

Planificación de Procesos

Scheduler

David Alejandro González Márquez

Clase disponible en: <https://github.com/fokerman/computingSystemsCourse>

Planificador o *Scheduler*

En un sistema donde múltiples procesos son ejecutados compartiendo tiempo de CPU, debe existir un componente encargado de coordinar su uso.

Planificador o *Scheduler*

En un sistema donde múltiples procesos son ejecutados compartiendo tiempo de CPU, debe existir un componente encargado de coordinar su uso.

scheduler o planificador

Componente del Sistema Operativo encargado de **coordinar** el uso del procesador o procesadores por los procesos en ejecución. Decide en que momento un proceso debe ser ejecutado y cuando su ejecución debe ser detenida.

Planificador o *Scheduler*

En un sistema donde múltiples procesos son ejecutados compartiendo tiempo de CPU, debe existir un componente encargado de coordinar su uso.

scheduler o planificador

Componente del Sistema Operativo encargado de **coordinar** el uso del procesador o procesadores por los procesos en ejecución. Decide en que momento un proceso debe ser ejecutado y cuando su ejecución debe ser detenida.

La implementación del *scheduler* es crítica para evaluar el desempeño de los Sistemas Operativos.

Una gran parte del esfuerzo de optimización y captura de métricas tiene foco en el *scheduler*, sin embargo **no existe un algoritmo de planificación perfecto**.

El desempeño de los *scheduler* dependen del **workload al que este expuesto el sistema**.

Criterios para la planificación

Los algoritmos de planificación miden su eficiencia según los **objetivos que buscan optimizar**.

Criterios para la planificación

Los algoritmos de planificación miden su eficiencia según los **objetivos que buscan optimizar**.

Tiempo de ejecución (turnaround)

Tiempo que demora un proceso en completar su ejecución.

Crterios para la planificacin

Los algoritmos de planificacin miden su eficiencia segn los **objetivos que buscan optimizar**.

Tiempo de ejecucin (turnaround)

Tiempo que demora un proceso en completar su ejecucin.

Utilizacin de la CPU

Porcentaje de uso de la CPU.

Crterios para la planificacin

Los algoritmos de planificacin miden su eficiencia segn los **objetivos que buscan optimizar**.

Tiempo de ejecucin (turnaround)

Tiempo que demora un proceso en completar su ejecucin.

Utilizacin de la CPU

Porcentaje de uso de la CPU.

Tiempo de respuesta

Tiempo percibido por el usuario en procesos interactivos.

Criterios para la planificación

Los algoritmos de planificación miden su eficiencia según los **objetivos que buscan optimizar**.

Tiempo de ejecución (turnaround)

Tiempo que demora un proceso en completar su ejecución.

Utilización de la CPU

Porcentaje de uso de la CPU.

Tiempo de respuesta

Tiempo percibido por el usuario en procesos interactivos.

Rendimiento (throughput)

La cantidad de procesos que se terminan por unidad de tiempo.

Criterios para la planificación

Los algoritmos de planificación miden su eficiencia según los **objetivos que buscan optimizar**.

Tiempo de ejecución (turnaround)

Tiempo que demora un proceso en completar su ejecución.

Utilización de la CPU

Porcentaje de uso de la CPU.

Tiempo de respuesta

Tiempo percibido por el usuario en procesos interactivos.

Rendimiento (throughput)

La cantidad de procesos que se terminan por unidad de tiempo.

Tiempo de espera

Tiempo entre que el proceso está preparado y es ejecutado.

Criterios para la planificación

Los algoritmos de planificación miden su eficiencia según los **objetivos que buscan optimizar**.

Tiempo de ejecución (turnaround)

Tiempo que demora un proceso en completar su ejecución.

Utilización de la CPU

Porcentaje de uso de la CPU.

Tiempo de respuesta

Tiempo percibido por el usuario en procesos interactivos.

Rendimiento (throughput)

La cantidad de procesos que se terminan por unidad de tiempo.

Tiempo de espera

Tiempo entre que el proceso está preparado y es ejecutado.

Ahora, ¿Qué sucede si un proceso conoce el criterio de planificación y busca aprovecharlo para usar más tiempo el CPU?

Un *malware* podría y el sistema debería detectarlo, ya que en caso contrario afectaría al resto de los procesos ejecutando en el sistema.

Intercambio de procesos

El intercambio de procesos se puede dar bajo las siguientes situaciones:

Intercambio de procesos

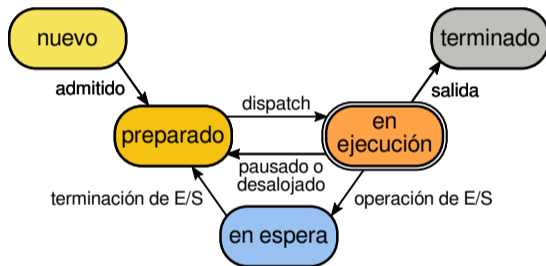
El intercambio de procesos se puede dar bajo las siguientes situaciones:

- **desalojado:** en ejecución → preparado

El proceso deja de ser ejecutado cuando termina su tiempo asignado (*quantum*).

- **pausado:** en ejecución → preparado

El proceso deja de ser ejecutado producto de una *interrupción*.



Intercambio de procesos

El intercambio de procesos se puede dar bajo las siguientes situaciones:

- **desalojado:** en ejecución → preparado

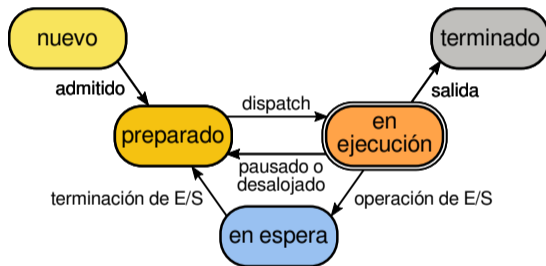
El proceso deja de ser ejecutado cuando termina su tiempo asignado (*quantum*).

- **pausado:** en ejecución → preparado

El proceso deja de ser ejecutado producto de una **interrupción**.

- **operación de E/S:** en ejecución → en espera

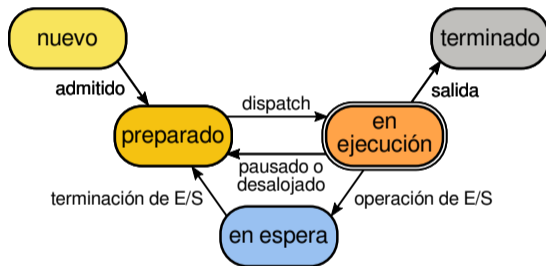
El proceso deja de ser ejecutado producto de una **solicitud de E/S**.



