

# Lección 3

## Planificación de procesos

Sistemas Operativos  
Ingeniería Informática

# Lecturas recomendadas

---

## Base



1. Carretero 2020:
  1. Cap. 5
2. Carretero 2007:
  1. Cap. 3 y 4

## Recomendada



1. Tanenbaum 2006(en):
  1. Cap.3
2. Stallings 2005:
  1. 3.2, 3.3 y 3.5
3. Silberschatz 2006:
  1. 3.1 y 3.3

# ¡ATENCIÓN!

---

- ❑ Este material es un guión de la clase pero no son los apuntes de la asignatura.
- ❑ Los libros dados en la bibliografía junto con lo explicado en clase representa el material de estudio para el temario de la asignatura.

# Contenidos

---

1. Conceptos básicos de planificación de sistemas operativos.
2. Planificación y activación
3. Algoritmos de programación más comunes
  - ▶ FIFO, SJF, RR y PRIORIDAD.
4. Estructuras de datos de planificación en el núcleo.
  - ▶ Planificación en LINUX: envejecimiento.
5. Llamadas de planificación de procesos.

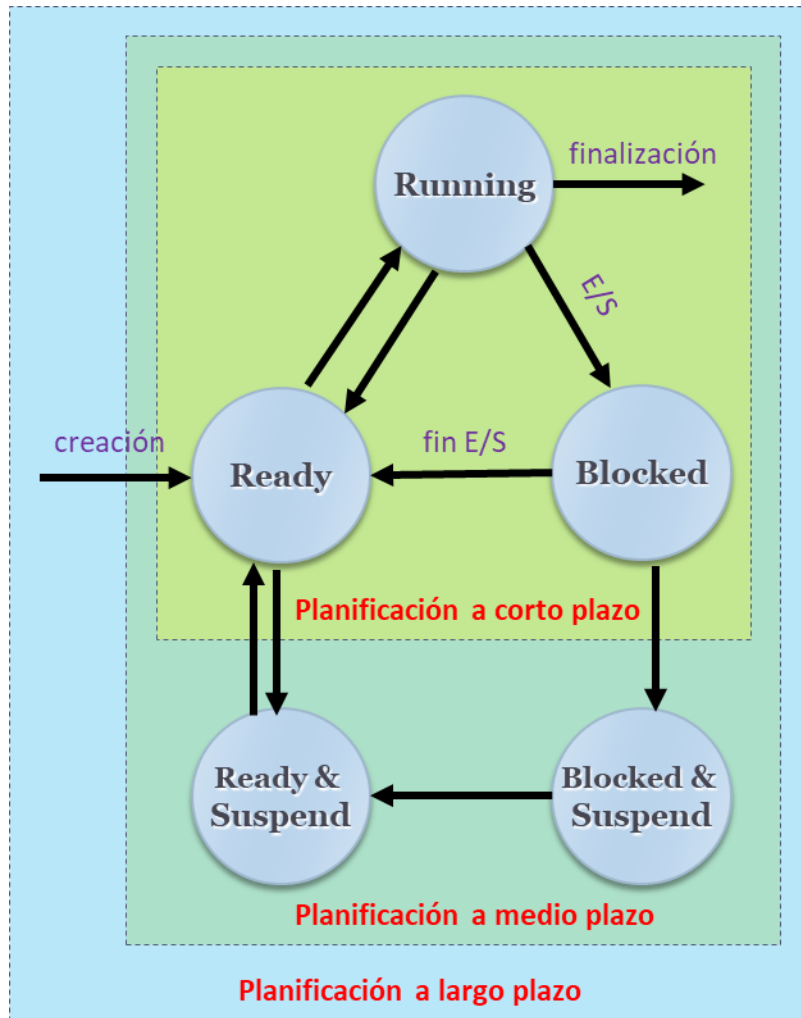
# Contenidos

---

1. **Conceptos básicos de planificación de sistemas operativos.**
2. Planificación y activación
3. Algoritmos de programación más comunes
  - ▶ FIFO, SJF, RR y PRIORIDAD.
4. Estructuras de datos de planificación en el núcleo.
  - ▶ Planificación en LINUX: envejecimiento.
5. Llamadas de planificación de procesos.

# Planificación de procesos

## niveles de planificación



- ▶ **A largo plazo**
  - ▶ Añadir procesos a ejecutar
    - ▶ Usado en procesamiento por lotes *batch*
- ▶ **A medio plazo**
  - ▶ Procesos a añadir/quitar de memoria principal
- ▶ **A corto plazo**
  - ▶ Selecciona el siguiente proceso a ejecutar
    - ▶ Invocado frecuentemente, rápido

# Planificación de procesos

## objetivos de los algoritmos de planificación (según sistema)

---

- ▶ Todos los sistemas:
  - ▶ **Equitativo** – ofrece a cada proceso una parte equitativa de la CPU
  - ▶ **Expeditivo** – cumplimiento de la política emprendida de reparto
  - ▶ **Balanceado** – mantener todas las partes del sistema ocupadas
- ▶ Sistemas *batch*:
  - ▶ **Productividad** – maximizar el número de trabajos por hora
  - ▶ **Tiempo de espera** – minimizar el tiempo entre emisión y terminación del trabajo
  - ▶ **Uso de CPU** – mantener la CPU ocupada todo el tiempo
- ▶ Sistemas **Interactivos**:
  - ▶ **Tiempo de respuesta** – responder a las peticiones lo más rápido posible
  - ▶ **Ajustado** – satisfacer las expectativas de los usuarios
- ▶ Sistemas de **tiempo real**:
  - ▶ **Cumplimiento de plazos** – evitar la pérdida de datos
  - ▶ **Predecible** – evitar la degradación de calidad en sistemas multimedia

# Planificación de procesos

## características de los algoritmos de planificación (1/2)

---

### ► *Preemption:*

#### ► Sin expulsión (no apropiativa):

- El proceso conserva la CPU mientras desee.
- Cambios de contexto voluntarios (C.C.V.)
- [V] Solución fácil a la compartición de recursos
- [I] Un proceso puede bloquear al resto
- Windows 3.1, Windows 95 (16 bits), NetWare, MacOS 9.x.

