5 ITINERACIÓN DE LA CPU

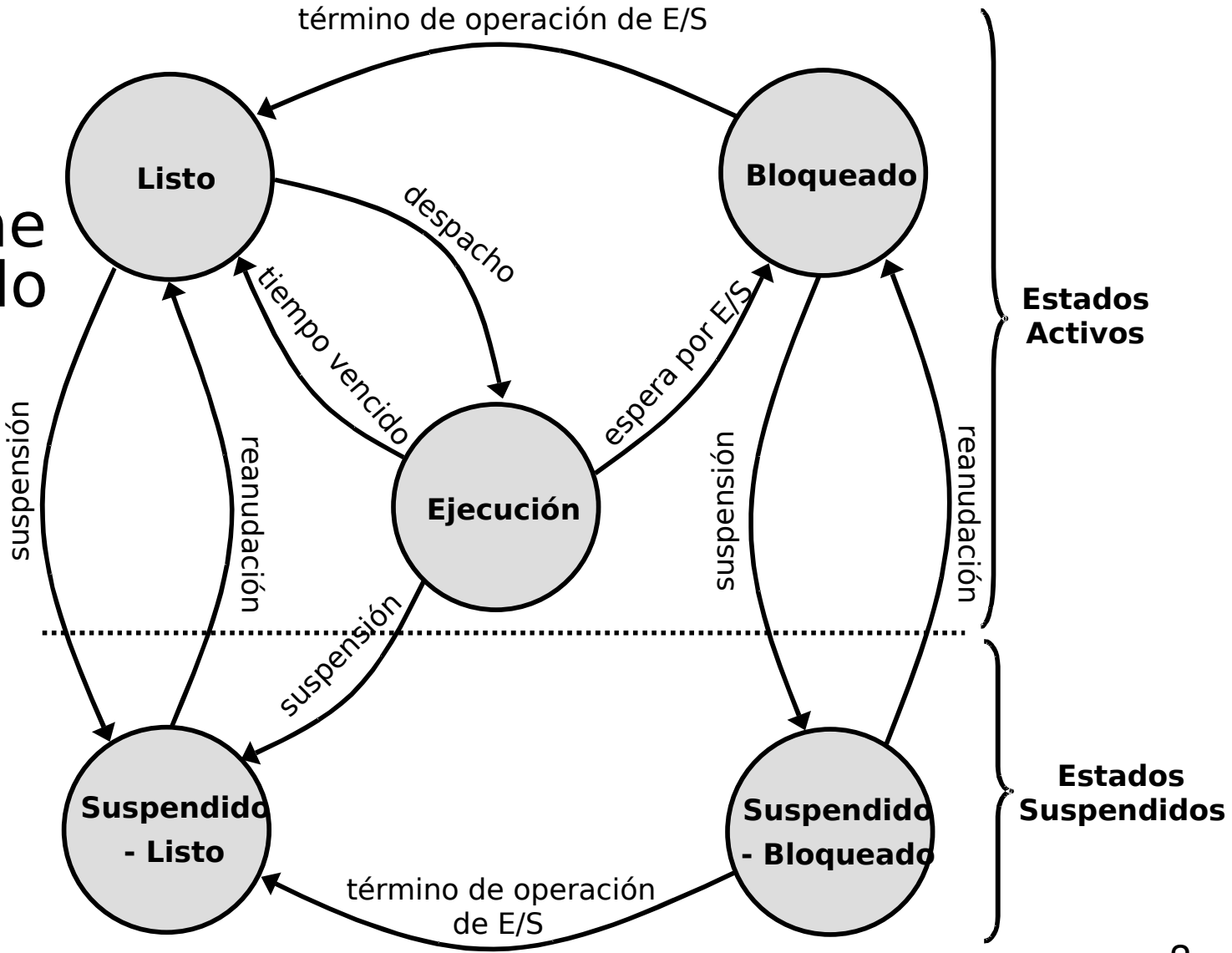
1 CONCEPTOS BÁSICOS
2 CRITERIOS DE ITINERACIÓN
3 ALGORITMOS DE ITINERACIÓN
4 ITINERACIÓN DE MÚLTIPLES PROCESADORES
5 ITINERACIÓN EN TIEMPO REAL
6 EVALUACIÓN DE ALGORITMOS

5.1 CONCEPTOS BÁSICOS

1 Ciclo de ráfagas de CPU y E/S
2 Itinerador de la CPU (scheduler)
3 Itineración Expropiativa
4 Despachador (dispatcher)

5.1 CONCEPTOS BÁSICOS

Transiciones de estado de procesos

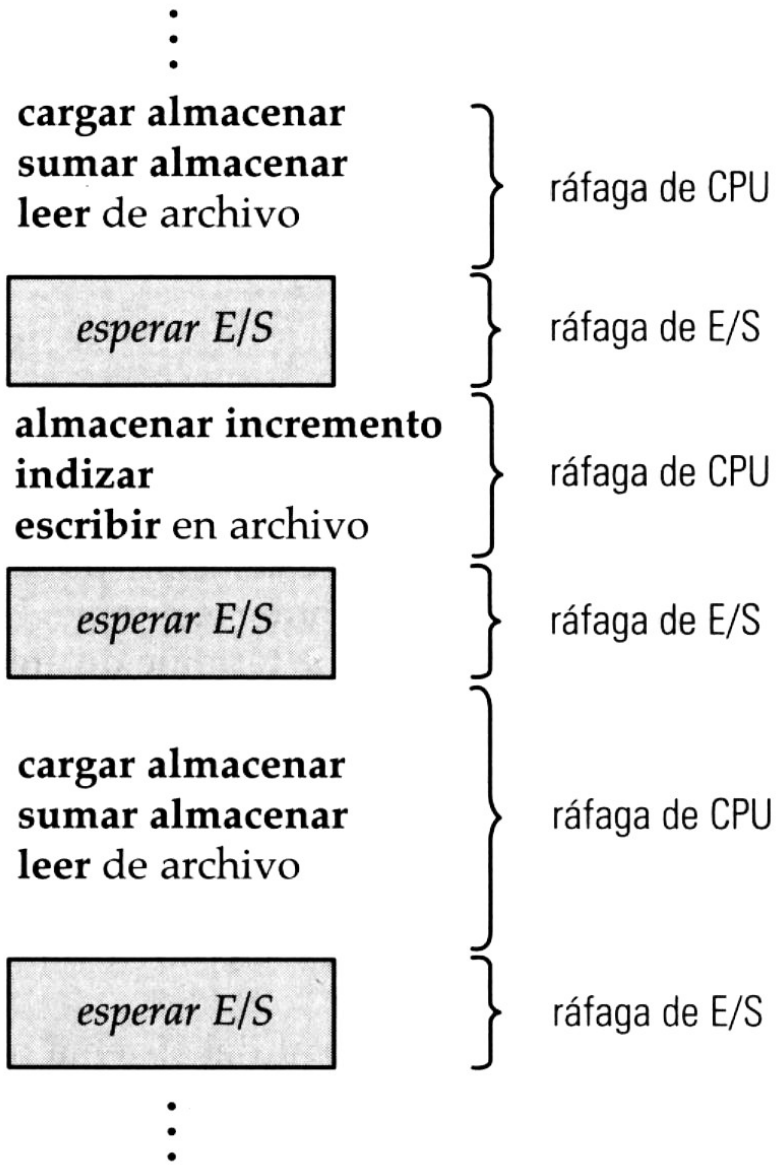


5.1.1 Ciclo de ráfagas de CPU y E/S

- Utilización máxima de la CPU, obtenida mediante la multiprogramación.
- Ciclo de ráfagas CPU-E/S; La ejecución de procesos consiste en un ciclo (alternancia) de ejecución en la CPU y espera por E/S.

