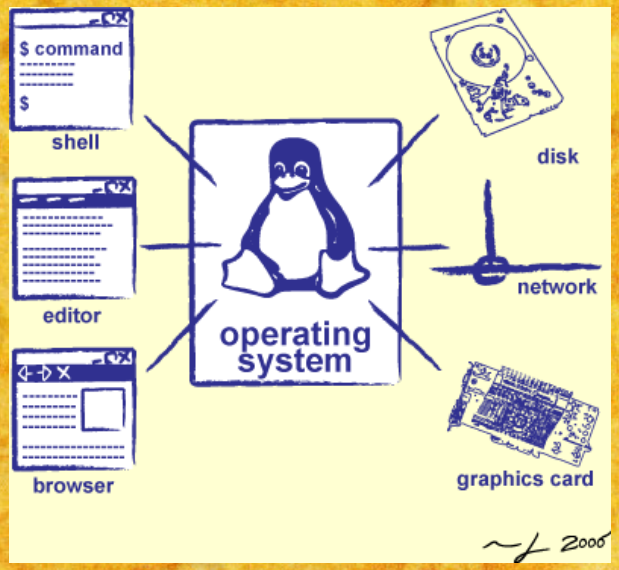
# 5.- Despacho de CPU



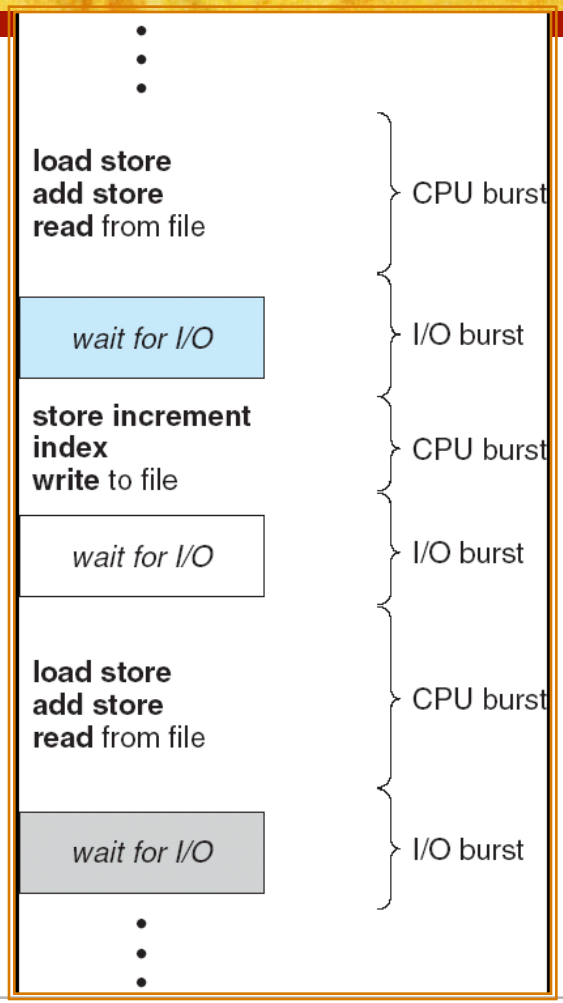
# Capítulo 5: Despacho de CPU

- Conceptos Básicos
- Criterio de Asignación
- Algoritmos de Asignación
- Despacho de Threads
- Ejemplos de SOs
- Despacho de Threads de Java
- Evaluación de Algoritmos

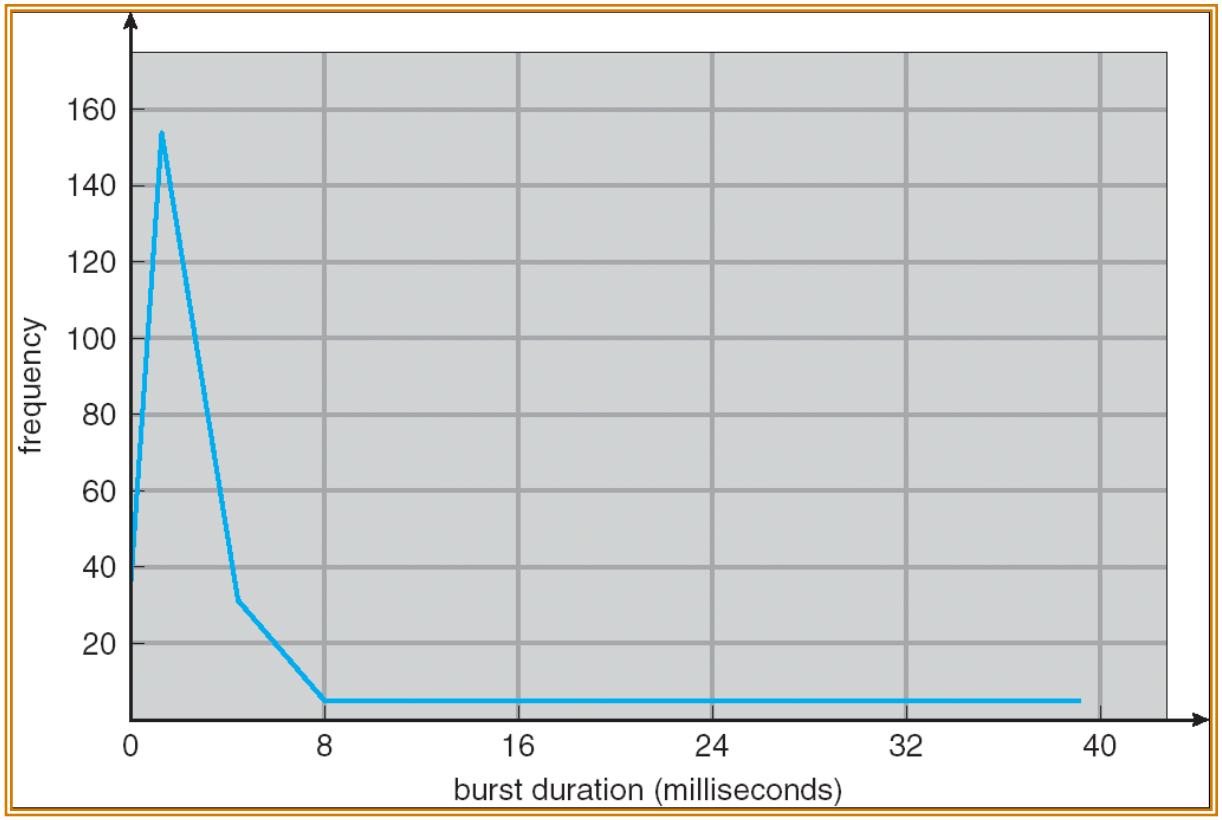
# Conceptos Básicos

- Utilización máxima de CPU con multiprogramación
- Ciclo CPU–I/O – Ejecución de un proceso = *ciclo* de CPU (ejecución) y espera de I/O
- Distribución de periodos CPU (bursts)