Intercambio de procesos

El intercambio de procesos se puede dar bajo las siguientes situaciones:

- **desalojado:** en ejecución → preparado
El proceso deja de ser ejecutado cuando termina su tiempo asignado (*quantum*).
- **pausado:** en ejecución → preparado
El proceso deja de ser ejecutado producto de una **interrupción**.
- **operación de E/S:** en ejecución → en espera
El proceso deja de ser ejecutado producto de una **solicitud de E/S**.
- **salida:** en ejecución → terminado
El proceso en ejecución **termina**.

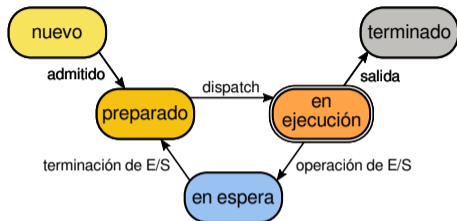


Tipos de planificación

Tipos de planificación

Sin desalojo o Cooperativa

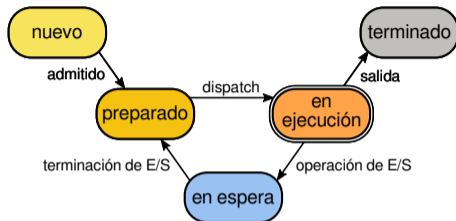
El proceso solo puede dejar de ejecutar si termina, o realiza una operación de E/S.



Tipos de planificación

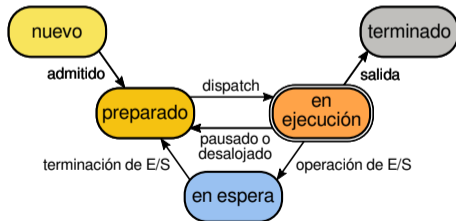
Sin desalojo o Cooperativa

El proceso solo puede dejar de ejecutar si termina, o realiza una operación de E/S.



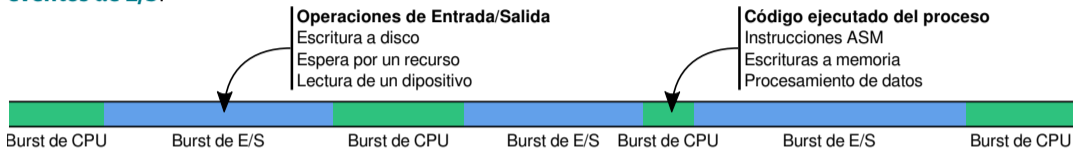
Con desalojo o Apropiativa

El proceso puede dejar de ejecutar bajo cualquier condición, incluso si el planificador así lo decide.



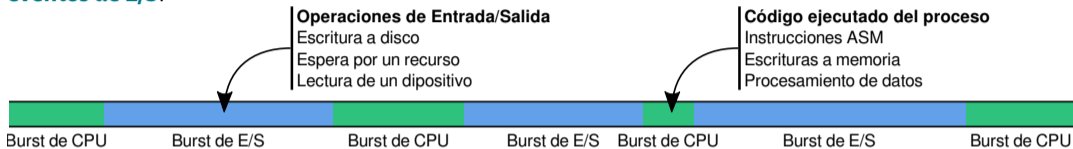
Comportamiento de los procesos

Para el planificador un proceso puede entenderse como secuencia intercalada de **instrucciones** y **eventos de E/S**.



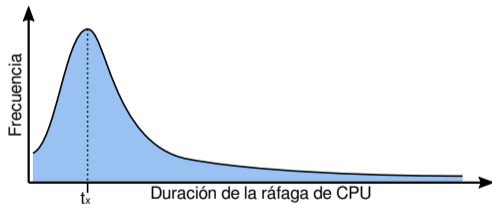
Comportamiento de los procesos

Para el planificador un proceso puede entenderse como secuencia intercalada de **instrucciones** y **eventos de E/S**.



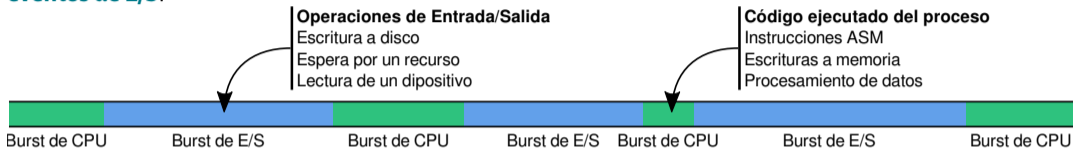
Dependiendo del tipo de proceso, puede generar mayor o menor cantidad de eventos de entrada/salida.

La figura indica la distribución de tiempos entre que un proceso comienza a ser ejecutado hasta que realiza un evento de E/S.



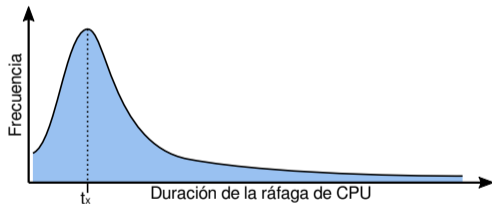
Comportamiento de los procesos

Para el planificador un proceso puede entenderse como secuencia intercalada de **instrucciones** y **eventos de E/S**.



Dependiendo del tipo de proceso, puede generar mayor o menor cantidad de eventos de entrada/salida.

La figura indica la distribución de tiempos entre que un proceso comienza a ser ejecutado hasta que realiza un evento de E/S.



Las ráfagas o *burst* de E/S suelen durar **mucho más tiempo**, que las ráfagas de CPU.

Por esta razón los procesos están la mayor parte del tiempo esperando que finalicen operaciones de E/S.

Tiempo de ráfaga

Cuando aparece un **nuevo proceso**, no tenemos información de cuanto tiempo va a demorar ejecutando.

Entonces, ¿Cómo podemos estimar el tiempo de ráfaga?

Tiempo de ráfaga

Cuando aparece un **nuevo proceso**, no tenemos información de cuanto tiempo va a demorar ejecutando.

Entonces, **¿Cómo podemos estimar el tiempo de ráfaga?**

Una vez que ejecuta por primera vez, reportamos su tiempo de ráfaga.

Predecimos a partir de ahí, el próximo tiempo de ráfaga en base a las ráfagas anteriores.