#### ► Con expulsión (apropiativa):

- Exige un reloj que interrumpe periódicamente:
  - cuando pasa el quantum de un proceso se cambia a otro
- (Se añade) Cambios de contexto involuntarios (C.C.I.)
- [V] Mejora la interactividad
- [I] Precisa de mecanismos para condiciones de carrera
- AmigaOS (1985), Windows NT-XP-Vista-7, Linux, BSD, MacOS X

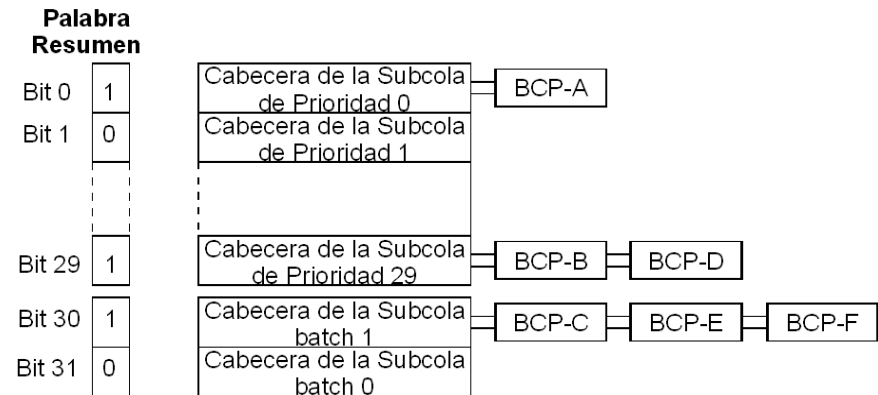


# Planificación de procesos

## características de los algoritmos de planificación (2/2)

### ► Clasificación de procesos (BCP \*) en las colas:

- Sin clasificación (cola única)
- Por tipo:
  - CPU-bound: + rachas de uso de CPU
  - IO-bound: + rachas de espera a E/S
- Por prioridad

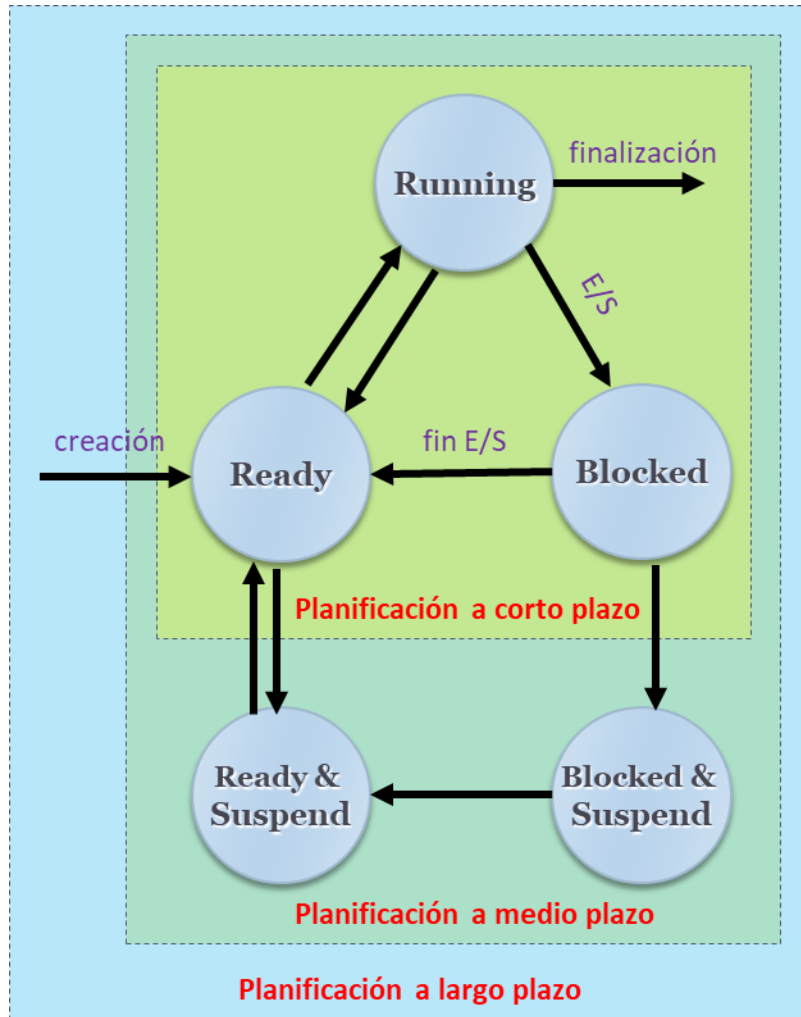


### ► CPU-aware:

- Afinidad:
  - Los procesos tienen 'afinidad' (*affinity*) a una CPU: «mejor volver a la misma CPU»
- Simetría:
  - Los procesos se ejecutan en la CPU que tienen unas capacidades específicas a dicha CPU

# Planificación de procesos

## puntos de decisión de planificación



### ► Posibles transiciones con replanificación:

1. Proceso se bloquea (por evento)
2. Al tratarse interrupción:
  - Interrupción del reloj.
  - Interrupción fin espera de evento.
3. Fin de ejecución del proceso