# Secuencia CPU - I/O



# Histograma de Tiempos de CPU



# Despachador de CPU

- Selecciona un proceso (ready) en memoria y le asigna el CPU
- Decisiones de asignación de CPU se requieren cuando un proceso:
  1. Cambia de estado running a waiting
  2. Cambia de running a ready
  3. Cambia de waiting a ready
  4. Termina
- Asignación en 1 y 4 es sin desalojo (*nonpreemptive*)
- Las otas son con desalojo (*preemptive*)

# Despachador de CPU

- El Despachador le pasa el control del CPU a los procesos seleccionados por el despachador de corto plazo; ésto involucra:
  - Context switching
  - Cambio a user mode
  - Salto a la localidad adecuada en el programa para resumir ejecución
- *Latencia de Despacho* – tiempo que le lleva al despachador en deterner un proceso y resumir el otro

# Criterio de Despacho

- Utilización de CPU – mantener el CPU ocupado
- Throughput – # de procesos que completan ejecución por unidad de tiempo
- Tiempo Turnaround – time para ejecutar un proceso en particular
- Tiempo de Espera (Waiting time) – time que un proceso ha estado esperando en la cola ready
- Tiempo de Respuesta – Tiempo entre el punto en que el proceso fue lanzado hasta que se produce la primer respuesta, **no** output (para ambientes time-sharing)

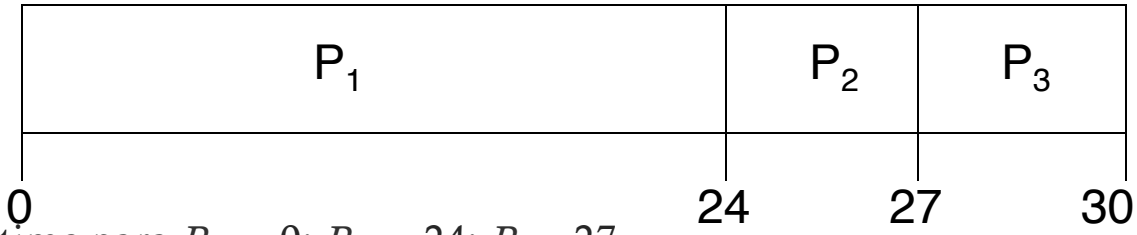
# Criterio de Optimización

- Max utilización de CPU
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

# Despacho First-Come, First-Served (FCFS)

- Suponga que los procesos llegan en orden: $P_1$ , $P_2$ , $P_3$
- La gráfica de Gantt del despacho es:

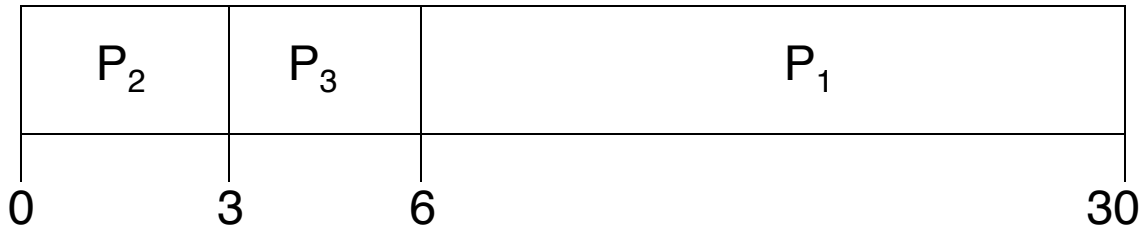
| Process | Burst Time |
|---|---|
| $P_1$ | 24 |
| $P_2$ | 3 |
| $P_3$ | 3 |




- Waiting time para $P_1$ = 0; $P_2$ = 24; $P_3$ = 27
- Waiting time promedio: (0 + 24 + 27)/3 = 17

# FCFS Scheduling (Cont.)

- Suponga que los procesos llegan en el orden $P_2$ , $P_3$ , $P_1$
- La gráfica de Gantt del despacho es:




- Waiting time para $P_1 = 6; P_2 = 0, P_3 = 3$
- Waiting time promedio: (6 + 0 + 3)/3 = 3
- Mucho mejor que el anterior
- *Efecto Convoy*: procesos cortos antes que los largos

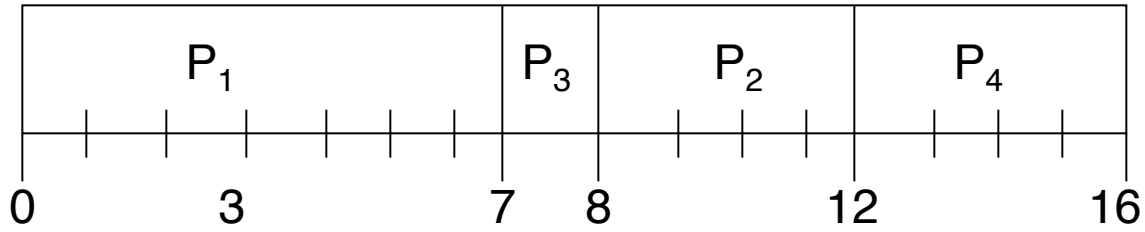
# Despacho Shortest-Job-First (SJF)

- Asociate con cada proceso la longitud de su próximo espacio de CPU (burst). Usar esas longitudes para asignar el procesador al proceso con el menor tiempo
- Dos esquemas :
  - nonpreemptive – una vez que se asigna el CPU a un proceso, no puede ser desalojado, hasta que termina su CPU burst
  - preemptive – si llega un proceso nuevo, con un CPU burst menor que el tiempo restante del proceso actual, desalojar. Este esquema se denomina Shortest-Remaining-Time-First (SRTF)
- SJF es óptimo – produce el mínimo promedio waiting time para un conjunto de procesos

