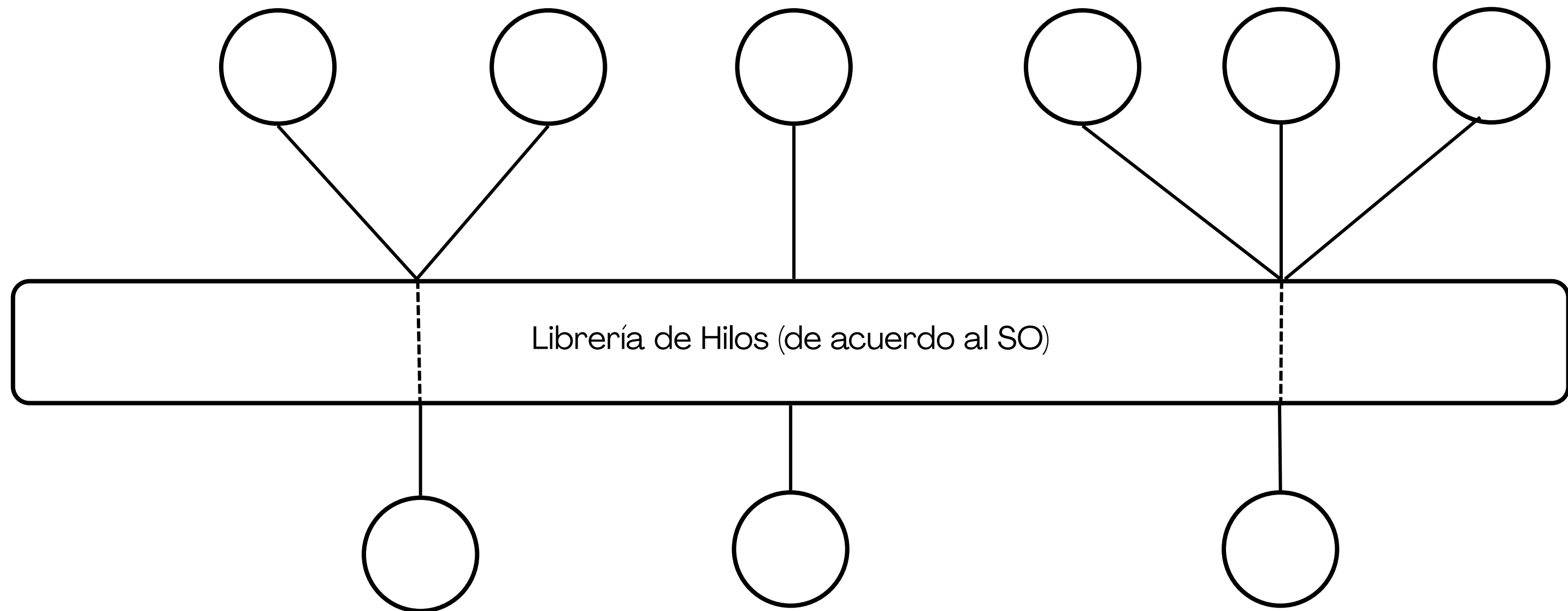
- **Hilos a nivel de usuario (User-level threads):** Gestionados por bibliotecas del usuario; más rápidos, pero no paralelizables en múltiples CPUs sin soporte del sistema operativo, son más eficientes en cambios de contexto porque no requieren intervención del kernel, pero tienen menos acceso a recursos del sistema.
- **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.

Cada sistema operativo ofrece al usuario librerías para el manejo de hilos, y en la mayoría de la veces esto resulta en un manejo híbrido, utilizando hilos tanto a nivel de usuario como a nivel de kernel.

Aunque los hilos comparten el mismo espacio de memoria dentro de un proceso, cada hilo puede manejar su propia pila y recursos específicos. Esto les permite ejecutar tareas concurrentemente sin la sobrecarga de crear procesos completos.

Planificación de Hilos

Hilos a nivel de usuario



Hilos a nivel de kernel