### ► Relación momento de decisión con tipo:

- Apropiativa: 1, 2 y 3
- NO apropiativa: 1 y 3

# Contenidos

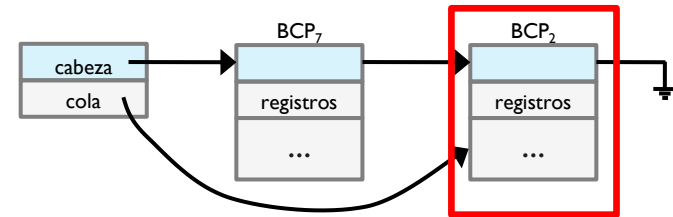
---

1. Conceptos básicos de planificación de sistemas operativos.
2. **Planificación y activación**
3. Algoritmos de programación más comunes
  - ▶ FIFO, SJF, RR y PRIORIDAD.
4. Estructuras de datos de planificación en el núcleo.
  - ▶ Planificación en LINUX: envejecimiento.
5. Llamadas de planificación de procesos.

# Planificador y activador

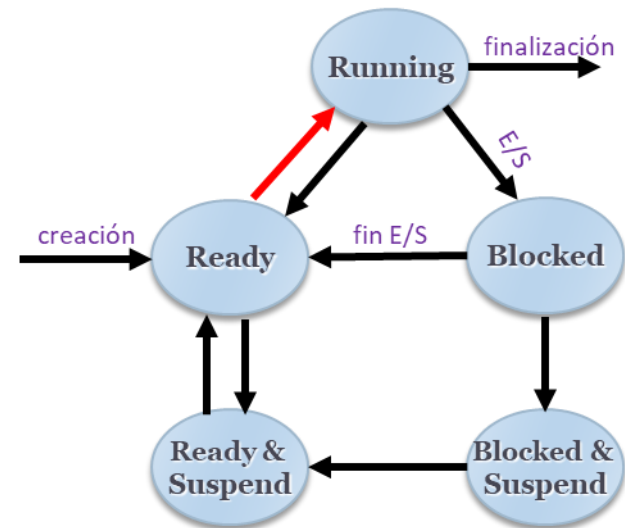
## ► Planificador:

Selecciona el proceso a ser ejecutado entre los que están listos para ejecutar



## ► Activador:

Da control al proceso que el planificador ha seleccionado (cambio de contexto - restaurar)



# Política vs mecanismo

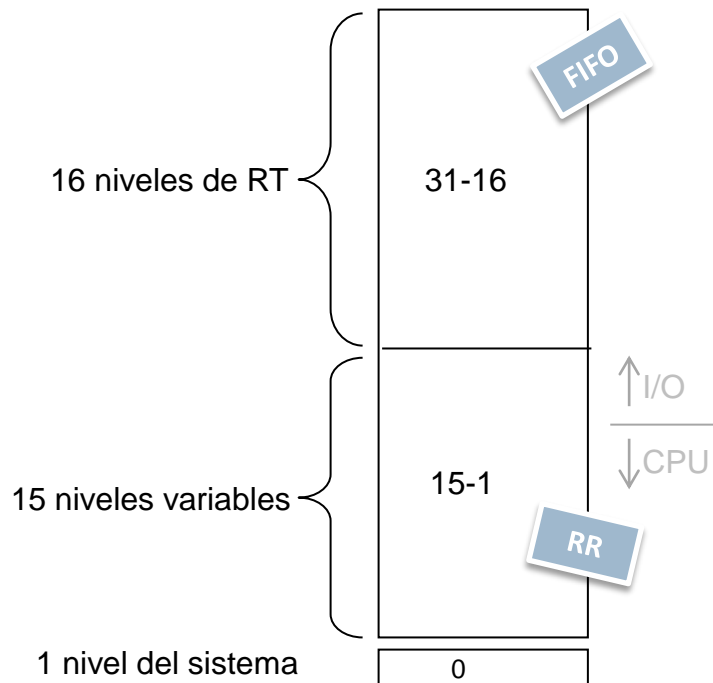
---

- ▶ Separación de lo qué se puede hacer de cómo se puede hacer
  - ▶ Normalmente, un proceso conoce cuál es el hilo más prioritario, el que más E/S necesitará, etc.
- ▶ Uso de algoritmos de planificación parametrizados
  - ▶ Mecanismo en el kernel
- ▶ Parámetros rellenos por los procesos de usuarios
  - ▶ Política establecida por los procesos de usuario