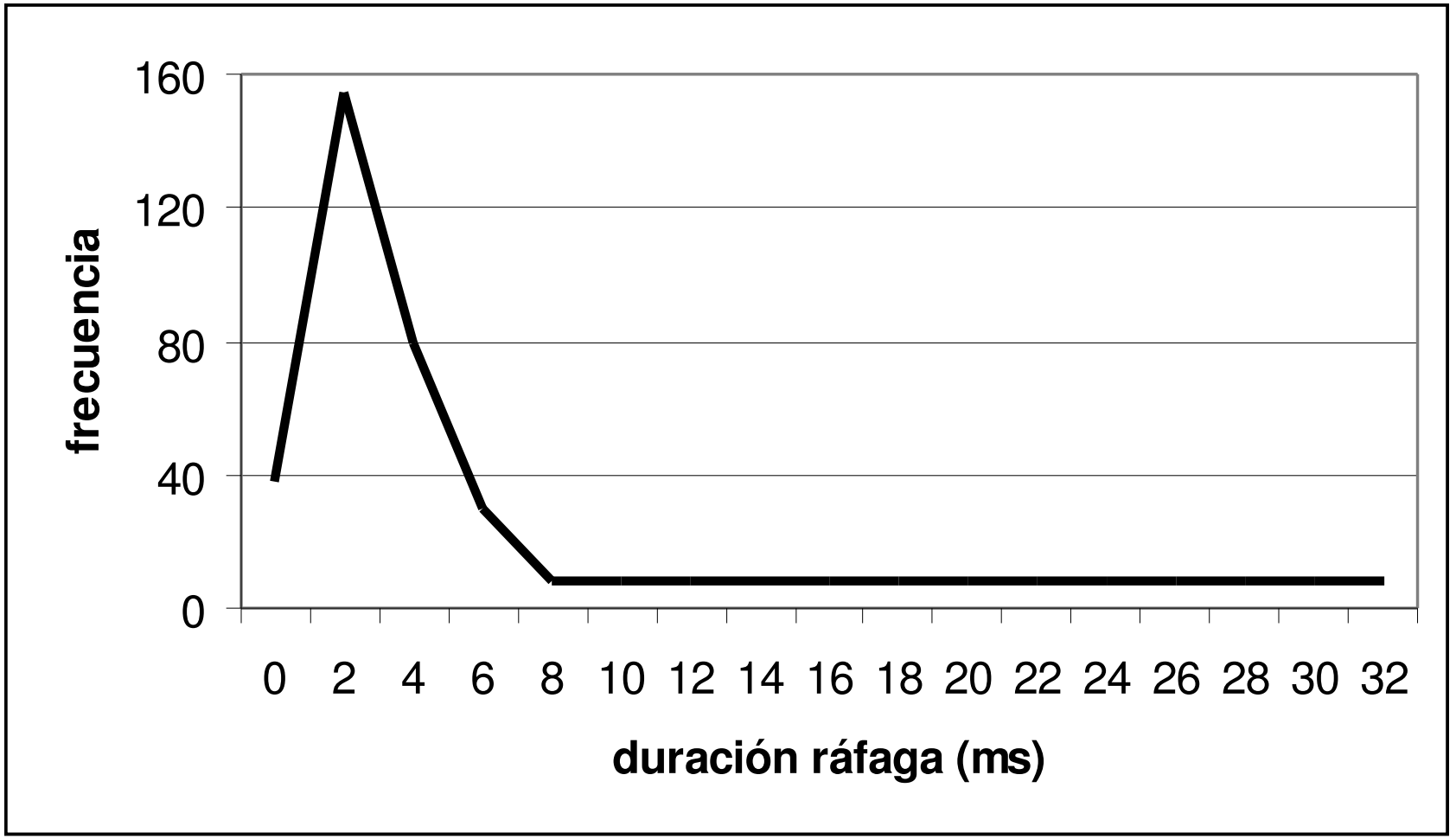
5.1.1 Ciclo de ráfagas de CPU y E/S

Secuencia alternada de ráfagas de CPU y E/S



5.1.1 Ciclo de ráfagas de CPU y E/S

Histograma de duración de ráfagas de CPU



5.1.2 Itinerador de la CPU

El Itinerador de la CPU (o itinerador a corto plazo) selecciona de entre los procesos en memoria que están listos para ejecutarse (en cola ready) y le asigna la CPU a alguno de ellos.

5.1.3 Itineración Apropiativa

- Las decisiones de itineración de la CPU pueden ocurrir cuando un proceso:
  1 Se cambia de estado running a waiting.
  2 Se cambia de estado running a ready.
  3 Se cambia de estado waiting a ready.
  4 Termina.
- Las itineraciones 1 y 4 son no apropiativas o no expropiativas
- Las itineraciones 2 y 3 son apropiativas o expropiativas

5.1.4 Despachador (dispatcher)

- Este módulo le traspasa el control de la CPU al proceso seleccionado por el itinerador de corto plazo; esto involucra:
  - Cambio de contexto
  - Cambiar a modo de usuario
  - "Saltar" a la ubicación apropiada en el programa de usuario para recomenzar dicho programa
- Latencia de Despacho: es el tiempo que demora el despachador en detener un proceso y recomenzar el otro; es *overhead*.

5.2 CRITERIOS DE ITINERACIÓN

- **Utilización de CPU**: mantenerla lo más ocupada posible.
- **Throughput (Rendimiento)**: número de procesos que completan su ejecución por unidad de tiempo.
- **Tiempo de Turnaround (retorno)**: cantidad de tiempo en ejecutar un proceso en particular. Considera todos los tiempos.
- **Tiempo de Espera**: cantidad de tiempo que un proceso ha estado esperando en la cola ready.
- **Tiempo de Respuesta**: cantidad de tiempo que transcurre desde que un requerimiento es submitido hasta que la primera respuesta es producida (sólo para time–sharing).

5.2 CRITERIOS DE ITINERACIÓN

Optimización de los Criterios

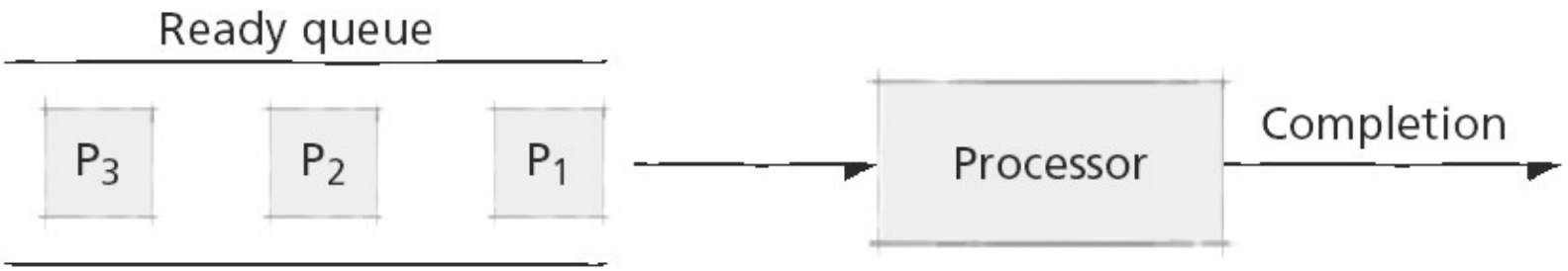
- Máxima utilización de la CPU.
- Máximo Throughput.
- Mínimo tiempo de Turnaround.
- Mínimo tiempo de espera.
- Mínimo tiempo de respuesta.

5.3 ALGORITMOS DE ITINERACIÓN

1 Por orden de llegada (FCFS)
2 Por trabajo más breve (SJF)
3 Por prioridad
4 Por turno circular (RR)
5 Colas multinivel
6 Colas multinivel con realimentación

5.3.1 Por orden de llegada (FCFS)

Modelo

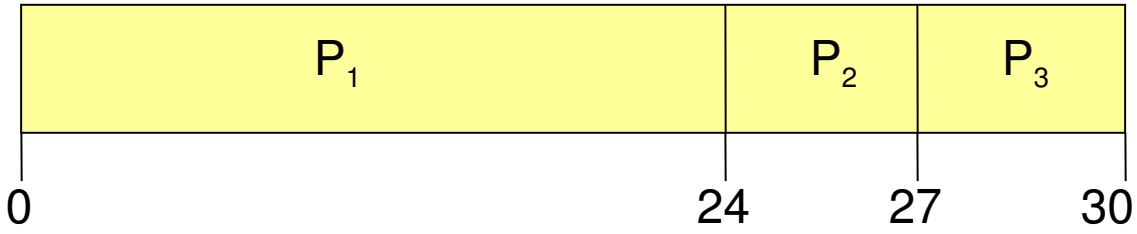


5.3.1 Por orden de llegada (FCFS)

Ejemplo:

Proceso	Duración ráfaga
P_1	24
P_2	3
P_3	3

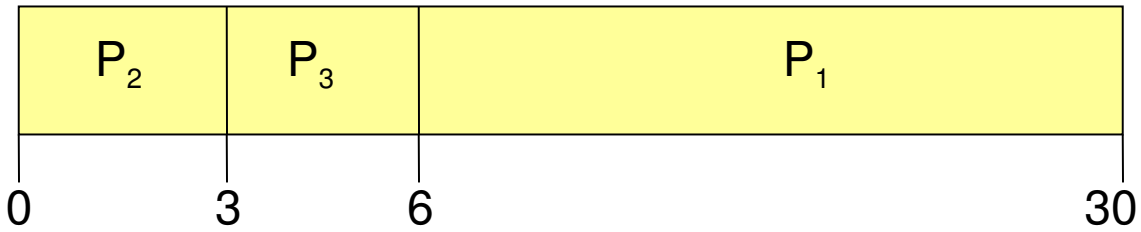
Suponer que los procesos arriban en orden: P_1, P_2, P_3. La carta Gantt para la itineración es:



- Tiempo de espera para $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
- Tiempo de espera Promedio: (0 + 24 + 27) / 3 = 17

5.3.1 Por orden de llegada (FCFS)

- Si los procesos arriban en el orden: P_2, P_3, P_1.
- La carta Gantt para la itineración es:



- Tiempo de espera para $P_1 = 6$; $P_2 = 0$; $P_3 = 3$
- Tiempo de espera Promedio : (6 + 0 + 3) / 3 = 3
- Es mucho mejor que el caso anterior.
- Efecto Convoy: el proceso más breve detrás del proceso más extenso.