# Ejemplo de un SJF sin Desalojo

| Proceso | Tiempo de Llegada | Burst Time |
|---|---|---|
| $P_1$ | 0.0 | 7 |
| $P_2$ | 2.0 | 4 |
| $P_3$ | 4.0 | 1 |
| $P_4$ | 5.0 | 4 |

- SJF (non-preemptive)

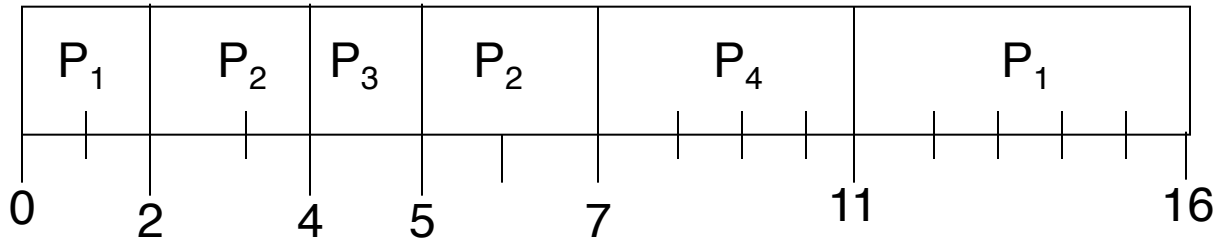



- Waiting time promedio = (0 + 6 + 3 + 7)/4 = 4

# Ejemplo de SJF con Desalojo

| Proceso | Tiempo de llegada | Burst Time |
|---|---|---|
| $P_1$ | 0.0 | 7 |
| $P_2$ | 2.0 | 4 |
| $P_3$ | 4.0 | 1 |
| $P_4$ | 5.0 | 4 |

- SJF (preemptive)




- Tiempo de espera (Waiting time) promedio
  = (9 + 1 + 0 +2)/4 = 3

# Longitud del Siguiente CPU Burst

- Estimación de la longitud

$$t_n = \text{longitud del } n - \textit{ésimo} \text{ CPU burst}$$

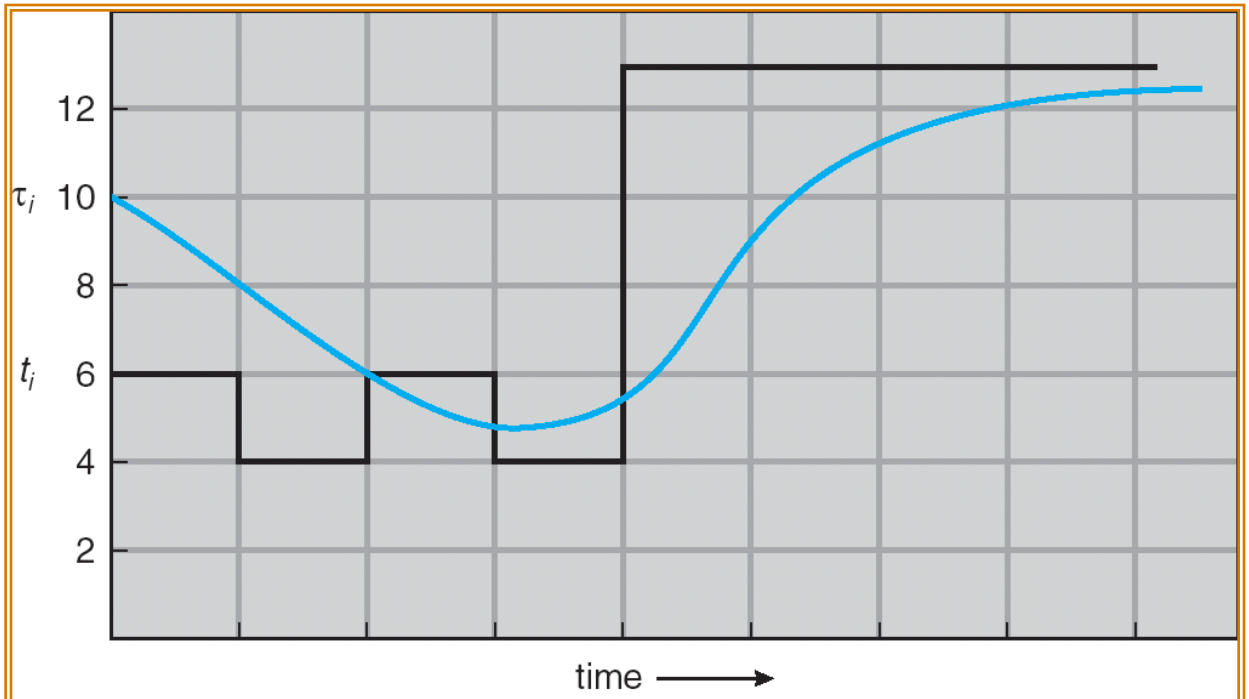
- Promedio exponencial de las longitudes de los CPU bursts previos

$$\tau_{n+1} = \text{predicción del siguiente CPU burst}$$

- Para $0 \le \alpha \le 1$ definir:

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

# Predicción de la Longitud del Siguiente CPU Burst




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| CPU burst ($t_i$) | 6 | 4 | 6 | 4 | 13 | 13 | 13 | ... |
| "guess" ($\tau_i$) | 10 | 8 | 6 | 6 | 5 | 9 | 11 | 12 | ... |

# Ejemplos de Promediado Exponencial

- $\alpha = 0$
  - $\tau_{n+1} = \tau_n$
  - La historia reciente no cuenta
- $\alpha = 1$
  - $\tau_{n+1} = \alpha\, t_n$
  - Solo el último CPU burst cuenta
- Expandiendo la fórmula:

$$\tau_{n+1} = \alpha\, t_n + (1 - \alpha)\alpha\, t_{n-1} + \ldots$$
$$+ (1 - \alpha)^j \alpha\, t_{n-j} + \ldots$$
$$+ (1 - \alpha)^{n+1} \tau_0$$

- Como $\alpha$ y $(1 - \alpha)$ son menores o iguales a 1, cada término sucesivo tiene menos peso que su predecesor

# Despacho por Prioridad

- Se asocia un número (entero) de prioridad con cada proceso
- El CPU se asigna al proceso con la prioridad más alta (número más pequeño ≡ prioridad más alta)
  - Preemptive
  - nonpreemptive
- SJF es un caso de despacho por prioridad, donde la prioridad es el tiempo predicho del siguiente CPU burst
- Problema ≡ Inanición (Starvation) – procesos de baja prioridad pueden quedarse en la cola para siempre
- Solución ≡ Envejecimiento (Aging) – Incrementar la prioridad conforme avanza el tiempo