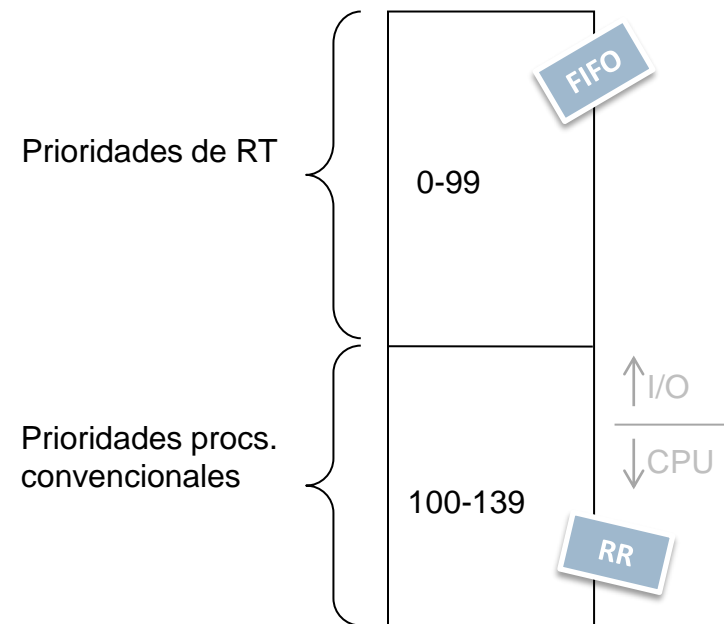
# Planificación multipolítica

## Windows 2000 y Linux

### Windows 2000



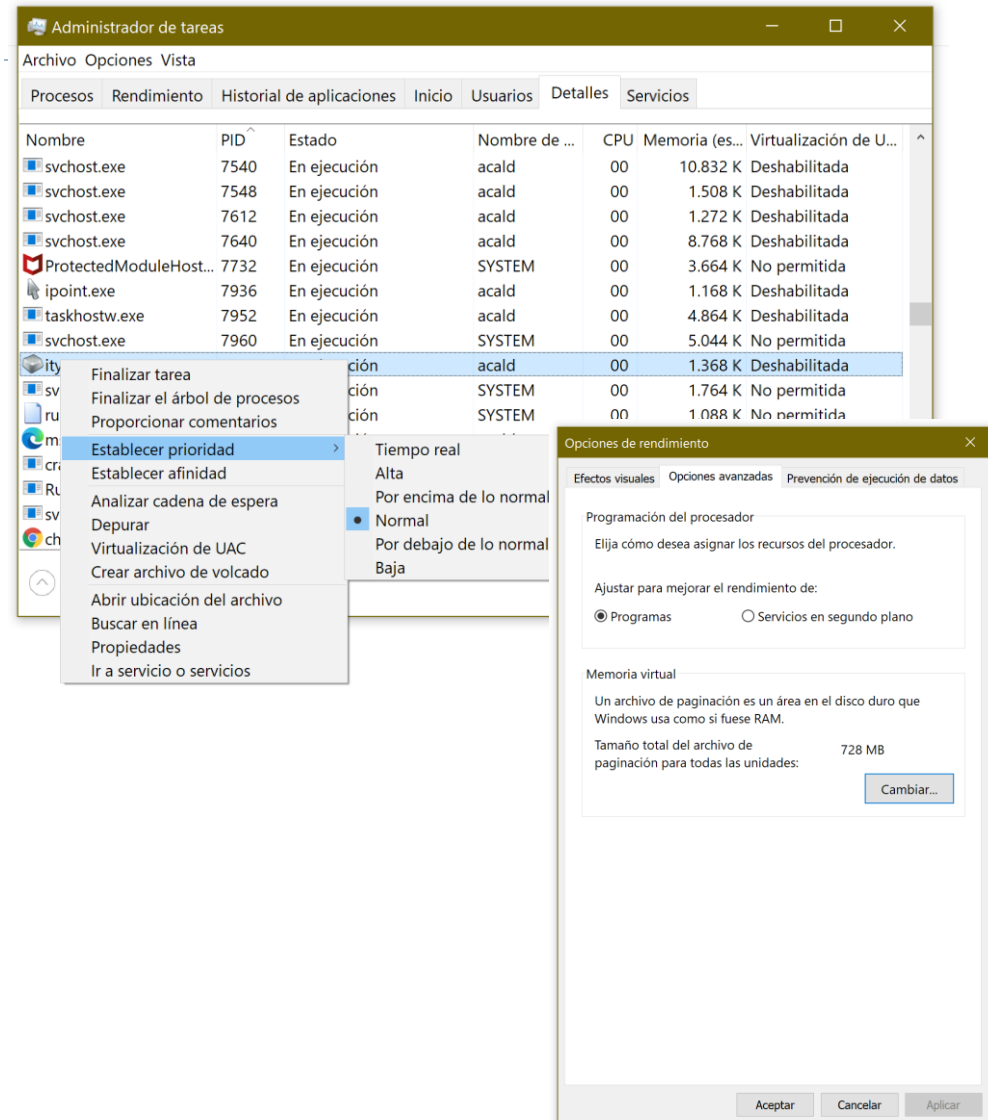
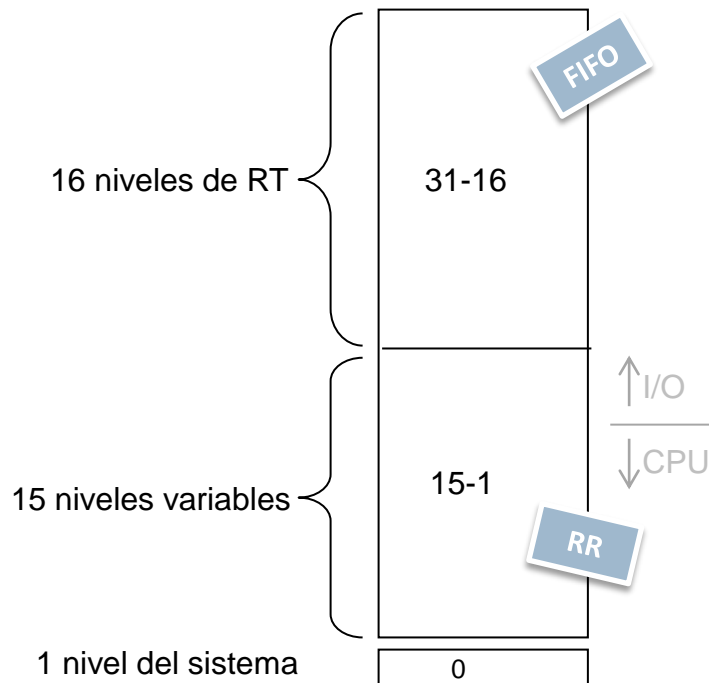
### Linux



# Planificación multipolítica

## Windows 2000 y Linux

### Windows 2000



# Contenidos

---

1. Conceptos básicos de planificación de sistemas operativos.
2. Planificación y activación
3. **Algoritmos de programación más comunes**
  - ▶ **FIFO, SJF, RR y PRIORIDAD.**
4. Estructuras de datos de planificación en el núcleo.
  - ▶ Planificación en LINUX: envejecimiento.
5. Llamadas de planificación de procesos.