5.3.2 Por Trabajo más Breve (SJF)

- Se asocia con cada proceso la duración de su próxima ráfaga de CPU, la cual es usada para itinerar, seleccionando el proceso con el tiempo más breve.
- Hay dos esquemas:
  - No Expropiativo: luego que la CPU ha sido asignada, se debe esperar que el proceso complete su ráfaga.
  - Expropiativo: si un proceso nuevo arriba con una du-ración de ráfaga de CPU menor que el tiempo res-tante del proceso actualmente en ejecución, puede ser interrumpido. Es conocido como Shortest-Remaining-Time-First (SRTF).
- SJF es óptimo: asigna con tiempo de espera pro-medio mínimo, para un conjunto de procesos.

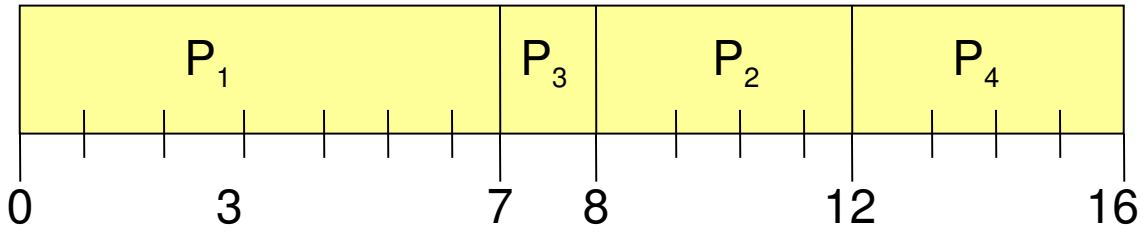
5.3.2 Por Trabajo más Breve (SJF)

Ejemplo de SJF

Proceso	Tiempo de Arribo	Duración Ráfaga
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

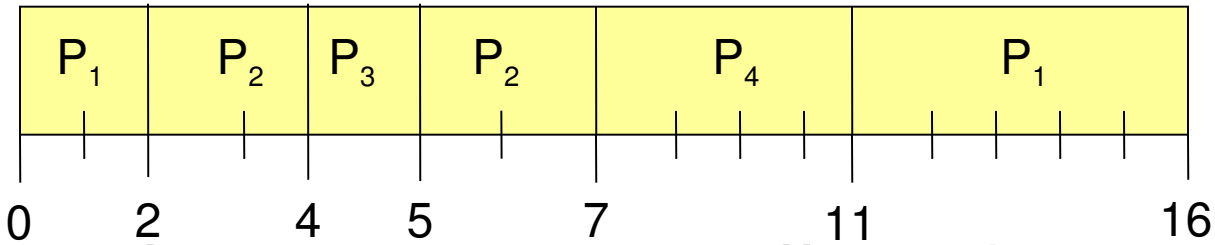
5.3.2 Por Trabajo más Breve (SJF)

SJF no expropiativo:



Tiempo de Espera Promedio = (0+6+3+7)/4 = 4

SJF expropiativo:



Tiempo de Espera Promedio = (9+1+0+2)/4 = 3

5.3.2 Por Trabajo más Breve (SJF)

Determinación de la duración de la próxima ráfaga de CPU

- Sólo se puede estimar la duración.
- Puede ser hecho utilizando la duración de las ráfagas de CPU previas, usando promedio exponencial.

  1 t_n = duración real de la ráfaga n **ésina**

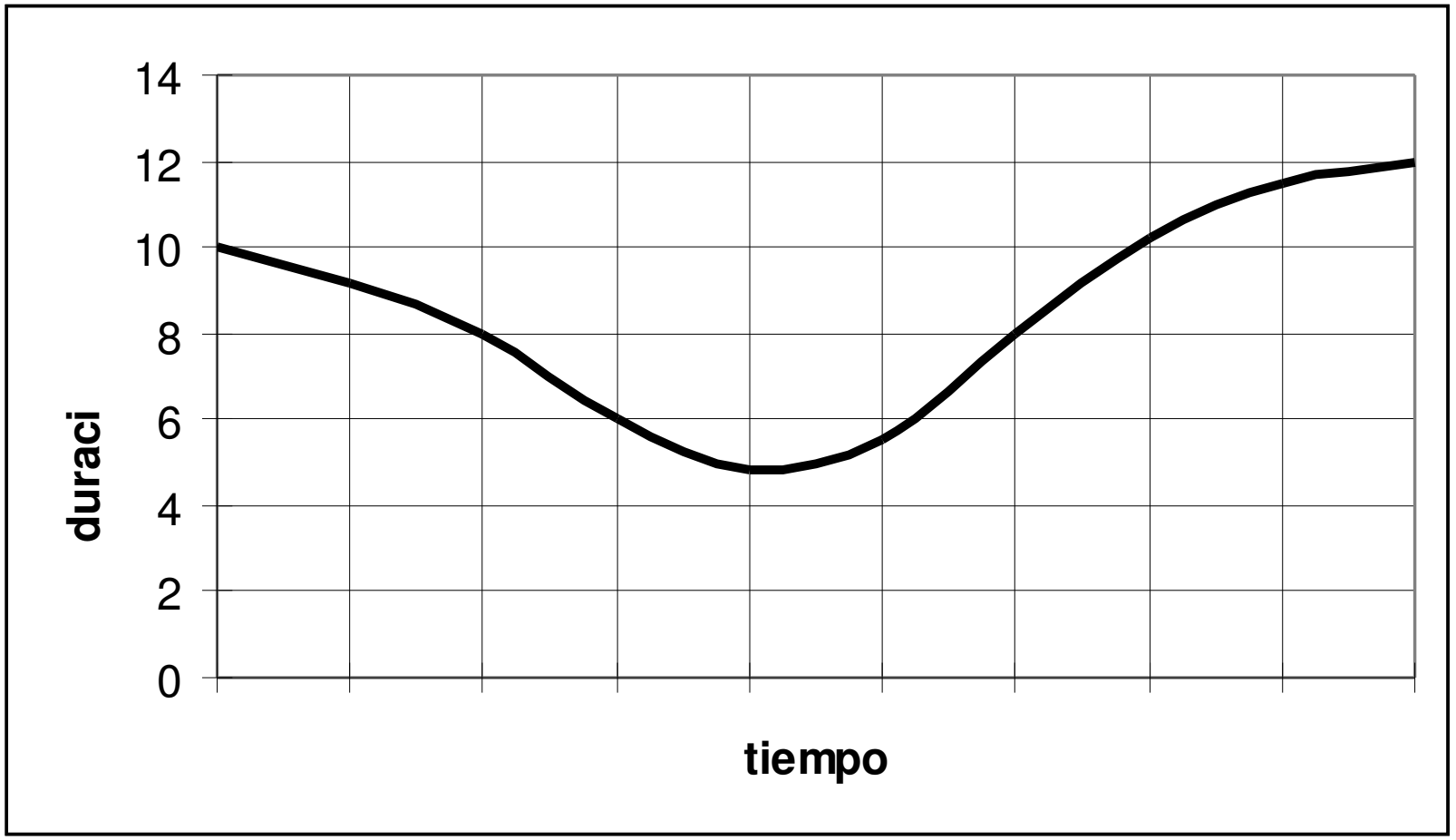
  2 τ_{n+1} = valor estimado siguiente ráfaga

  3 $\alpha : 0 \leq \alpha \leq 1$.

  4 Se define : $\tau_{n+1} = \alpha\ t_n + (1 - \alpha)\ t_n$

5.3.2 Por Trabajo más Breve (SJF)

Determinación de la duración de la próxima ráfaga de CPU (cont)

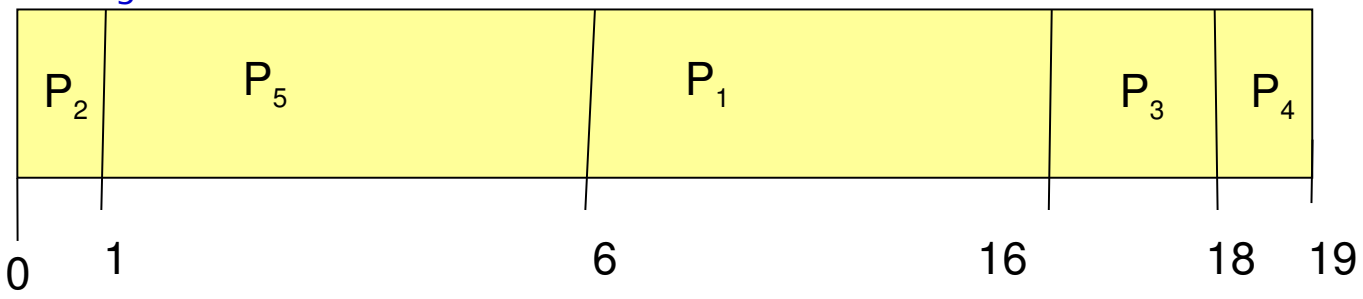


5.3.3 Itineración por Prioridad

- Un número de prioridad (entero) es asociado a cada proceso.
- Se asigna la CPU al proceso con mayor prioridad (número menor = prioridad mayor). Puede ser:
  - Expropiativa (llega un proceso con prioridad mayor)
  - No Expropiativa
- Procesos con igual prioridad se atienden en orden FCFS
- SJF es una itineración basada en prioridad, donde la prioridad es el tiempo estimado para la siguiente ráfaga de CPU.