La estimación se realiza usando un promedio entre las ráfagas, dando mayor peso a las ráfagas más recientes y menor peso a las ráfagas más antiguas.

Tiempo de ráfaga

Cuando aparece un **nuevo proceso**, no tenemos información de cuanto tiempo va a demorar ejecutando.

Entonces, **¿Cómo podemos estimar el tiempo de ráfaga?**

Una vez que ejecuta por primera vez, reportamos su tiempo de ráfaga.

Predecimos a partir de ahí, el próximo tiempo de ráfaga en base a las ráfagas anteriores.

La estimación se realiza usando un promedio entre las ráfagas, dando mayor peso a las ráfagas más recientes y menor peso a las ráfagas más antiguas.

En general se utiliza el **promedio exponencial** que se calcula como:

$$\tau_{n+1} = \alpha \cdot t_n + (1 - \alpha) \cdot \tau_n$$

Donde t_n es el tiempo de ráfaga en el instante n , y τ_{n+1} es el tiempo de ráfaga estimado.

El coeficiente α regula la incidencia de las ráfagas anteriores en el cálculo.

Para $\alpha = 0$ la última ráfaga no tiene ningún peso, mientras que para $\alpha = 1$ se considera solo la última. En general, se suele usar $\alpha = 0,5$.

Algoritmos de planificación

Los algoritmos pueden ser muy variados, vamos a conocer algunos de estos, entendiendo funcionamiento, ventajas y desventajas.

- **First-come, first-served (FCFS)**
- **Round Robin (RR)**
- **Shortest Job First (SJF)**
- **Shortest Remaining Time First (SRTF)**
- **Priority Scheduling**
- **Multilevel Queue Scheduling**
- **Multilevel Feedback Queue Scheduling**

Además, para ilustrar su funcionamiento vamos a construir *diagramas de Gantt*, indicando como cada algoritmo asigna procesos al CPU y calculando el tiempo promedio de espera para medir su eficiencia.

Algoritmos de planificación: First-come, first-served (FCFS)

Los procesos **se ejecutan según el orden de llegada**. Se trata de un algoritmo simple y **cooperativo**, ya que los procesos nunca son desalojados.

Algoritmos de planificación: First-come, first-served (FCFS)

Los procesos **se ejecutan según el orden de llegada**. Se trata de un algoritmo simple y **cooperativo**, ya que los procesos nunca son desalojados.

Se implementa utilizando una cola de procesos ordenados según su llegada.

Algoritmos de planificación: First-come, first-served (FCFS)

Los procesos **se ejecutan según el orden de llegada**. Se trata de un algoritmo simple y **cooperativo**, ya que los procesos nunca son desalojados.

Se implementa utilizando una cola de procesos ordenados según su llegada.

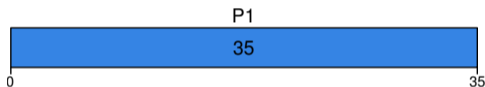
Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	35
P2	0	10
P3	0	5

Algoritmos de planificación: First-come, first-served (FCFS)

Los procesos **se ejecutan según el orden de llegada**. Se trata de un algoritmo simple y **cooperativo**, ya que los procesos nunca son desalojados.

Se implementa utilizando una cola de procesos ordenados según su llegada.



Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	35
P2	0	10
P3	0	5

Algoritmos de planificación: First-come, first-served (FCFS)

Los procesos **se ejecutan según el orden de llegada**. Se trata de un algoritmo simple y **cooperativo**, ya que los procesos nunca son desalojados.

Se implementa utilizando una cola de procesos ordenados según su llegada.



Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	35
P2	0	10
P3	0	5

Algoritmos de planificación: First-come, first-served (FCFS)

Los procesos **se ejecutan según el orden de llegada**. Se trata de un algoritmo simple y **cooperativo**, ya que los procesos nunca son desalojados.

Se implementa utilizando una cola de procesos ordenados según su llegada.



Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	35
P2	0	10
P3	0	5

Algoritmos de planificación: First-come, first-served (FCFS)

Los procesos **se ejecutan según el orden de llegada**. Se trata de un algoritmo simple y **cooperativo**, ya que los procesos nunca son desalojados.

Se implementa utilizando una cola de procesos ordenados según su llegada.



Tiempo de espera promedio = $\frac{0+35+45}{3} = \frac{80}{3} = 26,66$ ms.

Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	35
P2	0	10
P3	0	5

Algoritmos de planificación: First-come, first-served (FCFS)

Los procesos **se ejecutan según el orden de llegada**. Se trata de un algoritmo simple y **cooperativo**, ya que los procesos nunca son desalojados.

Se implementa utilizando una cola de procesos ordenados según su llegada.



Tiempo de espera promedio = $\frac{0+35+45}{3} = \frac{80}{3} = 26,66$ ms.

Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	35
P2	0	10
P3	0	5

Algoritmos de planificación: First-come, first-served (FCFS)

Los procesos **se ejecutan según el orden de llegada**. Se trata de un algoritmo simple y **cooperativo**, ya que los procesos nunca son desalojados.

Se implementa utilizando una cola de procesos ordenados según su llegada.



$$\text{Tiempo de espera promedio} = \frac{0+35+45}{3} = \frac{80}{3} = 26,66 \text{ ms.}$$

En los diagramas se ignora el tiempo de *task switch*.

Este tiempo se debe considerar cada vez que se cambia de un proceso a otro.

Los algoritmos de *scheduling* lo hacen para reducir los tiempos muertos.

Para el ejemplo, la cantidad de *task switch* = 2.

Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	35
P2	0	10
P3	0	5

Algoritmos de planificación: First-come, first-served (FCFS)

¿Qué sucede si en el ejemplo anterior el orden de llegada de los procesos fuera a la inversa?

Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P3	0	5
P2	0	10
P1	0	35

Algoritmos de planificación: First-come, first-served (FCFS)

¿Qué sucede si en el ejemplo anterior el orden de llegada de los procesos fuera a la inversa?

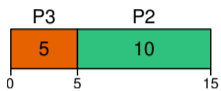


Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P3	0	5
P2	0	10
P1	0	35

Algoritmos de planificación: First-come, first-served (FCFS)

¿Qué sucede si en el ejemplo anterior el orden de llegada de los procesos fuera a la inversa?



Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P3	0	5
P2	0	10
P1	0	35

Algoritmos de planificación: First-come, first-served (FCFS)

¿Qué sucede si en el ejemplo anterior el orden de llegada de los procesos fuera a la inversa?



Tiempo de espera promedio = $\frac{0+5+15}{3} = \frac{20}{3} = 6,66$ ms.

Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P3	0	5
P2	0	10
P1	0	35

Algoritmos de planificación: First-come, first-served (FCFS)

¿Qué sucede si en el ejemplo anterior el orden de llegada de los procesos fuera a la inversa?



Tiempo de espera promedio = $\frac{0+5+15}{3} = \frac{20}{3} = 6,66$ ms.

Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P3	0	5
P2	0	10
P1	0	35

FCFS es un algoritmo muy simple de implementar y aprovecha la CPU al máximo, ya que los procesos no son desalojados. Evitando el *overhead* del *task switch*.

Por otro lado, un proceso largo puede tomar el procesador por mucho tiempo, demorando al resto de los procesos (*efecto convoy*).

Este efecto se ve en el ejemplo anterior para el proceso P1.

Algoritmos de planificación: Round Robin (RR)

Cada proceso tiene asignado un **quantum** (tiempo máximo que puede correr). **Cuando un proceso consume su *quantum*, se lo desaloja y se ejecuta el siguiente proceso.**

Algoritmo similar al FCFS pero con **desalojo**.

Algoritmos de planificación: Round Robin (RR)

Cada proceso tiene asignado un **quantum** (tiempo máximo que puede correr). **Cuando un proceso consume su *quantum*, se lo desaloja y se ejecuta el siguiente proceso.**

Algoritmo similar al FCFS pero con **desalojo**.

Se implementa utilizando una lista circular que se recorre en orden.

Algoritmos de planificación: Round Robin (RR)

Cada proceso tiene asignado un **quantum** (tiempo máximo que puede correr). **Cuando un proceso consume su *quantum*, se lo desaloja y se ejecuta el siguiente proceso.**

Algoritmo similar al FCFS pero con **desalojo**.

Se implementa utilizando una lista circular que se recorre en orden.

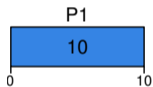
Ejemplo		<i>quantum</i> = 10
Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	25
P2	0	20
P3	0	5

Algoritmos de planificación: Round Robin (RR)

Cada proceso tiene asignado un **quantum** (tiempo máximo que puede correr). **Cuando un proceso consume su *quantum*, se lo desaloja y se ejecuta el siguiente proceso.**

Algoritmo similar al FCFS pero con **desalojo**.

Se implementa utilizando una lista circular que se recorre en orden.



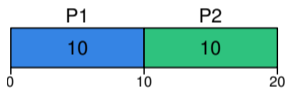
Ejemplo		<i>quantum</i> = 10
Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	25
P2	0	20
P3	0	5

Algoritmos de planificación: Round Robin (RR)

Cada proceso tiene asignado un **quantum** (tiempo máximo que puede correr). **Cuando un proceso consume su *quantum*, se lo desaloja y se ejecuta el siguiente proceso.**

Algoritmo similar al FCFS pero con **desalojo**.

Se implementa utilizando una lista circular que se recorre en orden.



Ejemplo

quantum = 10

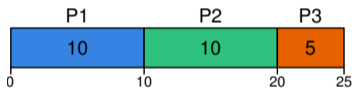
Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	25
P2	0	20
P3	0	5

Algoritmos de planificación: Round Robin (RR)

Cada proceso tiene asignado un **quantum** (tiempo máximo que puede correr). **Cuando un proceso consume su *quantum*, se lo desaloja y se ejecuta el siguiente proceso.**

Algoritmo similar al FCFS pero con **desalojo**.

Se implementa utilizando una lista circular que se recorre en orden.



Ejemplo

$quantum = 10$

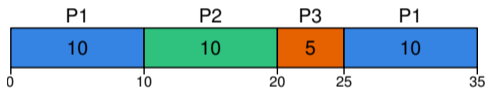
Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	25
P2	0	20
P3	0	5

Algoritmos de planificación: Round Robin (RR)

Cada proceso tiene asignado un **quantum** (tiempo máximo que puede correr). **Cuando un proceso consume su *quantum*, se lo desaloja y se ejecuta el siguiente proceso.**

Algoritmo similar al FCFS pero con **desalojo**.

Se implementa utilizando una lista circular que se recorre en orden.



Ejemplo

$quantum = 10$

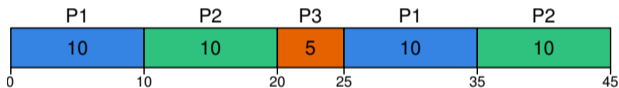
Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	25
P2	0	20
P3	0	5

Algoritmos de planificación: Round Robin (RR)

Cada proceso tiene asignado un **quantum** (tiempo máximo que puede correr). **Cuando un proceso consume su *quantum*, se lo desaloja y se ejecuta el siguiente proceso.**

Algoritmo similar al FCFS pero con **desalojo**.

Se implementa utilizando una lista circular que se recorre en orden.



Ejemplo

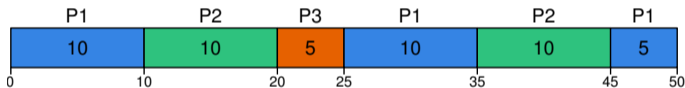
Proceso	<i>quantum</i> = 10	
	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	25
P2	0	20
P3	0	5

Algoritmos de planificación: Round Robin (RR)

Cada proceso tiene asignado un **quantum** (tiempo máximo que puede correr). **Cuando un proceso consume su *quantum*, se lo desaloja y se ejecuta el siguiente proceso.**

Algoritmo similar al FCFS pero con **desalojo**.

Se implementa utilizando una lista circular que se recorre en orden.



Ejemplo

quantum = 10

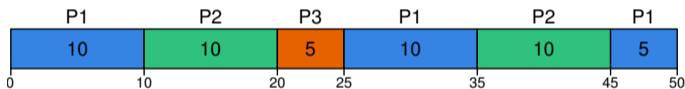
Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	25
P2	0	20
P3	0	5

Algoritmos de planificación: Round Robin (RR)

Cada proceso tiene asignado un **quantum** (tiempo máximo que puede correr). **Cuando un proceso consume su quantum, se lo desaloja y se ejecuta el siguiente proceso.**

Algoritmo similar al FCFS pero con **desalojo**.

Se implementa utilizando una lista circular que se recorre en orden.