# Planificación: Medidas

Nombre	Cálculo	Definición	Objetivo
Utilización de CPU	$U = TU / T$	Porcentaje de tiempo que se usa la CPU	Maximizar
Productividad	$P = NW / T$	Número de trabajos terminados por unidad de tiempo	Maximizar

Nombre	Cálculo	Definición	Objetivo
$T_q$ Tiempo de retorno	$T_q = T_f - T_i$	Tiempo que está un proceso en el sistema.	Minimizar
$T_s$ Tiempo de servicio	$T_s = T_{CPU} + T_{E/S}$	Tiempo dedicado a tareas productivas (CPU, Entrada/Salida).	
$T_e$ Tiempo de espera	$T_e = T_q - T_s$	Tiempo que un proceso pasa en colas de espera	
$T_n$ Tiempo de retorno normalizado	$T_n = T_q / T_s$	Indica el retardo experimentado. (tiempo de retorno / t. servicio)	

# Planificación: Principales algoritmos

	Nombre	Funcionamiento	Apropiativo	Desventaja
FCFS FIFO	<i>First to Come</i> <i>First to Serve</i>	Primer en llegar primero en servir	NO	<ul style="list-style-type: none"><li>• Penaliza a los procesos cortos</li></ul>
SJF	<i>Shortest Job</i> <i>First</i>	Primero el trabajo más corto	NO	<ul style="list-style-type: none"><li>• Se ha de saber de antemano la duración de cada trabajo.</li><li>• Posibilidad de inanición de trabajos largos (llegada de trabajos cortos continua)</li></ul>
RR	Cíclico o <i>Round-Robin</i>	Turno rotatorio	SI	<ul style="list-style-type: none"><li>• Los cambios de contextos generan retraso (aunque Rodaja &gt;&gt; tiempo de cambio de contexto)</li></ul>
Prio	Por prioridades	Se selecciona primero procesos de + prioridad		<ul style="list-style-type: none"><li>• Si prioridad fija entonces problema de inanición</li><li>• Mecanismos de envejecimiento</li></ul>

# Asignación FCFS

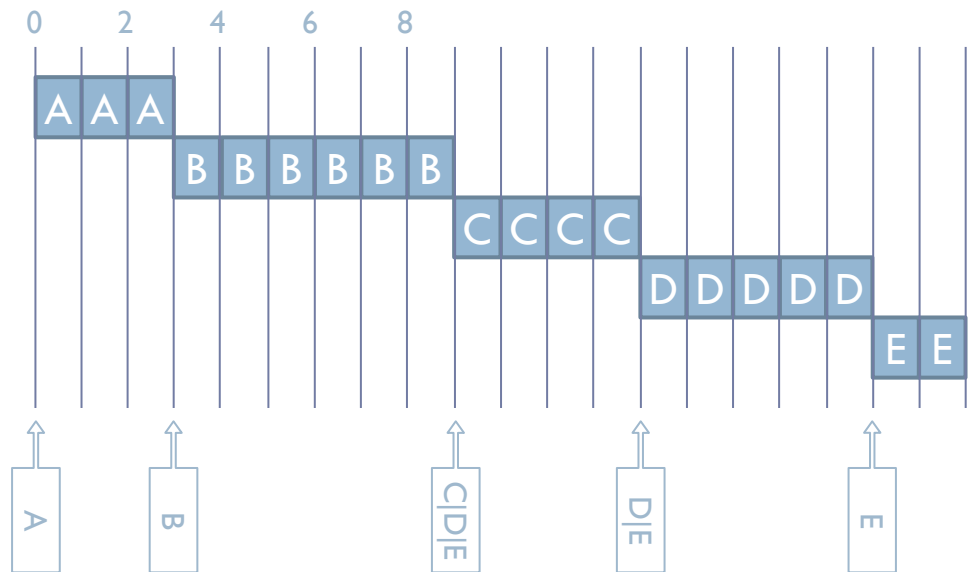
---

- ▶ **Iniciales:** FCFS (también FIFO)
- ▶ **Nombre:** *First to Come First to Serve*
- ▶ **Funcionamiento:** Primer en llegar primero en servir
- ▶ **Apropiativo:** NO
- ▶ **Desventajas:**
  - ▶ Penaliza a los procesos cortos

# Ejemplo FCFS (1/2)

- ▶ Mientras que haya procesos:
  - ▶ Se selecciona el proceso con menor T. llegada en el sistema.
  - ▶ Se ejecuta dicho proceso durante el T. servicio.