5.3.3 Itineración por Prioridad

Proceso	Prioridad	Duración Ráfaga
P_1	3	10
P_2	1	1
P_3	4	2
P_4	5	1
P_5	2	5



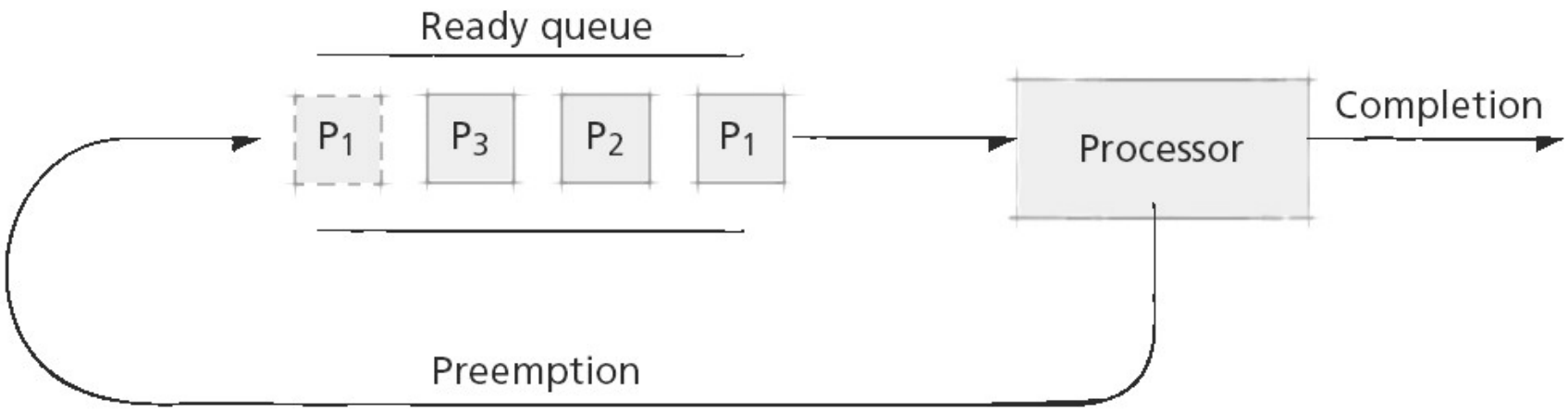
Tiempo de espera promedio
= (6+0+16+18+1)/5=8.2 ms

5.3.3 Itineración por Prioridad

- Problema: bloqueo indefinido o hambruna (starvation); los procesos de menor prioridaad puede que nunca se ejecuten.
- Solución: envejecimiento (aging); a medida que el tiempo progresa, aumentar la prioridad del proceso.
- La prioridad puede ser Interna o Externa.

5.3.4 Turno Circular (Round Robin)

Modelo



5.3.4 Turno Circular (Round Robin)

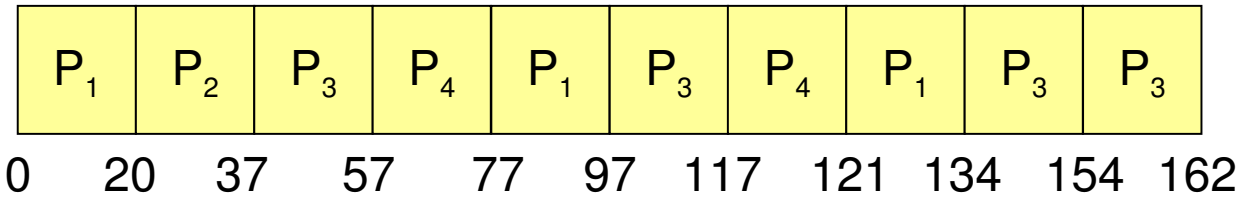
- Cada proceso obtiene un pequeño lapso de CPU (***q***), usualmente de 10 a 100 ms.
- Luego de transcurrido este lapso, el proceso es interrumpido y puesto al final de la cola ready.
- Si hay n procesos en la cola, cada proceso ob-tiene $^{1}/_{n}$ del tiempo de CPU de duración máxima q. Luego, ninguno esperará más de (n-1)q.
- Rendimiento:
  - Si q es muy grande $\Rightarrow$ FIFO
  - Si q es muy pequeño $\Rightarrow$ debe ser suficientemente grande respecto de la duración del cambio de contexto, de otra

5.3.4 Turno Circular (Round Robin)

Ejemplo : RR con q=20

Proceso	Duración ráfaga
P_1	53 = 20 + 20 + 13
P_2	17= 17
P_3	68 = 20 + 20 + 20 + 8
P_4	24 = 20 + 4

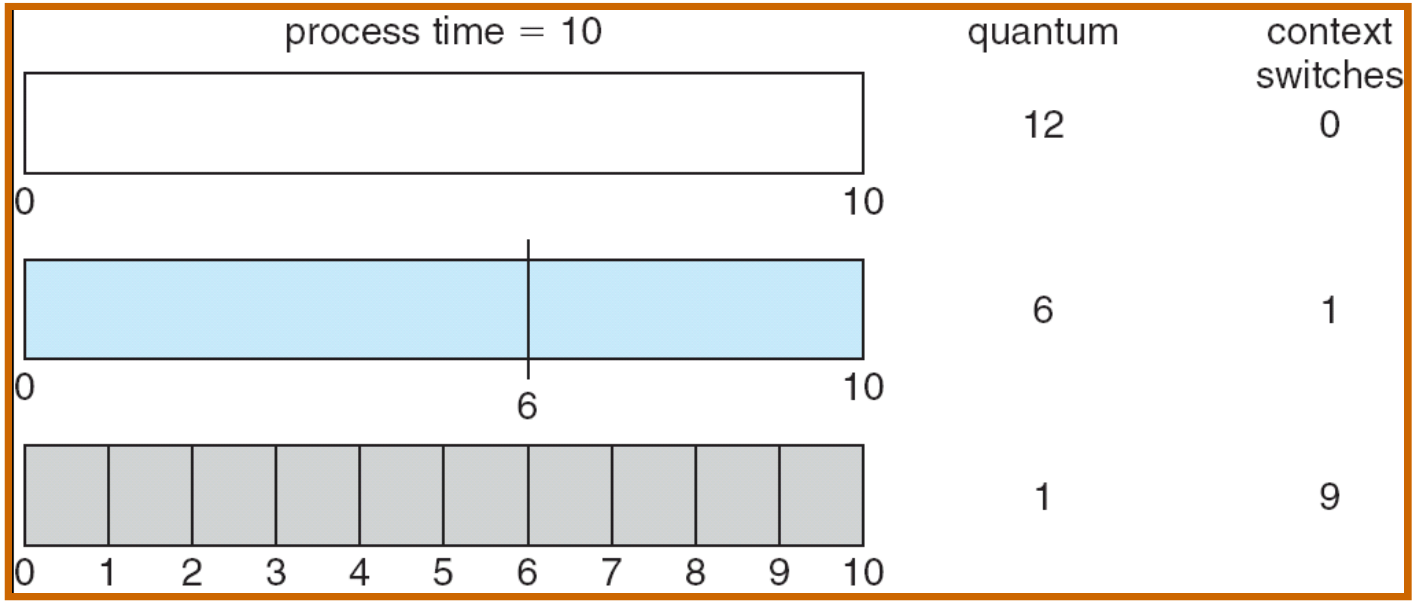
Carta Gantt:



Típicamente, tiene un tiempo promedio de turnaround mayor que SRTF, pero un mejor tiempo de respuesta.