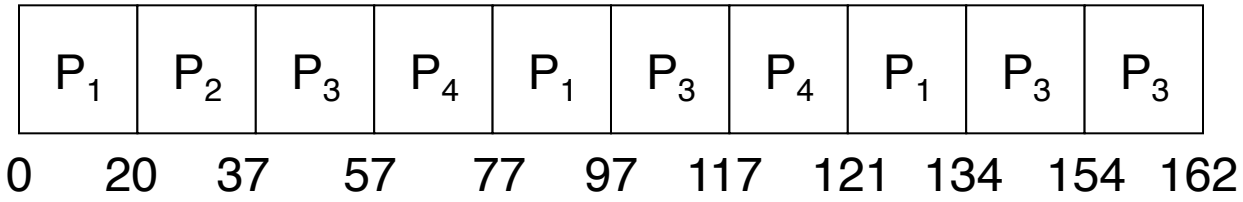
# Round Robin (RR)

- Se asigna a cada proceso una unidad de tiempo de CPU (*time quantum*), típicamente 10-100 milisegundos. Cuando se le termina el tiempo, el CPU se le retira y el proceso se envía al final de la cola ready queue.
- Si hay $n$ procesos en la cola y el time quantum es $q$, cada proceso obtiene $1/n$ del tiempo de CPU en periodos de a lo más $q$ unidades de tiempo consecutivas. Ningún proceso espera más de $(n\text{-}1)$ $q$ unidades de tiempo.
- Desempeño (Performance)
  - $q$ grande $\Rightarrow$ FIFO
  - $q$ pequeño $\Rightarrow$ $q$ debe ser grande con respecto al tiempo para hacer context switch, si no, el costo (overhead) es demasiado

# Ejemplo de RR con Time Quantum = 20

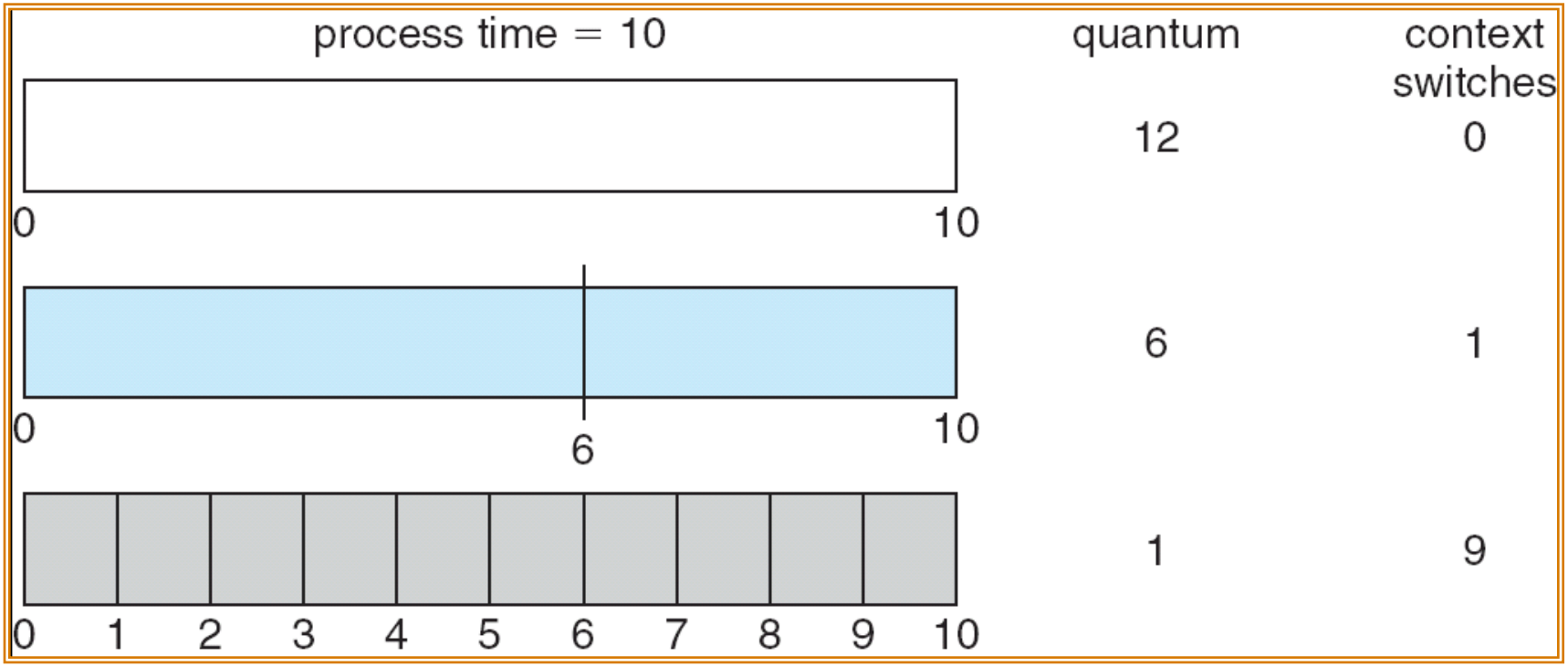
| Proceso | CPU Burst |
|---|---|
| $P_1$ | 53 |
| $P_2$ | 17 |
| $P_3$ | 68 |
| $P_4$ | 24 |

- Gráfica de Gantt:

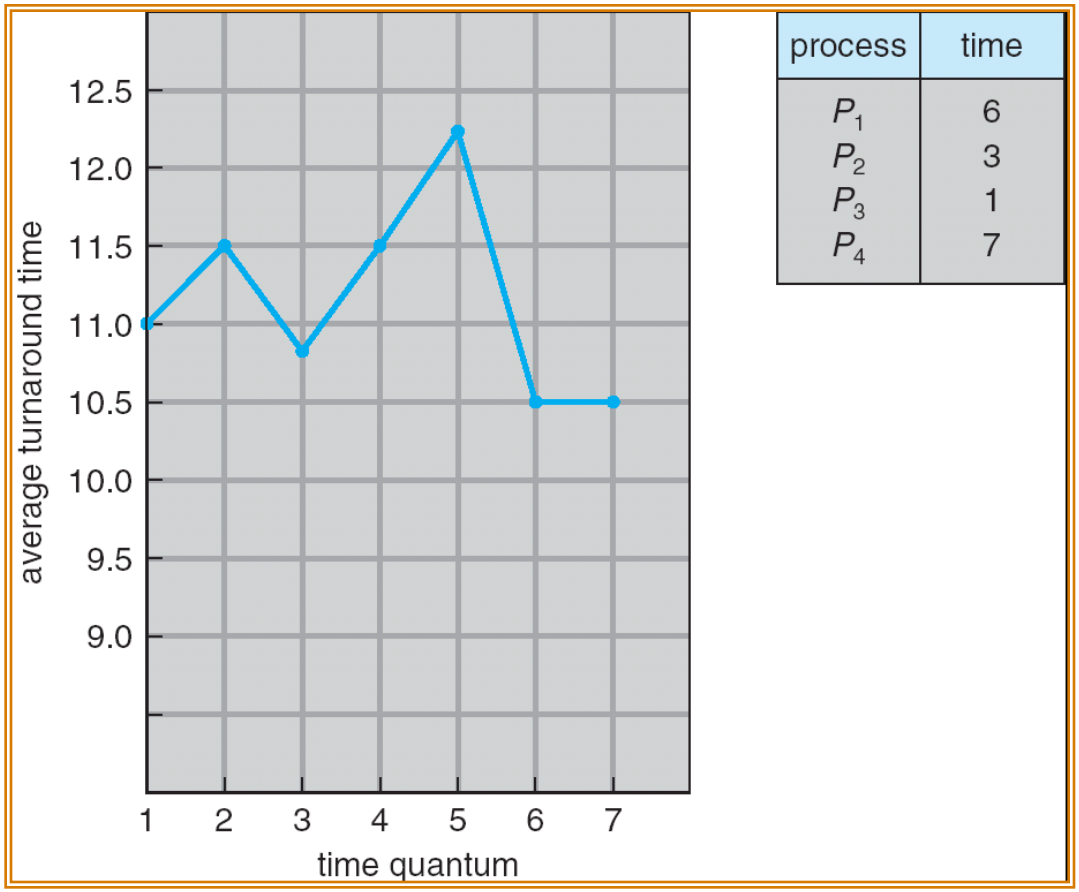



- Típicamente, el tiempo turnaround es mayor que en SJF, pero se tiene mejor *respuesta*

# Time Quantum y el Tiempo de Context Switch



# Tiempo Turnaround y el Time Quantum

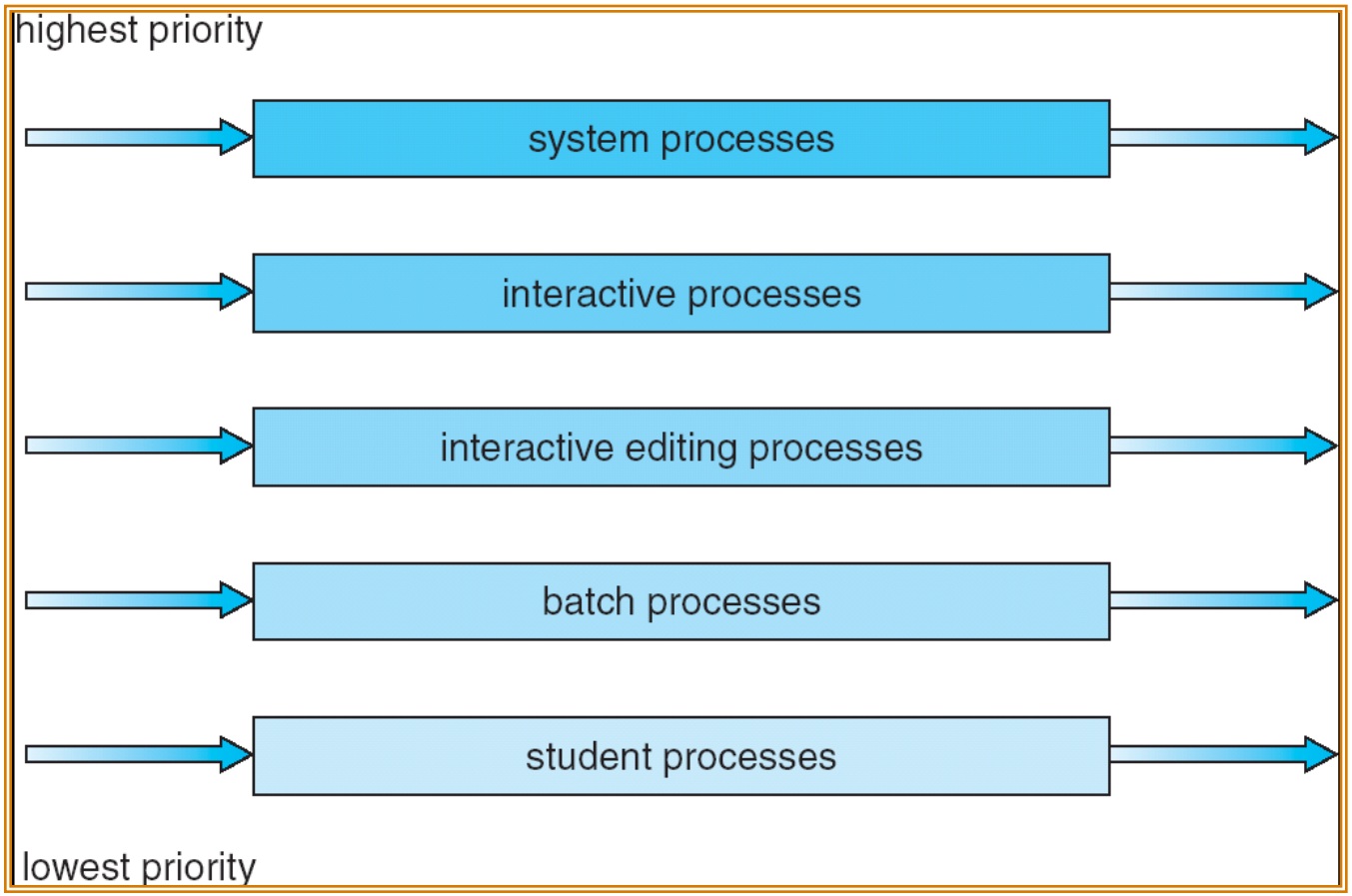



| process | time |
|---|---|
| $P_1$ | 6 |
| $P_2$ | 3 |
| $P_3$ | 1 |
| $P_4$ | 7 |