Proceso	Llegada	Servicio
<b>A</b>	0	3
<b>B</b>	2	6
<b>C</b>	4	4
<b>D</b>	6	5
<b>E</b>	8	2



## Ejemplo FCFS (2/2)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno normalizado
<b>A</b>	0	3					
<b>B</b>	2	6					
<b>C</b>	4	4					
<b>D</b>	6	5					
<b>E</b>	8	2					

1. Rellenar tiempo de inicio ( $T_i$ ) y fin ( $T_f$ )
  1.  $T_i$  es primero **0** y luego el  **$T_f$**  del anterior ejecutado.
  2.  $T_f$  es  $T_i +$  tiempo de servicio
  3. FIFO: mirar  **$T_i$**  y tomar el siguiente  **$T_i$**
2. Rellenar tiempo de retorno  $T_q = T_f - T_{llegada}$

## Ejemplo FCFS (2/2)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno normalizado
<b>A</b>	0	3	<b>0</b>	<b>3</b>	<b>3</b>		
<b>B</b>	2	6	<b>3</b>	<b>9</b>	<b>7</b>		
<b>C</b>	4	4	<b>9</b>	<b>13</b>	<b>9</b>		
<b>D</b>	6	5	<b>13</b>	<b>18</b>	<b>12</b>		
<b>E</b>	8	2	<b>18</b>	<b>20</b>	<b>12</b>		

### 1. Rellenar tiempo de inicio ( $T_i$ ) y fin ( $T_f$ )

1.  $T_i$  es primero **0** y luego el  **$T_f$**  del anterior ejecutado.
2.  $T_f$  es  $T_i +$  tiempo de servicio
3. FIFO: mirar  **$T_i$**  y tomar el siguiente  **$T_i$**

### 2. Rellenar tiempo de retorno $T_q = T_f - T_{llegada}$

## Ejemplo FCFS (2/2)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno normalizado
<b>A</b>	0	3	0	3	3		
<b>B</b>	2	6	3	9	7		
<b>C</b>	4	4	9	13	9		
<b>D</b>	6	5	13	18	12		
<b>E</b>	8	2	18	20	12		

1. Rellenar tiempo de espera:  $T_e = T_q - T_s$
2. Rellenar tiempo de retorno normalizado:  $T_n = T_q / T_s$

## Ejemplo FCFS (2/2)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno normalizado
<b>A</b>	0	3	0	3	3	<b>0</b>	<b>3/3=1</b>
<b>B</b>	2	6	3	9	7	<b>1</b>	<b>7/6=1.16</b>
<b>C</b>	4	4	9	13	9	<b>5</b>	<b>9/4=1.25</b>
<b>D</b>	6	5	13	18	12	<b>7</b>	<b>12/5=2.4</b>
<b>E</b>	8	2	18	20	12	<b>10</b>	<b>12/2=6</b>

1. Rellenar tiempo de espera:  $T_e = T_q - T_s$
2. Rellenar tiempo de retorno normalizado:  $T_n = T_q / T_s$



## Ejemplo FCFS (2/2)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno normalizado
<b>A</b>	0	3	0	3	3	0	$3/3=1$
<b>B</b>	2	6	3	9	7	1	$7/6=1.16$
<b>C</b>	4	4	9	13	9	5	$9/4=1.25$
<b>D</b>	6	5	13	18	12	7	$12/5=2.4$
<b>E</b>	8	2	18	20	12	10	$12/2=6$

- ▶  $T_e$ : Tiempo medio de espera: **4.6**
- ▶  $T_n$ : Tiempo medio de retorno normalizado: **2.5**

# Asignación SJF

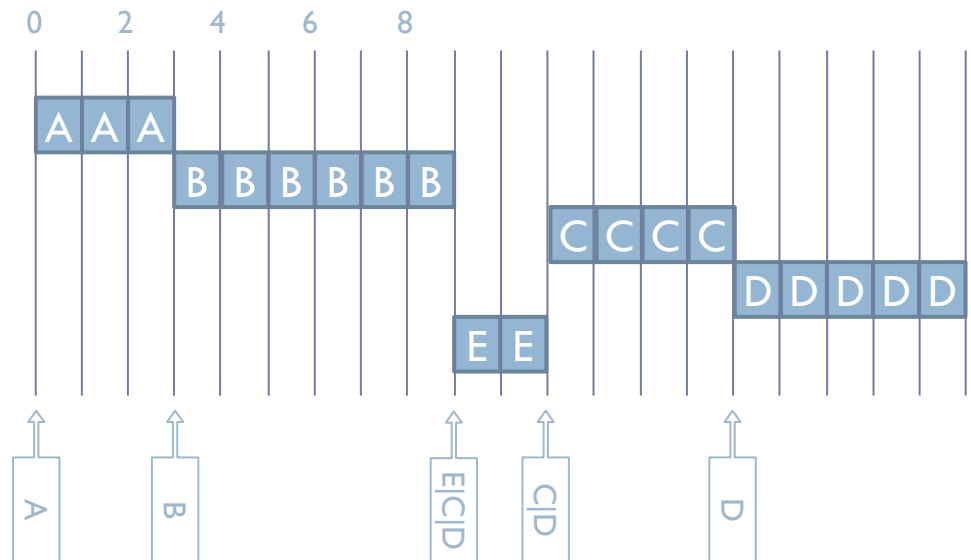
---

- ▶ **Iniciales:** SJF
- ▶ **Nombre:** *Shortest Job First*
- ▶ **Funcionamiento:** Primer el trabajo más corto:  
se selecciona el trabajo más corto
- ▶ **Apropiativo:** NO
- ▶ **Desventajas:**
  - ▶ Se ha de saber de antemano la duración de cada trabajo.
  - ▶ Posibilidad de inanición de trabajos largos  
(llegada de trabajos cortos continua)

# Ejemplo SJF (1 / 2)

- ▶ Mientras que haya procesos:
  - ▶ Se selecciona el proceso con menor T. servicio en el sistema.
  - ▶ Se ejecuta dicho proceso durante el T. servicio.