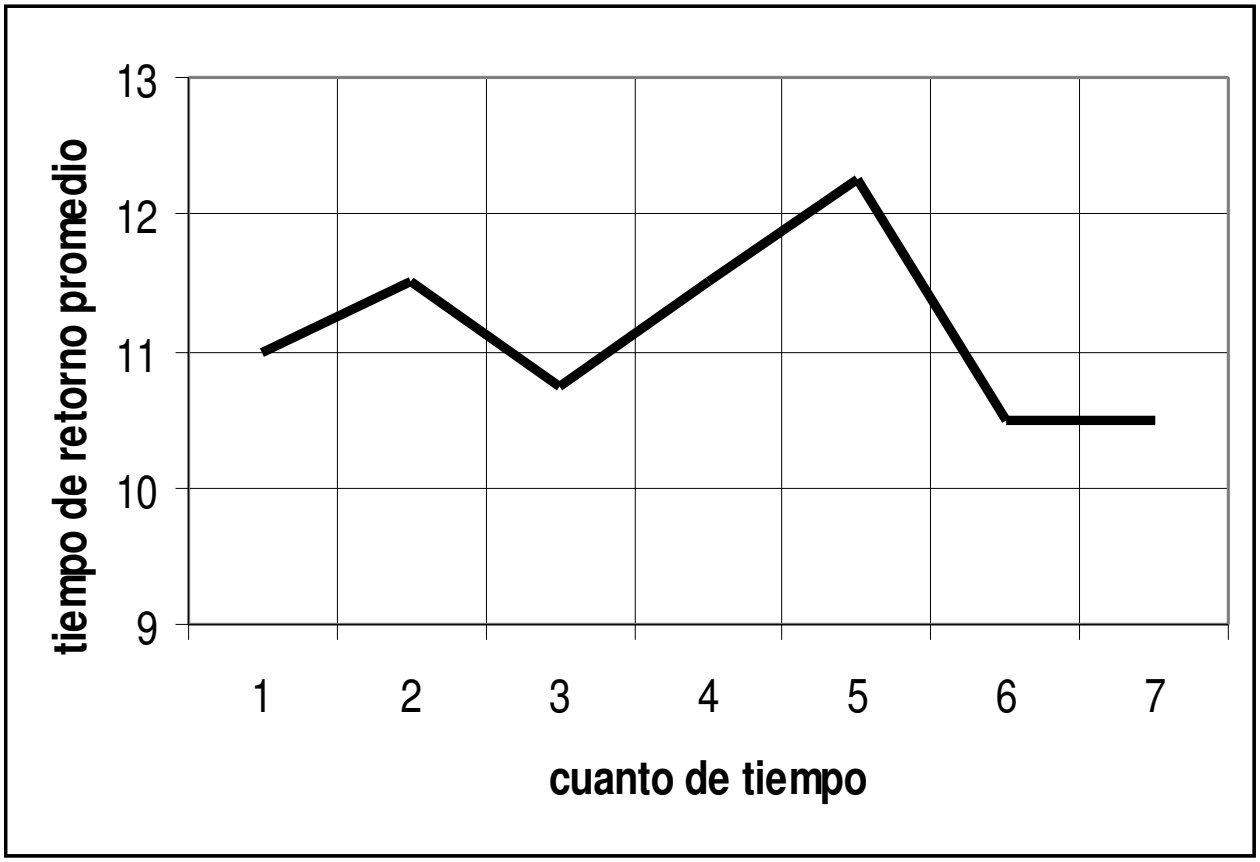
5.3.4 Turno Circular (Round Robin)

ómo q más pequeño aumenta el Cambio de Context



5.3.4 Turno Circular (Round Robin)

Variación del Tiempo de Turnaround respecto de q



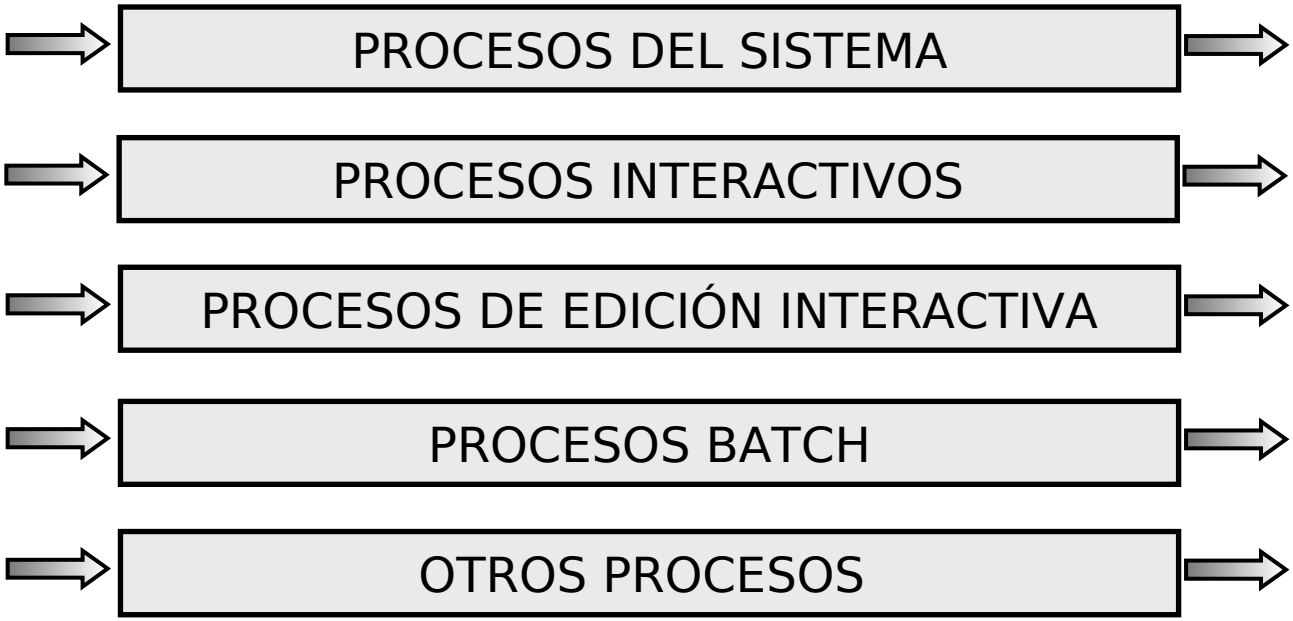
Proceso	Tiempo
P1	6
P2	3
P3	1
P4	7

5.3.5 Colas Multinivel

- La cola ready es particionada en colas separadas:
  - Foreground (interactiva)
  - Background (batch)
- Cada cola tiene su propio algoritmo de itineración
  - Foreground: RR
  - Background: FCFS
- Debe hacerse itineración entre las colas.
  - Itineración de prioridad fija; es decir, servir a todos los procesos de la cola foreground y luego a los de la cola background. Esto puede generar hambruna.
  - Tajada de tiempo: cada cola obtiene una

5.3.5 Colas Multinivel

Mayor Prioridad



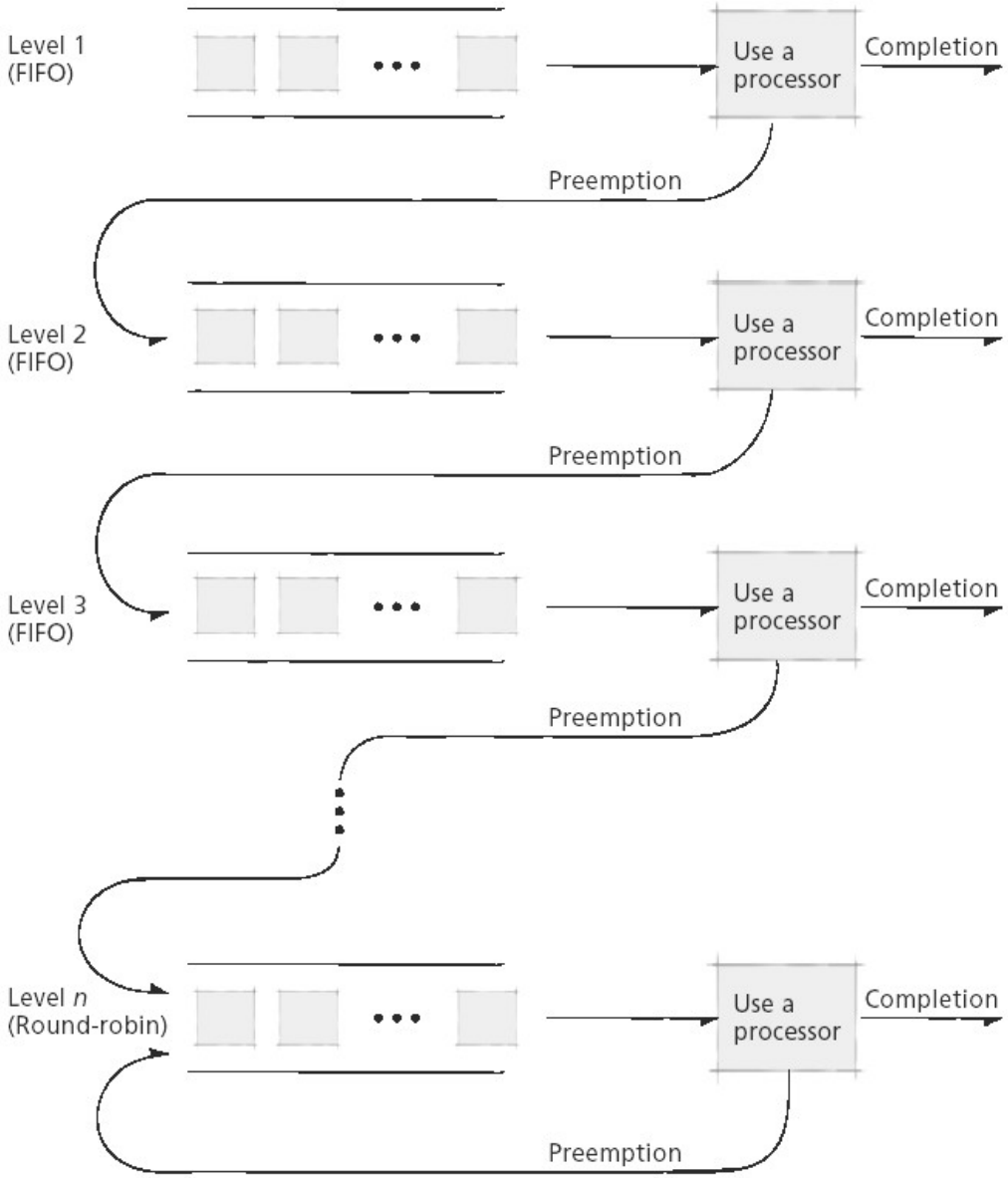
Menor Prioridad

5.3.6 Colas Multinivel con Realimentación

- Un proceso puede moverse entre varias colas; de esta manera, se puede implementar envejecimiento.
- Un itinerador de colas multinivel con realimentación puede ser definido por los siguientes parámetros :
  - Número de colas
  - Algoritmo de itineración para cada cola
  - Método usado para determinar cuándo cambiar un proceso de cola (hacia arriba o hacia abajo)
  - Método usado para determinar a cuál cola puede entrar un proceso cuando necesite algún servicio.