Ejemplo

quantum = 10

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	25
P2	0	20
P3	0	5

$$\text{Tiempo de espera promedio} = \frac{((25-10) + (45-35)) + ((10-0) + (35-20)) + (20-0)}{3} = \frac{25+25+20}{3} = \frac{70}{3} = 23,33 \text{ ms.}$$

Algoritmos de planificación: Round Robin (RR)

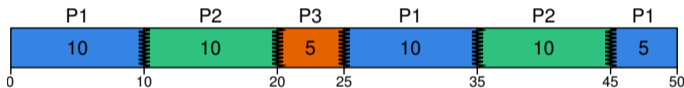
Cada proceso tiene asignado un **quantum** (tiempo máximo que puede correr). **Cuando un proceso consume su quantum, se lo desaloja y se ejecuta el siguiente proceso.**

Algoritmo similar al FCFS pero con **desalojo**.

Se implementa utilizando una lista circular que se recorre en orden.

Ejemplo

quantum = 10		
Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	25
P2	0	20
P3	0	5



Cantidad de *task switch* = 5.

$$\text{Tiempo de espera promedio} = \frac{((25-10) + (45-35)) + ((10-0) + (35-20)) + (20-0)}{3} = \frac{25+25+20}{3} = \frac{70}{3} = 23,33 \text{ ms.}$$

Algoritmos de planificación: Round Robin (RR)

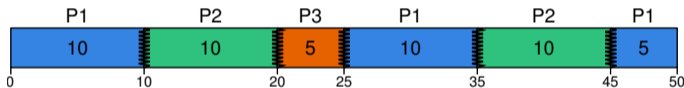
Cada proceso tiene asignado un **quantum** (tiempo máximo que puede correr). **Cuando un proceso consume su quantum, se lo desaloja y se ejecuta el siguiente proceso.**

Algoritmo similar al FCFS pero con **desalojo**.

Se implementa utilizando una lista circular que se recorre en orden.

Ejemplo

quantum = 10		
Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	25
P2	0	20
P3	0	5



Cantidad de *task switch* = 5.

$$\text{Tiempo de espera promedio} = \frac{((25-10) + (45-35)) + ((10-0) + (35-20)) + (20-0)}{3} = \frac{25+25+20}{3} = \frac{70}{3} = 23,33 \text{ ms.}$$

Duración de *quantum*, ¿fijo o variable?

Si es muy corto, gran parte del tiempo estará dedicado a planificar y a hacer cambios de contexto.

Si es muy largo, degenera en FCFS, y se vuelve inadecuado para procesos interactivos.

En la práctica se usan *quantums* de entre 10 y 100 ms mientras que el cambio de contexto toma 10 μs (microsegundos).

Es decir, un cambio de contexto equivale a 0,001 *quantums*.

Algoritmos de planificación: Shortest Job First (SJF)

Se ordenan los procesos según su tiempo de ráfaga, **se elige al proceso con la ráfaga de CPU más corta**. Si tienen el mismo tiempo de ráfaga, se elige al que llega primero.

Algoritmos de planificación: Shortest Job First (SJF)

Se ordenan los procesos según su tiempo de ráfaga, **se elige al proceso con la ráfaga de CPU más corta**. Si tienen el mismo tiempo de ráfaga, se elige al que llegó primero.

Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	10
P2	0	3
P3	0	30
P4	0	4
P5	0	3

Algoritmos de planificación: Shortest Job First (SJF)

Se ordenan los procesos según su tiempo de ráfaga, **se elige al proceso con la ráfaga de CPU más corta**. Si tienen el mismo tiempo de ráfaga, se elige al que llega primero.



Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	10
P2	0	3
P3	0	30
P4	0	4
P5	0	3

Algoritmos de planificación: Shortest Job First (SJF)

Se ordenan los procesos según su tiempo de ráfaga, **se elige al proceso con la ráfaga de CPU más corta**. Si tienen el mismo tiempo de ráfaga, se elige al que llegó primero.

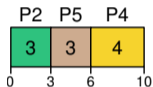


Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	10
P2	0	3
P3	0	30
P4	0	4
P5	0	3

Algoritmos de planificación: Shortest Job First (SJF)

Se ordenan los procesos según su tiempo de ráfaga, **se elige al proceso con la ráfaga de CPU más corta**. Si tienen el mismo tiempo de ráfaga, se elige al que llega primero.

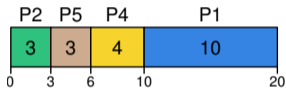


Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	10
P2	0	3
P3	0	30
P4	0	4
P5	0	3

Algoritmos de planificación: Shortest Job First (SJF)

Se ordenan los procesos según su tiempo de ráfaga, **se elige al proceso con la ráfaga de CPU más corta**. Si tienen el mismo tiempo de ráfaga, se elige al que llega primero.



Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	10
P2	0	3
P3	0	30
P4	0	4
P5	0	3

Algoritmos de planificación: Shortest Job First (SJF)

Se ordenan los procesos según su tiempo de ráfaga, **se elige al proceso con la ráfaga de CPU más corta**. Si tienen el mismo tiempo de ráfaga, se elige al que llegó primero.



$$\text{Tiempo de espera promedio} = \frac{10+0+20+6+3}{5} = \frac{39}{5} = 7,8 \text{ ms.}$$

Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	10
P2	0	3
P3	0	30
P4	0	4
P5	0	3

Algoritmos de planificación: Shortest Job First (SJF)

Se ordenan los procesos según su tiempo de ráfaga, **se elige al proceso con la ráfaga de CPU más corta**. Si tienen el mismo tiempo de ráfaga, se elige al que llegó primero.



Tiempo de espera promedio = $\frac{10+0+20+6+3}{5} = \frac{39}{5} = 7,8$ ms.

Cantidad de *task switch* = 4.

Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	10
P2	0	3
P3	0	30
P4	0	4
P5	0	3

Algoritmos de planificación: Shortest Job First (SJF)

Se ordenan los procesos según su tiempo de ráfaga, **se elige al proceso con la ráfaga de CPU más corta**. Si tienen el mismo tiempo de ráfaga, se elige al que llegó primero.



Tiempo de espera promedio = $\frac{10+0+20+6+3}{5} = \frac{39}{5} = 7,8$ ms.

Cantidad de *task switch* = 4.

Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga (ms)
P1	0	10
P2	0	3
P3	0	30
P4	0	4
P5	0	3

Este algoritmo es **óptimo** para minimiza el tiempo de espera promedio.

Sin embargo, no lo es, ya que se cuenta con una predicción del tiempo de ráfaga de cada proceso.

Algoritmos de planificación: Shortest Remaining Time First (SRTF)

Similar a JSF pero con **desalojo**.

Se elige al proceso con la ráfaga de CPU más corta y se lo ejecuta hasta que su quantum termine.