Proceso	Llegada	Servicio
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2



## Ejemplo SJF (2/2)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno normalizado
<b>A</b>	0	3					
<b>B</b>	2	6					
<b>C</b>	4	4					
<b>D</b>	6	5					
<b>E</b>	8	2					

### 1. Rellenar tiempo de inicio ( $T_i$ ) y fin ( $T_f$ )

1.  $T_i$  es primero **0** y luego el  **$T_f$**  del anterior ejecutado.
2.  $T_f$  es  $T_i +$  tiempo de servicio
3. SJF: mirar  $T_f$  y tomar el primero de los procesos con  $T_i$  menor o igual

### 2. Rellenar tiempo de retorno $T_q = T_f - T_{\text{llegada}}$

## Ejemplo SJF (2/2)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno normalizado
A	0	3	0	3	3		
B	2	6	3	9	7		
C	4	4	11	15	11		
D	6	5	15	20	14		
E	8	2	9	11	3		

### 1. Rellenar tiempo de inicio ( $T_i$ ) y fin ( $T_f$ )

1.  $T_i$  es primero **0** y luego el  **$T_f$**  del anterior ejecutado.
2.  $T_f$  es  $T_i +$  tiempo de servicio
3. SJF: mirar  $T_f$  y tomar el primero de los procesos con  $T_i$  menor o igual

### 2. Rellenar tiempo de retorno $T_q = T_f - T_{\text{llegada}}$

## Ejemplo SJF (2/2)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno normalizado
<b>A</b>	0	3	0	3	3		
<b>B</b>	2	6	3	9	7		
<b>C</b>	4	4	11	15	11		
<b>D</b>	6	5	15	20	14		
<b>E</b>	8	2	9	11	3		

1. Rellenar tiempo de espera:  $T_e = T_q - T_s$
2. Rellenar tiempo de retorno normalizado:  $T_n = T_q / T_s$

## Ejemplo SJF (2/2)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno normalizado
<b>A</b>	0	3	0	3	3	<b>0</b>	<b>3/3=1</b>
<b>B</b>	2	6	3	9	7	<b>1</b>	<b>7/6=1.16</b>
<b>C</b>	4	4	11	15	11	<b>7</b>	<b>11/4=2.75</b>
<b>D</b>	6	5	15	20	14	<b>9</b>	<b>14/5=2.8</b>
<b>E</b>	8	2	9	11	3	<b>1</b>	<b>3/2=1.5</b>

1. Rellenar tiempo de espera:  $T_e = T_q - T_s$
2. Rellenar tiempo de retorno normalizado:  $T_n = T_q / T_s$

## Ejemplo SJF (2/2)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno normalizado
<b>A</b>	0	3	0	3	3	0	$3/3=1$
<b>B</b>	2	6	3	9	7	1	$7/6=1.16$
<b>C</b>	4	4	11	15	11	7	$11/4=2.75$
<b>D</b>	6	5	15	20	14	9	$14/5=2.8$
<b>E</b>	8	2	9	11	3	1	$3/2=1.5$

- ▶  $T_e$ : Tiempo medio de espera: ~~4.6~~ **3.6**
- ▶  $T_n$ : Tiempo medio de retorno normalizado: ~~2.5~~ **1.84**



# Cíclico o Round-Robin

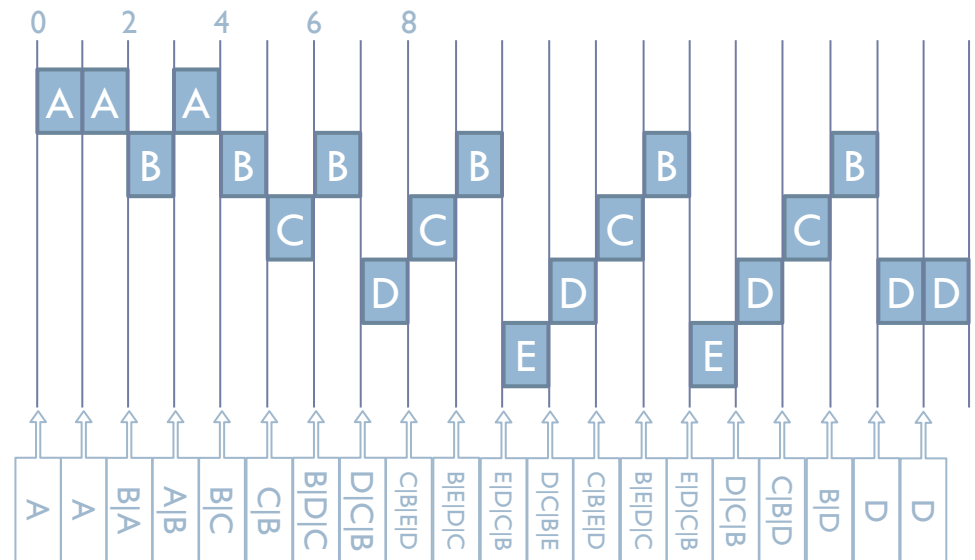
---

- ▶ **Iniciales:** RR
- ▶ **Nombre:** *Cíclico o Round-Robin*
- ▶ **Funcionamiento:** Turno rotatorio
  - ▶ Hay una cola FIFO con los procesos listos para ser ejecutados.
  - ▶ Se selecciona el primer proceso para ejecutar en procesador
  - ▶ El proceso ejecutará hasta
    - Finalizar durante un cuanto o rodaja de tiempo, y vuelve al final de la cola de listos.
    - Se queda bloqueado por un evento y va al final de la cola de bloqueados correspondiente.
  - ▶ Si proceso bloqueado y llega evento entonces pasa al final de cola de listos.
- ▶ **Apropiativo:** SI
- ▶ **Desventajas:**
  - ▶ Los cambios de contextos generan retraso  
(aunque Rodaja sea más grande que el tiempo de cambio de contexto)

# Ejemplo Round-Robin (q=1)

- ▶ Mientras que haya procesos:
  - ▶ Se selecciona el primer proceso listo para ejecutar de la lista
  - ▶ Se ejecuta dicho proceso durante  $q=1$  (rodaja) o fin de ejecución
    - ▶ Se pone al final de la lista de listos los que lleguen durante rodaja (e instante llegada)
  - ▶ Se pone al final de la lista de listo el proceso ejecutado