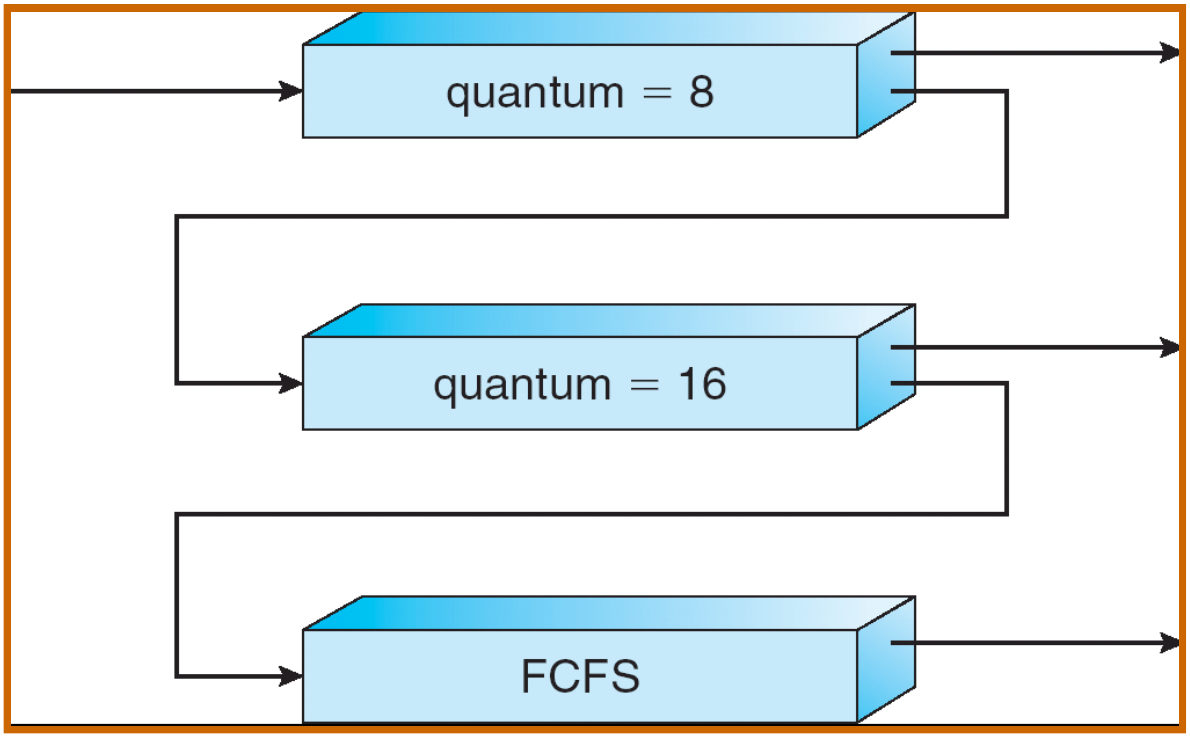
5.3.6 Colas Multinivel con Realimentación

Modelo



5.3.6 Colas Multinivel con Realimentación

Ejemplo



5.3.6 Colas Multinivel con Realimentación

Ejemplo

- Tres colas:
  - Q_0: RR con time quantum de 8 milisegundos
  - Q_1: RR con time quantum de 16 milisegundos
  - Q_2: FCFS
- Itineración:
  - Un nuevo job entra en la cola Q_0, donde es atendido en modalidad FCFS. Cuando obtiene la CPU, recibe 8 ms. Si no termina en 8 ms, es movido a la cola Q_1.
  - En Q_1, donde es nuevamente atendido en modalidad FCFS, obtiene 16 ms adicionales. Si aún no termina, es interrumpido y movido a la cola Q_2.

5.4 ITINERACIÓN DE MÚLTIPLES PROC.

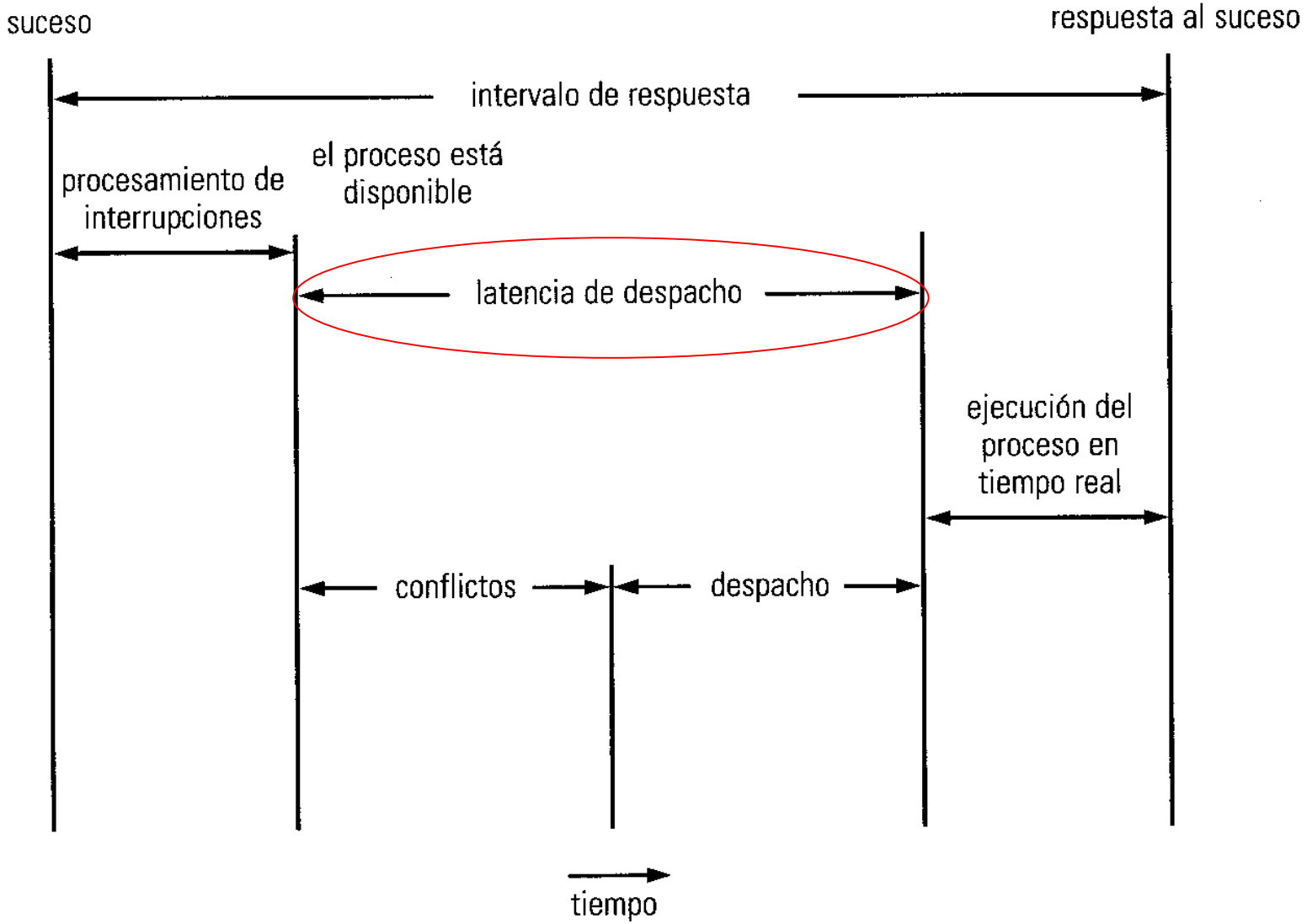
- La itineración de la CPU es más compleja cuando hay múltiples CPUs disponibles.
- En multiprocesamiento simétrico, con procesadores homogéneos, se puede tener compartimiento de la carga.
- En multiprocesamiento asimétrico, sólo un procesador accesa las estructuras de datos del sistema, aliviando la necesidad de compartir datos. Es más simple que el simétrico.