# Colas Multinivel

- La cola Ready queue se particiona en colas independientes:
  foreground (interactivos)
  background (batch)
- Cada cola tiene su propio algoritmo de despacho
  - foreground – RR
  - background – FCFS
- Se debe hacer despacho entre colas
  - Despacho de prioridad fija (i.e., servir a todos los del foreround y después a los del background). Posibilidad de inanición (starvation).
  - Rebanada de tiempo (Time slice) – cada cola obtiene una cantidad de tiempo de CPU, el cual puede despachar entre sus procesos; i.e., 80% al foreground en RR, 20% al background en FCFS

# Despacho en Colas Multinivel



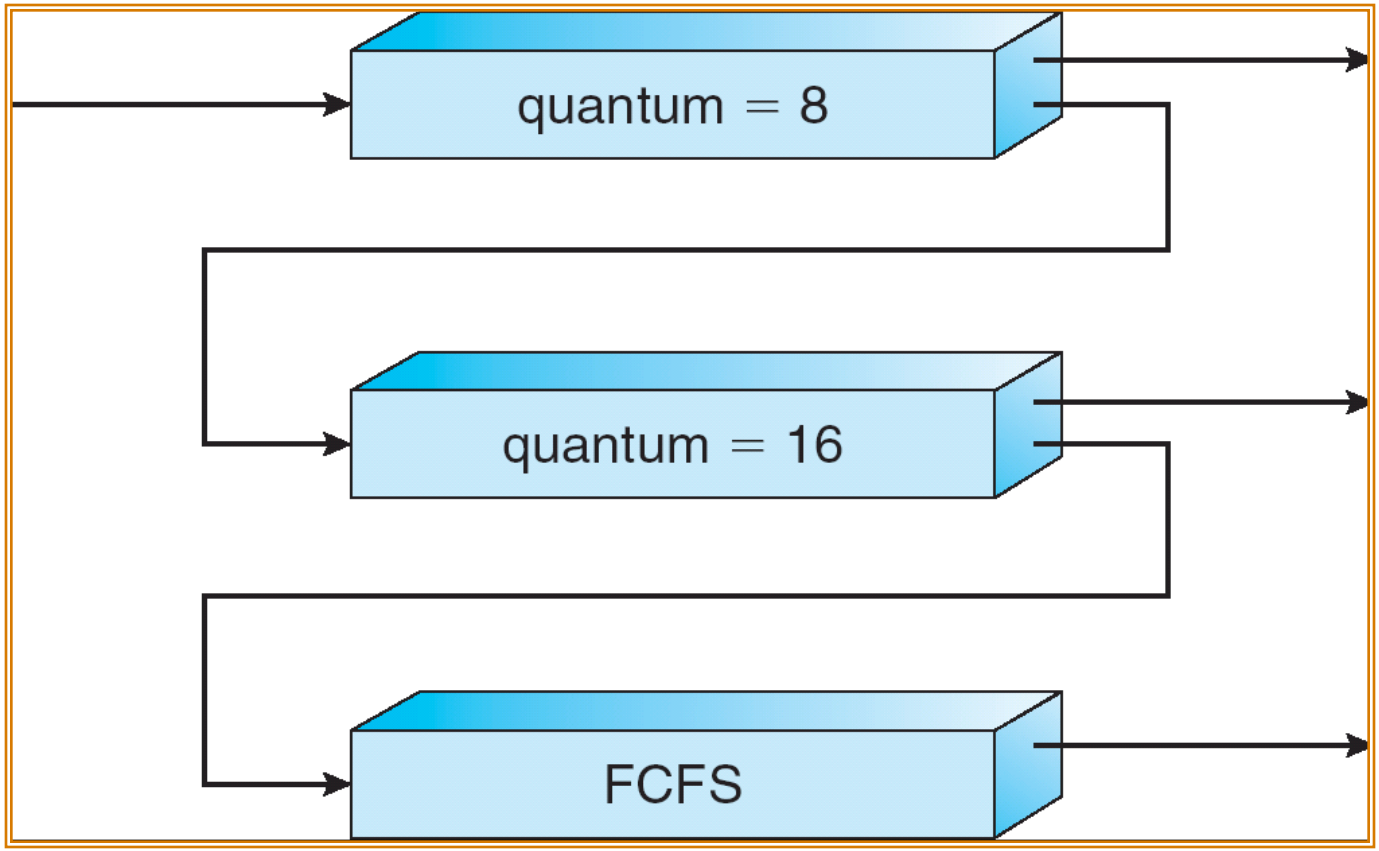
# Colas Multinivel con Retroalimentacón

- Un proceso puede migrar entre varias colas (i.e. aging)
- Un despachador multinivel con retroalimentación se define por los siguientes parámetros:
  - Número de colas
  - Algoritmos de despacho para cada cola
  - Método para determinar cuando promover un proceso
  - Método para determinar cuando demover a un proceso
  - Método para determinar a que cola se asigna un proceso cuando necesita servicio

# Ejemplo de Colas Multinivel con Retroalimentacón

- Tres colas:
  - $Q_0$ – RR con time quantum de 8 milisegundos
  - $Q_1$ – RR con time quantum de 16 milisegundos
  - $Q_2$ – FCFS
- Despacho (Scheduling)
  - Un proceso nuevo entra a $Q_0$ la cual opera en FCFS. Cuando le toca, recibe el CPU por 8 milisegundos. Si no termina en 8 milisegundos, migra a $Q_1$.
  - En $Q_1$ el proceso vuelve a ser servido mediante FCFS y recibe 16 milisegundos adicionales. Si aún no termina, se desaloja (preemption) y migra a $Q_2$.