Algoritmos de planificación: Shortest Remaining Time First (SRTF)

Similar a JSF pero con **desalojo**.

Se elige al proceso con la ráfaga de CPU más corta y se lo ejecuta hasta que su quantum termine.

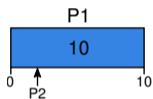
Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga	
P1	0	13	→ 13
P2	2	8	→
P3	12	4	→
P4	20	18	→
P5	30	7	→

Algoritmos de planificación: Shortest Remaining Time First (SRTF)

Similar a JSF pero con **desalojo**.

Se elige al proceso con la ráfaga de CPU más corta y se lo ejecuta hasta que su quantum termine.



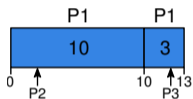
Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga	
P1	0	13	→ 13 3
P2	2	8	→ 8
P3	12	4	→
P4	20	18	→
P5	30	7	→

Algoritmos de planificación: Shortest Remaining Time First (SRTF)

Similar a JSF pero con **desalojo**.

Se elige al proceso con la ráfaga de CPU más corta y se lo ejecuta hasta que su quantum termine.



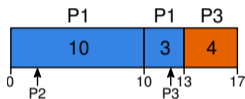
Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga	
P1	0	13	→ 13 3 0
P2	2	8	→ 8
P3	12	4	→ 4
P4	20	18	→
P5	30	7	→

Algoritmos de planificación: Shortest Remaining Time First (SRTF)

Similar a JSF pero con **desalojo**.

Se elige al proceso con la ráfaga de CPU más corta y se lo ejecuta hasta que su quantum termine.



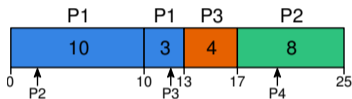
Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga	
P1	0	13	→ 13 3 0
P2	2	8	→ 8
P3	12	4	→ 4 0
P4	20	18	→
P5	30	7	→

Algoritmos de planificación: Shortest Remaining Time First (SRTF)

Similar a JSF pero con **desalojo**.

Se elige al proceso con la ráfaga de CPU más corta y se lo ejecuta hasta que su quantum termine.



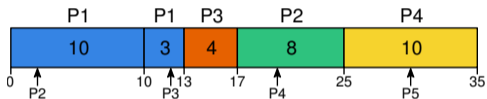
Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga	
P1	0	13	→ 13 3 0
P2	2	8	→ 8 0
P3	12	4	→ 4 0
P4	20	18	→ 18
P5	30	7	→

Algoritmos de planificación: Shortest Remaining Time First (SRTF)

Similar a JSF pero con **desalojo**.

Se elige al proceso con la ráfaga de CPU más corta y se lo ejecuta hasta que su quantum termine.



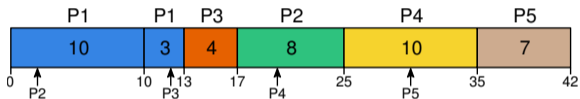
Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga	
P1	0	13	→ 13 3 0
P2	2	8	→ 8 0
P3	12	4	→ 4 0
P4	20	18	→ 18 8
P5	30	7	→ 7

Algoritmos de planificación: Shortest Remaining Time First (SRTF)

Similar a JSF pero con **desalojo**.

Se elige al proceso con la ráfaga de CPU más corta y se lo ejecuta hasta que su quantum termine.



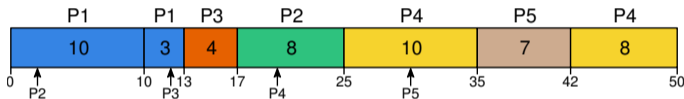
Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga	
P1	0	13	→ 13 3 0
P2	2	8	→ 8 0
P3	12	4	→ 4 0
P4	20	18	→ 18 8
P5	30	7	→ 7 0

Algoritmos de planificación: Shortest Remaining Time First (SRTF)

Similar a JSF pero con **desalojo**.

Se elige al proceso con la ráfaga de CPU más corta y se lo ejecuta hasta que su quantum termine.



Tiempo de espera promedio =

$$= \frac{0 + (17 - 2) + (13 - 12) + ((25 - 20) + (42 - 35)) + (35 - 30)}{5} = \frac{0 + 15 + 1 + (5 + 7) + 5}{5} = \frac{33}{5} = 6,6 \text{ ms.}$$

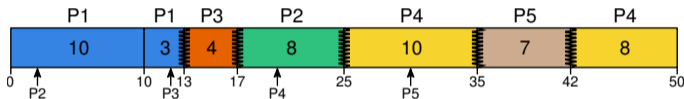
Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga	
P1	0	13	→ 13 3 0
P2	2	8	→ 8 0
P3	12	4	→ 4 0
P4	20	18	→ 18 8 0
P5	30	7	→ 7 0

Algoritmos de planificación: Shortest Remaining Time First (SRTF)

Similar a JSF pero con **desalojo**.

Se elige al proceso con la ráfaga de CPU más corta y se lo ejecuta hasta que su quantum termine.



Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga	
P1	0	13	→ 13 3 0
P2	2	8	→ 8 0
P3	12	4	→ 4 0
P4	20	18	→ 18 8 0
P5	30	7	→ 7 0

Tiempo de espera promedio =

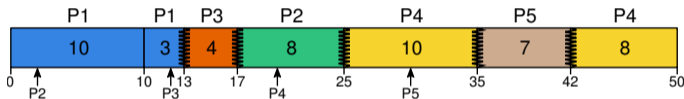
$$= \frac{0 + (17 - 2) + (13 - 12) + ((25 - 20) + (42 - 35)) + (35 - 30)}{5} = \frac{0 + 15 + 1 + (5 + 7) + 5}{5} = \frac{33}{5} = 6,6 \text{ ms.}$$

Cantidad de *task switch* = 5.

Algoritmos de planificación: Shortest Remaining Time First (SRTF)

Similar a JSF pero con **desalojo**.

Se elige al proceso con la ráfaga de CPU más corta y se lo ejecuta hasta que su quantum termine.



Ejemplo

Proceso	Tiempo de llegada	Tiempo de ráfaga	
P1	0	13	→ 13 3 0
P2	2	8	→ 8 0
P3	12	4	→ 4 0
P4	20	18	→ 18 8 0
P5	30	7	→ 7 0

Tiempo de espera promedio =

$$= \frac{0 + (17-2) + (13-12) + ((25-20) + (42-35)) + (35-30)}{5} = \frac{0+15+1+(5+7)+5}{5} = \frac{33}{5} = 6,6 \text{ ms.}$$

Cantidad de *task switch* = 5.