5.5 ITINERACIÓN EN TIEMPO REAL

- Sistemas "***hard***": requeridos para completar una tarea crítica dentro de una cantidad de tiempo restringida.
- Sistemas "***soft***": requieren que los procesos críticos tengan mayor prioridad, sobre los demás. Son menos restrictivos que los hard.

5.5 ITINERACIÓN EN TIEMPO REAL

Latencia de Despacho



5.6 EVALUACIÓN DE ALGORITMOS

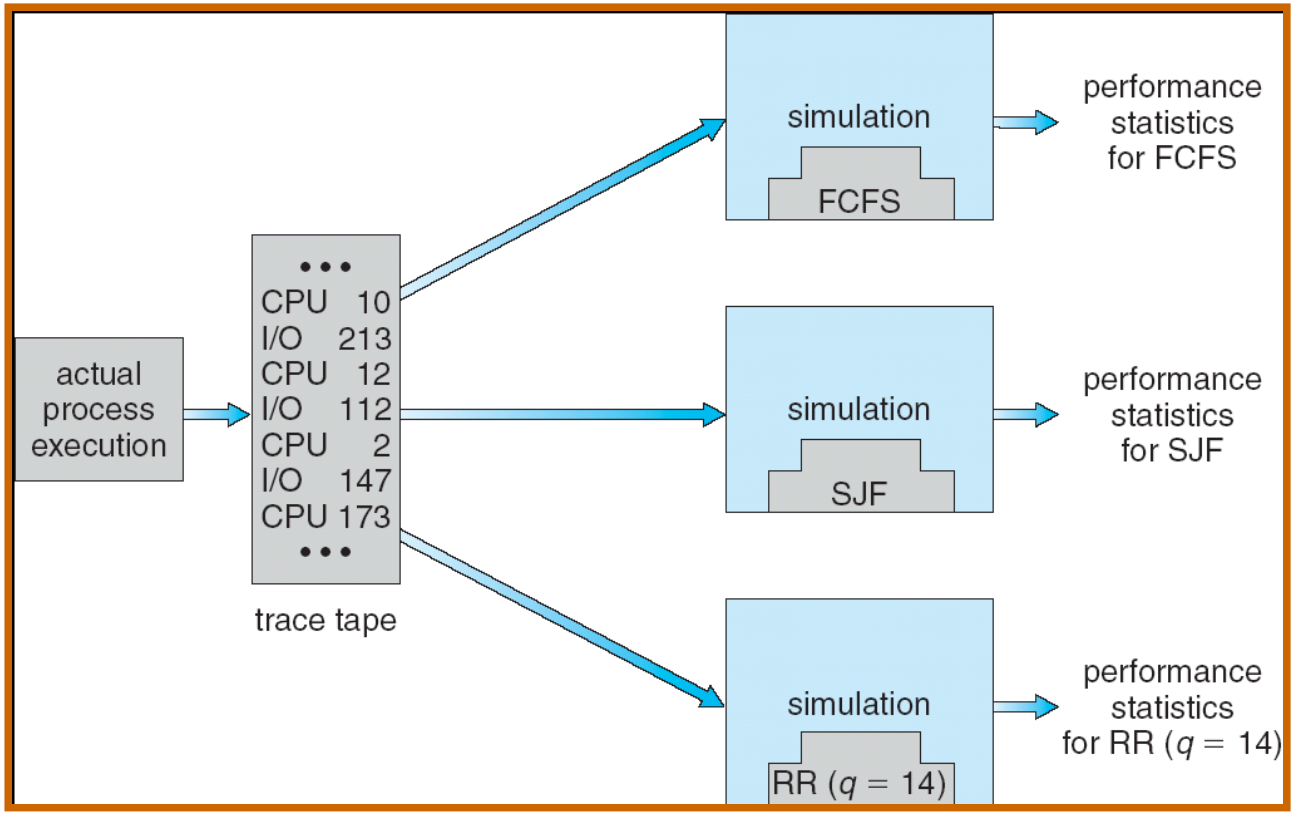
- Definir los criterios de evaluación.
- Definir la importancia relativa de los criterios. Por ejemplo:
  - Maximizar la utilización de CPU, bajo la restricción de que el tiempo de respuesta sea, a lo más, de 1 seg.
  - Maximizar el throughput, de manera que el tiempo promedio de turnaround sea directamente proporcional al tiempo total de ejecución.

5.6 EVALUACIÓN DE ALGORITMOS

- Una vez definidos los criterios, se evalúan los algoritmos.
- Formas :
  - Modelamiento determinístico: forma de evaluación analítica que toma una carga de trabajo particular predeterminada y define el rendimiento de cada algoritmo para aquella carga.
  - Modelos de colas de espera
  - Simulación
- Implementación

5.6 EVALUACIÓN DE ALGORITMOS

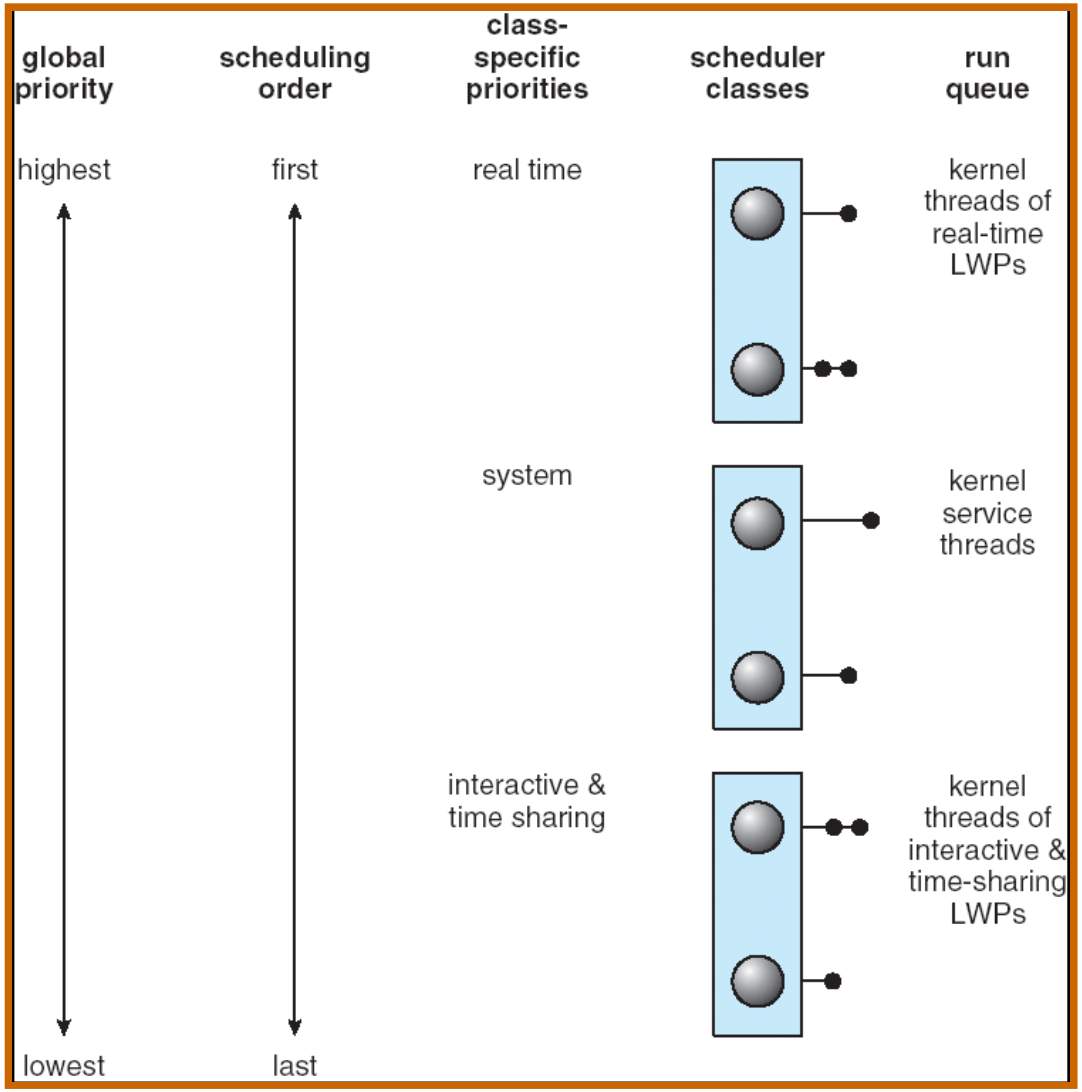
Evaluación de itineradores de CPU mediante Simulación



Ejemplos de S.O.

- Planificación en Solaris (UNIX): Basada en prioridades, definiendo 4 clases, en orden de prioridad:
  - Tiempo real
  - Sistema
  - Tiempo Compartido
  - Interactiva
- Dentro de cada clase hay diferentes prioridades y diferentes algoritmos de planificación.
- La clase por defecto: Tiempo compartido que utiliza colas multinivel realimentadas.

Solaris


global priority
scheduling order
class-specific priorities
scheduler classes
run queue
highest
first
real time
kernel threads of real-time LWPs
system
kernel service threads
interactive & time sharing
kernel threads of interactive & time-sharing LWPs
lowest
last

Solaris – 60 niveles de prioridad

Tabla de despacho de planificación de subprocesos interactivos y de tiempo compartido.