# Colas Multinivel con Retroalimentacón



# Despacho de Threads

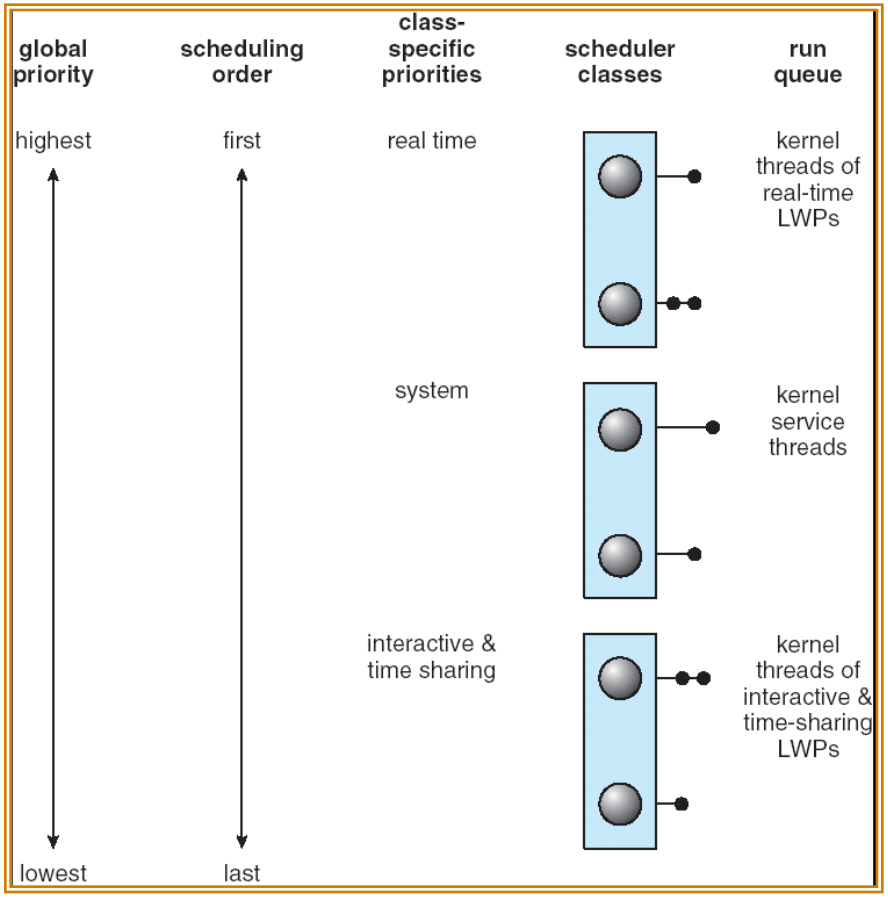
- Despacho local – ¿Cómo decide la biblioteca de threads qué thread poner en un LWP disponible?

- Despacho global – ¿Cómo decide el kernel que thread ejecutar?