Este algoritmo también es conocido como *Shortest Time to Completion (STCF)*

Algoritmos de planificación: Priority Scheduling

Se define un criterio de prioridad entre procesos, **los procesos de prioridad más alta, ejecutarán antes que los procesos de prioridad más baja.**

Algoritmos de planificación: Priority Scheduling

Se define un criterio de prioridad entre procesos, **los procesos de prioridad más alta, ejecutarán antes que los procesos de prioridad más baja.**

Los criterios de prioridad pueden ser:

- **Internos:** Se toman medidas y en base a estas se asigna una prioridad. (Ej: SJF)
- **Externos:** Se asigna una prioridad en base al usuario o alguna característica del proceso.

Algoritmos de planificación: Priority Scheduling

Se define un criterio de prioridad entre procesos, **los procesos de prioridad más alta, ejecutarán antes que los procesos de prioridad más baja.**

Los criterios de prioridad pueden ser:

- **Internos:** Se toman medidas y en base a estas se asigna una prioridad. (Ej: SJF)
- **Externos:** Se asigna una prioridad en base al usuario o alguna característica del proceso.

La aplicación de prioridades se puede combinar con otros algoritmos de planificación, ya sean cooperativos como apropiativos.

Algoritmos de planificación: Priority Scheduling

Se define un criterio de prioridad entre procesos, **los procesos de prioridad más alta, ejecutarán antes que los procesos de prioridad más baja.**

Los criterios de prioridad pueden ser:

- **Internos:** Se toman medidas y en base a estas se asigna una prioridad. (Ej: SJF)
- **Externos:** Se asigna una prioridad en base al usuario o alguna característica del proceso.

La aplicación de prioridades se puede combinar con otros algoritmos de planificación, ya sean cooperativos como apropiativos.

Asignar una prioridad arbitraria a los procesos puede generar **inanición** (*starvation*).
Es decir, un proceso de baja prioridad queda esperando indefinidamente tiempo de CPU.

Algoritmos de planificación: Priority Scheduling

Se define un criterio de prioridad entre procesos, **los procesos de prioridad más alta, ejecutarán antes que los procesos de prioridad más baja.**

Los criterios de prioridad pueden ser:

- **Internos:** Se toman medidas y en base a estas se asigna una prioridad. (Ej: SJF)
- **Externos:** Se asigna una prioridad en base al usuario o alguna característica del proceso.

La aplicación de prioridades se puede combinar con otros algoritmos de planificación, ya sean cooperativos como apropiativos.

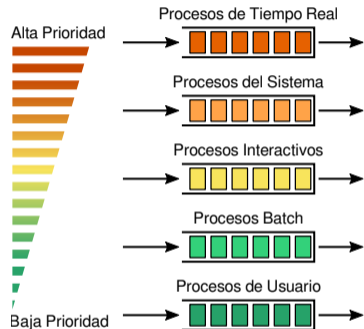
Asignar una prioridad arbitraria a los procesos puede generar **inanición** (*starvation*).
Es decir, un proceso de baja prioridad queda esperando indefinidamente tiempo de CPU.

Para evitar este problema se aplican criterios de **envejecimiento**.
A medida que aumenta el tiempo de espera, se aumenta la prioridad del proceso.

Algoritmos de planificación: Multilevel Queue Scheduling

En un planificador de colas multinivel, los procesos son **separados en distintas colas** según su tipo o prioridad.

Cada cola utilizará su **propio algoritmo de planificación** para elegir el próximo proceso a ejecutar.



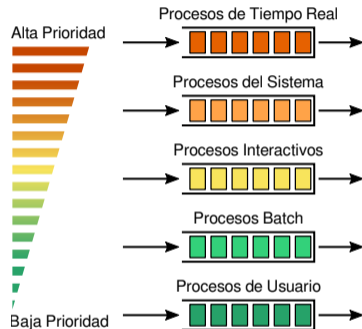
Algoritmos de planificación: Multilevel Queue Scheduling

En un planificador de colas multinivel, los procesos son **separados en distintas colas** según su tipo o prioridad.

Cada cola utilizará su **propio algoritmo de planificación** para elegir el próximo proceso a ejecutar.

Además se debe elegir la cola sobre la cual tomar procesos.
Este algoritmo suele ser de prioridades fijas y apropiativo.

Es decir, hasta que las colas de mayor prioridad no tengan procesos, no se puede ejecutar un procesos de una cola de menor prioridad, y si llega un proceso de una cola de mayor prioridad este debe ser ejecutado.



Algoritmos de planificación: Multilevel Queue Scheduling

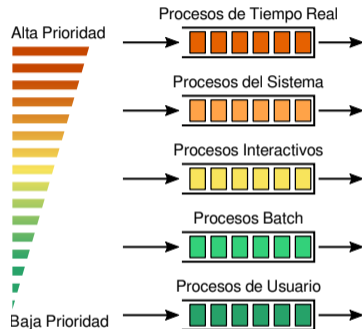
En un planificador de colas multinivel, los procesos son **separados en distintas colas** según su tipo o prioridad.

Cada cola utilizará su **propio algoritmo de planificación** para elegir el próximo proceso a ejecutar.

Además se debe elegir la cola sobre la cual tomar procesos.
Este algoritmo suele ser de prioridades fijas y apropiativo.

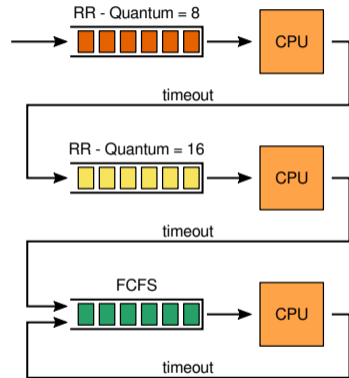
Es decir, hasta que las colas de mayor prioridad no tengan procesos, no se puede ejecutar un procesos de una cola de menor prioridad, y si llega un proceso de una cola de mayor prioridad este debe ser ejecutado.

El esquema resultante es **poco flexible** y no es libre de **inanición**.



Algoritmos de planificación: Multilevel Feedback Queue Scheduling

En este esquema los procesos se pueden **mover** entre las distintas colas de prioridades.