- Prioridad: Depende de la clase. N° Alto -> mayor prioridad.(Procesos interactivos)
- Quantum: Prioridad baja, más tiempo (procesos de CPU)
- Caducidad: Subproceso que consume su tiempo, se asigna nueva prioridad (se reduce)
- Retorno del estado dormido (esperando realizar una E/S): Prioridad que se asigna a un subproceso cuando el dispositivo que espera se libera y queda disponible. (prioridad aumenta por ser un proceso interactivo)

priority	time quantum	time quantum expired	return from sleep
0	200	0	50
5	200	0	50
10	160	0	51
15	160	5	51
20	120	10	52
25	120	15	52
30	80	20	53
35	80	25	54
40	40	30	55
45	40	35	56
50	40	40	58
55	40	45	58
59	20	49	59

Planificación en Windows XP

- Algoritmo de planificación expropiativo basado en prioridades.
- Siempre se ejecuta el subproceso de prioridad más alta
- Un subproceso seleccionado por el despachador se ejecutará hasta:
  - que sea desalojado por un subproceso de prioridad más alta
  - que termine
  - que su cuanto de tiempo concluya
  - Que invoque una llamada que lo bloquee (una operación de E/S)

Planificación en Windows XP

- Esquema de 32 niveles de prioridades
- Trabaja con dos clases:
  - Clase variable: prioridades 1-15
  - Clase de tiempo real: 16-31
  - Prioridad 0: subproceso asociado a la gestión de la memoria
- Colas multinivel realimentadas

Planificación en Windows XP

- Clases de prioridades: real_time, High, above_normal, normal, below_normal, idle (todas variables, menos real-time)
- Valores para las prioridades relativas: Time_Critical, Highest, Above_Normal, Normal, Below_Normal, Lowest, Idle

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

Prioridad base para cada clase de prioridad

Planificación en Windows XP

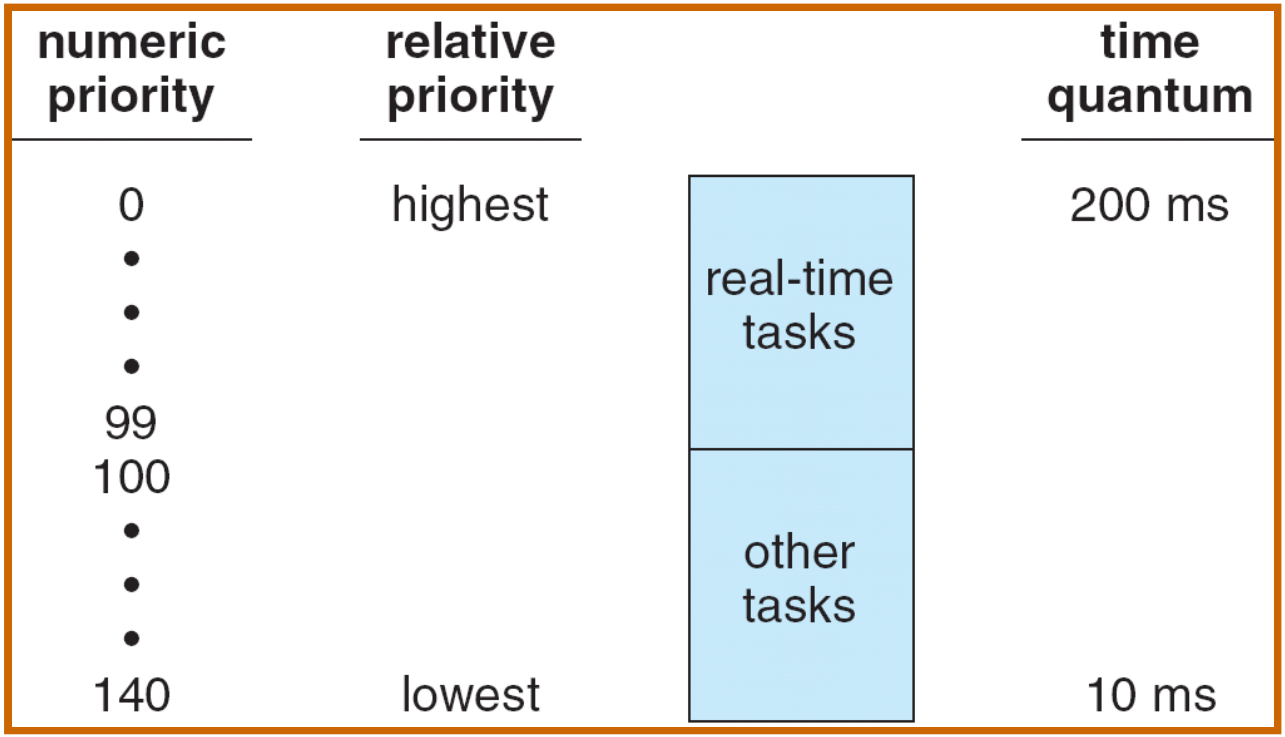
- Cuando se excede el cuanto de tiempo de un subproceso, el subproceso se desaloja y si pertenece a la clase variable, se reduce su prioridad.
- Subproceso que espera un dispositivo de E/S y uno que espera una operación de disco ¿En cuál aumenta la prioridad?
- Programa interactivo: Rendimiento bueno para estos procesos.
  - Procesos de primer plano (seleccionado en la pantalla): Se multiplica por un factor de 3 el cuanto de planificación. (tres veces más tiempo de ejecución antes que lo desalojen)
  - Procesos de segundo plano

Planificación en Linux

- Algoritmo basado en prioridades y expropiativo
- Dos rangos de prioridades:
  - Tiempo real: 0 a 99
  - Normal: 100 a 140
- Esquema de prioridades global, valores más bajos indican prioridades más altas.
  - Prioridad más alta-> cuantos de tiempos más largos
  - Prioridad más baja -> Cuantos de tiempos más cortos

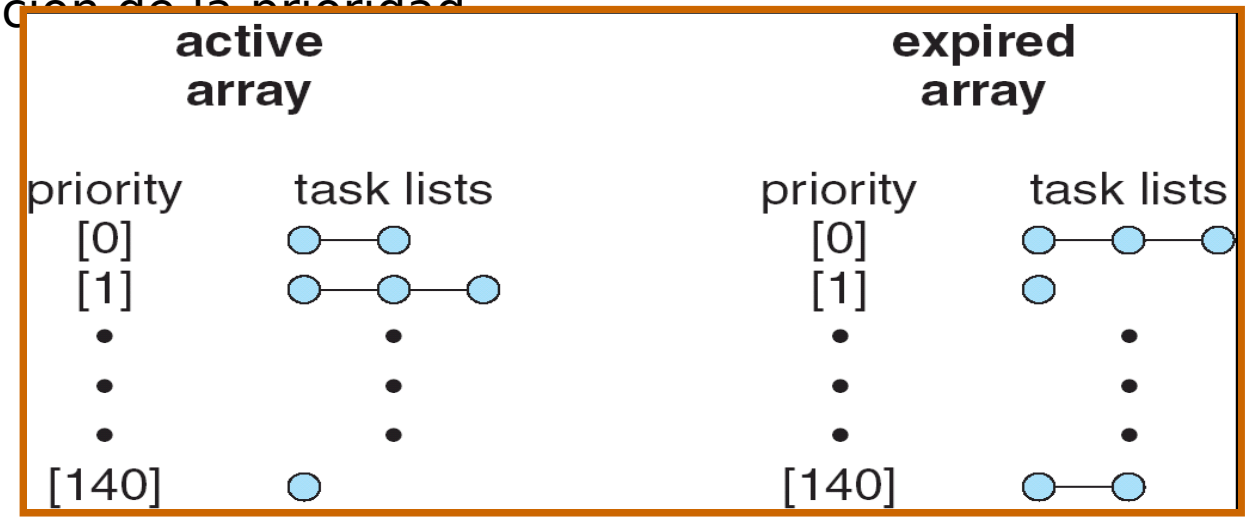
Planificación en Linux

- Relación entre la prioridad y la duración del cuanto de tiempo.



Planificación en Linux

- Una tarea ejecutable se considera elegible para ejecutarse en la CPU cuando todavía le quede tiempo de su cuanto de tiempo. (Round Robin)
- Dos matrices de prioridades:
  - Activa: Contiene todas las tareas que todavía disponen de tiempo en su cuanto de tiempo
  - Caducada: Tareas que han agotado su cuanto de tiempo
- Cada una de estas matrices tiene una lista indexada en función de la prioridad



Planificación en Linux

- Cuando la matriz activa está vacía (todas las tareas han agotado sus cuantos de tiempo), se intercambian las matrices y la caducada queda como activa. (se asignan nuevas prioridades y por ende, nuevos cuantos de tiempo).
- Las tareas más interactivas tienen, en general, una prioridad “normal”.
- Entonces las tareas que están limitadas por CPU tienen una prioridad más alta (valor de prioridad más bajo).