# Ejemplos

- Solaris scheduling
- Windows XP scheduling
- Linux scheduling

# Despacho en Solaris 2



# Tabla de Despacho en Solaris

| 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 |

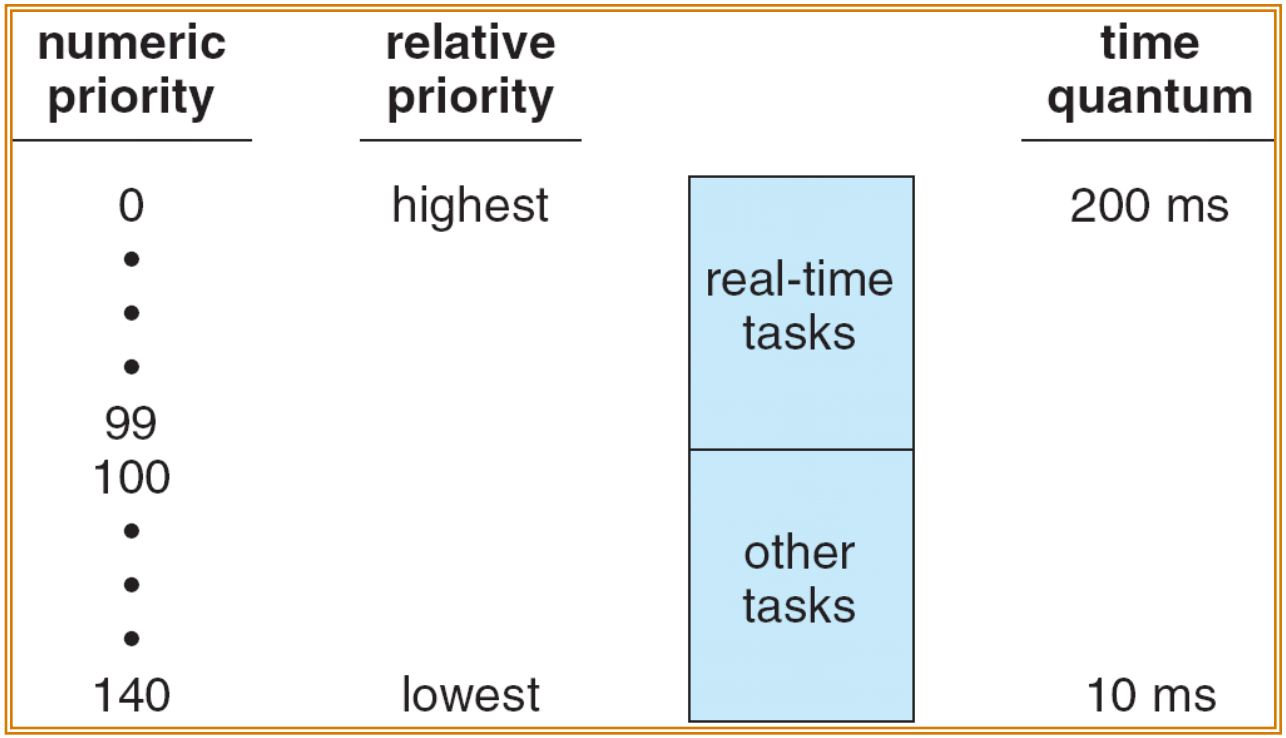
# Prioridades en Windows XP

| | 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 |

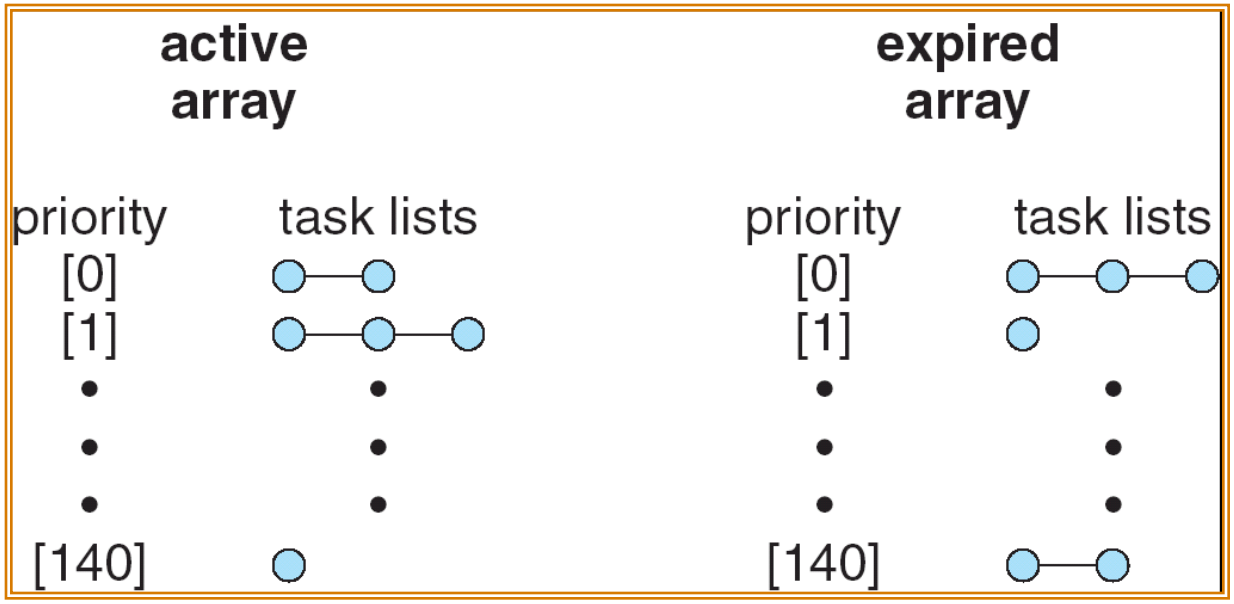
# Despacho en Linux

- Dos algoritmos: time-sharing y real-time
- Time-sharing
  - Priorides basadas en créditos – el proceso con más créditos es despachado
  - Restar crédito cuando el timer interrumpe
  - Cuando crédito = 0, se despacha otro proceso
  - Cuando todos los procesos tienen crédito = 0, restablecer créditos
    - Basado en prioridad, historia y otros factores
- Real-time
  - Soft real-time
  - Cumple con Posix.1b – dos clases
    - FCFS y RR
    - Procesos con alta prioridad son ejecutados primero

# Prioridades y el tamaño del Time-slice



# Listas de Tareas Indexadas por Prioridades

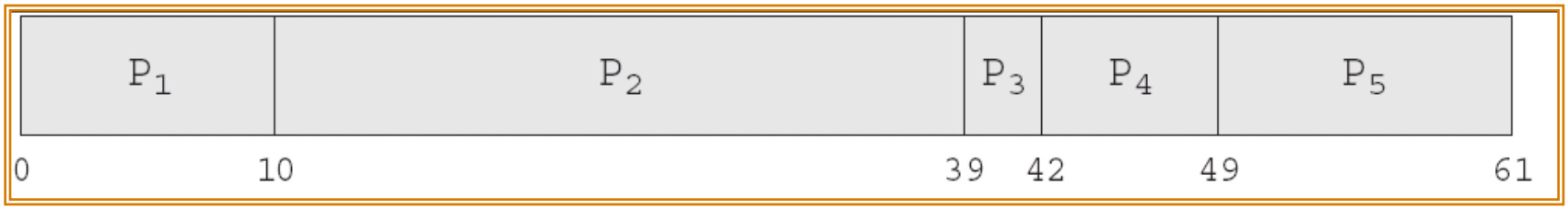


# Evaluación de Algoritmos

- Modelado determinista – toma una carga particular predetermnada y define el desempeño de cada algoritmo para esa carga
- Modelos de Colas (Queueing models)
- Implementación

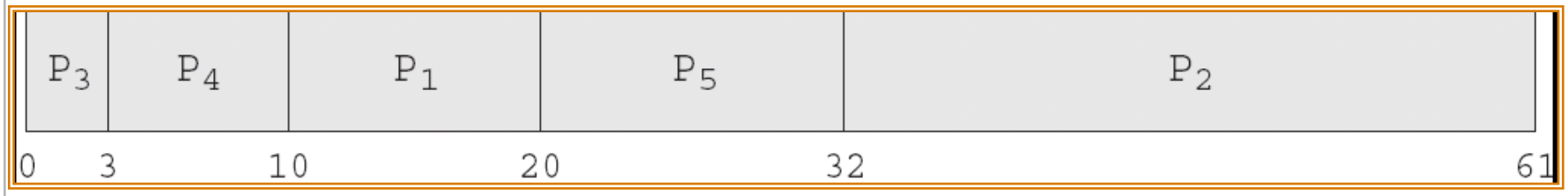
# Modelado Determinista FCFS

| Proceso | CPU Burst |
|---|---|
| $P_1$ | 10 |
| $P_2$ | 29 |
| $P_3$ | 3 |
| $P_4$ | 7 |
| $P_5$ | 12 |



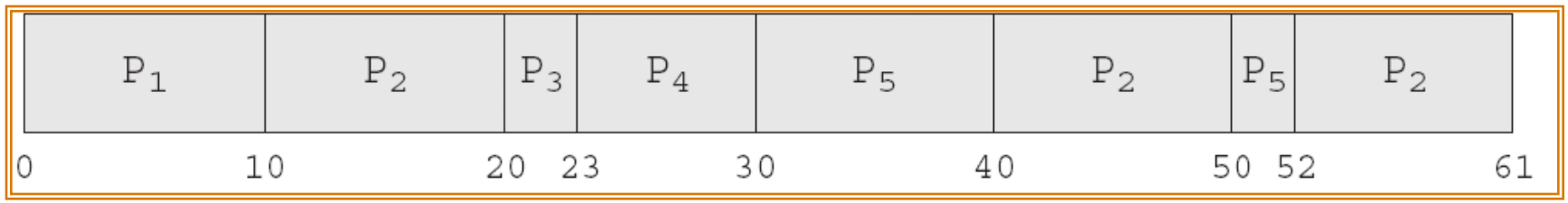
# Modelado Determinista SJF Sin Desalojo

| Proceso | CPU Burst |
|---|---|
| $P_1$ | 10 |
| $P_2$ | 29 |
| $P_3$ | 3 |
| $P_4$ | 7 |
| $P_5$ | 12 |



# Modelado Determinista Round Robin

| Proceso | CPU Burst |
|---|---|
| $P_1$ | 10 |
| $P_2$ | 29 |
| $P_3$ | 3 |
| $P_4$ | 7 |
| $P_5$ | 12 |



# Evaluación de Algoritmos