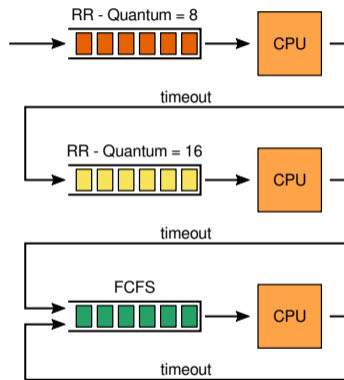


Algoritmos de planificación: Multilevel Feedback Queue Scheduling

En este esquema los procesos se pueden **mover** entre las distintas colas de prioridades.

Existirá una relación entre la prioridad de un proceso y el tiempo que tiene asignado para su *quantum*.

Procesos de mayor prioridad tendrán menor *quantum*, mientras que procesos de menor prioridad tendrán un *quantum* mayor.



Procesos de **ráfagas cortas** → en colas de **mayor prioridad**.
Procesos de **ráfagas largas** → en colas de **menor prioridad**.
Procesos de **ráfagas irregulares** → se moverán entre las colas.

Algoritmos de planificación: Multilevel Feedback Queue Scheduling

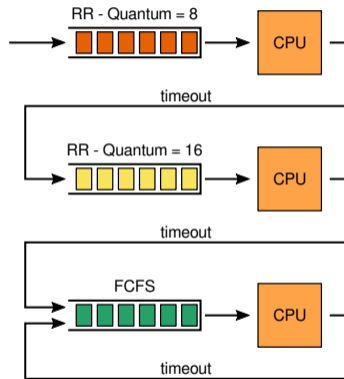
En este esquema los procesos se pueden **mover** entre las distintas colas de prioridades.

Existirá una relación entre la prioridad de un proceso y el tiempo que tiene asignado para su *quantum*.

Procesos de mayor prioridad tendrán menor *quantum*, mientras que procesos de menor prioridad tendrán un *quantum* mayor.

Cuando un proceso consume todo su *quantum*, se lo pasa a una cola de menor prioridad, es decir mayor *quantum*.

Podemos generar **inanición**, ya que siempre se ejecutan los procesos de mayor prioridad. Para evitar este problema los procesos que pasen mucho tiempo en las colas de menor prioridad, se los mueve a colas de mayor prioridad.



Procesos de **ráfagas cortas** → en colas de **mayor prioridad**.
Procesos de **ráfagas largas** → en colas de **menor prioridad**.
Procesos de **ráfagas irregulares** → se moverán entre las colas.

Algoritmos de planificación: Multilevel Feedback Queue Scheduling

En este esquema los procesos se pueden **mover** entre las distintas colas de prioridades.

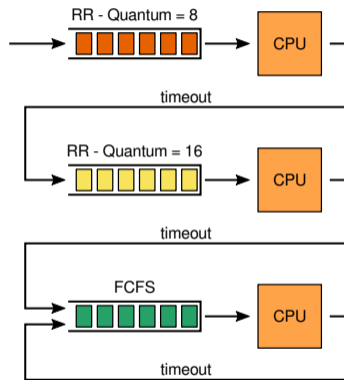
Existirá una relación entre la prioridad de un proceso y el tiempo que tiene asignado para su *quantum*.

Procesos de mayor prioridad tendrán menor *quantum*, mientras que procesos de menor prioridad tendrán un *quantum* mayor.

Cuando un proceso consume todo su *quantum*, se lo pasa a una cola de menor prioridad, es decir mayor *quantum*.

Podemos generar **inanición**, ya que siempre se ejecutan los procesos de mayor prioridad. Para evitar este problema los procesos que pasen mucho tiempo en las colas de menor prioridad, se los mueve a colas de mayor prioridad.

El esquema resulta más flexible y resuelve los problemas de las colas multinivel.



Procesos de **ráfagas cortas** → en colas de **mayor prioridad**.
Procesos de **ráfagas largas** → en colas de **menor prioridad**.
Procesos de **ráfagas irregulares** → se moverán entre las colas.

Algoritmos de Planificación

Para diseñar algoritmos de planificación se utilizan **modelos de *workloads*** de uso de sistemas.

Algoritmos de Planificación

Para diseñar algoritmos de planificación se utilizan **modelos de *workloads*** de uso de sistemas.

Estos modelos se basan en **datos reales** y buscan reflejar la complejidad de todo un conjunto de procesos ejecutando en un sistema.

Algoritmos de Planificación

Para diseñar algoritmos de planificación se utilizan **modelos de *workloads*** de uso de sistemas.

Estos modelos se basan en **datos reales** y buscan reflejar la complejidad de todo un conjunto de procesos ejecutando en un sistema.

En la práctica, los diseñadores de Sistemas Operativos **modifican variables de configuración** de sus *scheduler* para adaptarse a la evolución de los *workloads*.

Algoritmos de Planificación

Para diseñar algoritmos de planificación se utilizan **modelos de *workloads*** de uso de sistemas.

Estos modelos se basan en **datos reales** y buscan reflejar la complejidad de todo un conjunto de procesos ejecutando en un sistema.

En la práctica, los diseñadores de Sistemas Operativos **modifican variables de configuración** de sus *scheduler* para adaptarse a la evolución de los *workloads*.

Los *schedulers* de los Sistemas Operativos en general no administran procesos, sino *threads*.

Algoritmos de Planificación

Para diseñar algoritmos de planificación se utilizan **modelos de workloads** de uso de sistemas.

Estos modelos se basan en **datos reales** y buscan reflejar la complejidad de todo un conjunto de procesos ejecutando en un sistema.

En la práctica, los diseñadores de Sistemas Operativos **modifican variables de configuración** de sus *scheduler* para adaptarse a la evolución de los *workloads*.

Los schedulers de los Sistemas Operativos en general no administran procesos, sino threads.

Además tienen en cuenta en su planificación: el uso de la **memoria** (*memory stall*), el **balanceo de la carga** de los procesadores (*overloaded*), la **afinidad** con un determinado procesador (*cache*), y el **tipo** de procesador (*big.little*) entre otras características.

Bibliografía

- Silberschatz, “Fundamentos de Sistemas Operativos”, 7ma Edición, 2006.
 - **Capítulo 5 - Planificación de la CPU**, páginas 137-151 y 161-165
- Tanenbaum, “Modern Operating Systems”, 4th Edition, 2015.
 - **Chapter 2 - Processes and Threads**
 - 2.4 Scheduling - Páginas 149-165

Ejercicios

Con lo visto, ya pueden resolver todos los ejercicios de la Guía de Planificación de procesos.

¡Gracias!

Recuerden leer los comentarios adjuntos
en cada clase por aclaraciones.