Proceso	Llegada	Servicio
<b>A</b>	0	3
<b>B</b>	2	6
<b>C</b>	4	4
<b>D</b>	6	5
<b>E</b>	8	2



# Ejemplo Round-Robin (q=1)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno normalizado
A	0	3					
B	2	6					
C	4	4					
D	6	5					
E	8	2					

## 1. Dibujar y rellenar tiempo de inicio ( $T_i$ ) y fin ( $T_f$ )

1.  $T_i$  es primero 0 y luego el instante en que se planifica por primera vez.
2.  $T_f$  es el instante en que se ha ejecutado por completo.
3. RR: [llega proceso -> primero en lista], tomar primer proceso en lista + ejecuta rodaja (q) + se pone el último de la lista

## 2. Rellenar tiempo de retorno $T_q = T_f - T_{\text{llegada}}$

# Ejemplo Round-Robin (q=1)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno normalizado
A	0	3	0	4	4		
B	2	6	2	18	16		
C	4	4	5	17	13		
D	6	5	7	20	14		
E	8	2	10	15	7		

## 1. Dibujar y rellenar tiempo de inicio ( $T_i$ ) y fin ( $T_f$ )

1.  $T_i$  es primero 0 y luego el instante en que se planifica por primera vez.
2.  $T_f$  es el instante en que se ha ejecutado por completo.
3. RR: [llega proceso  $\rightarrow$  primero en lista], tomar primer proceso en lista + ejecuta rodaja (q) + se pone el último de la lista

## 2. Rellenar tiempo de retorno $T_q = T_f - T_{\text{llegada}}$

# Ejemplo Round-Robin (q=1)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno normalizado
A	0	3	0	4	4		
B	2	6	2	18	16		
C	4	4	5	17	13		
D	6	5	7	20	14		
E	8	2	10	15	7		

1. Rellenar tiempo de espera:  $T_e = T_q - T_s$
2. Rellenar tiempo de retorno normalizado:  $T_n = T_q / T_s$

# Ejemplo Round-Robin (q=1)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno normalizado
A	0	3	0	4	4	1	$4/3=1.33$
B	2	6	2	18	16	10	$16/6=2.66$
C	4	4	5	17	13	9	$13/4=3.25$
D	6	5	7	20	14	9	$14/5=2.8$
E	8	2	10	15	7	5	$7/2=3.5$

1. Rellenar tiempo de espera:  $T_e = T_q - T_s$
2. Rellenar tiempo de retorno normalizado:  $T_n = T_q / T_s$

# Ejemplo Round-Robin (q=1)

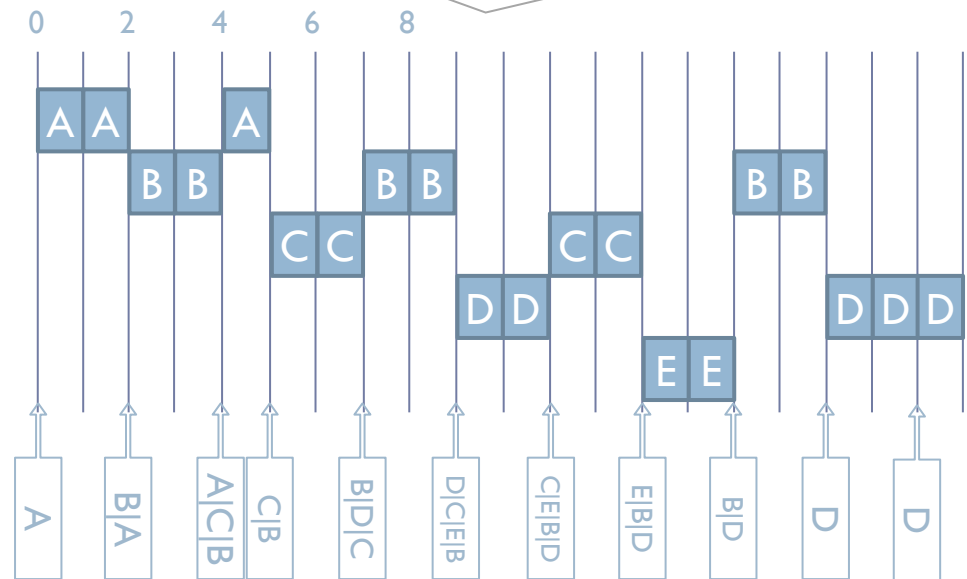
Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno normalizado
A	0	3	0	4	4	1	$4/3=1.33$
B	2	6	2	18	16	10	$16/6=2.66$
C	4	4	5	17	13	9	$13/4=3.25$
D	6	5	7	20	14	9	$14/5=2.8$
E	8	2	10	15	7	5	$7/2=3.5$

- ▶  $T_e$ : Tiempo medio de espera: ~~4.6~~ ~~3.6~~ **6.8**
- ▶  $T_n$ : Tiempo medio de retorno normalizado: ~~2.5~~ ~~1.84~~ **2.71**

# Ejemplo Round-Robin ( $q=2$ )

- ▶ Mientras que haya procesos:
  - ▶ Se selecciona el primer proceso listo para ejecutar de la lista
  - ▶ Se ejecuta dicho proceso durante  $q=2$  (rodaja) o fin de ejecución
    - ▶ Se pone al final de la lista de listos los que lleguen durante rodaja (e instante llegada)
  - ▶ Se pone al final de la lista de listo el proceso ejecutado

Proceso	Llegada	Servicio
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2





# Ejemplo Round-Robin (q=2)

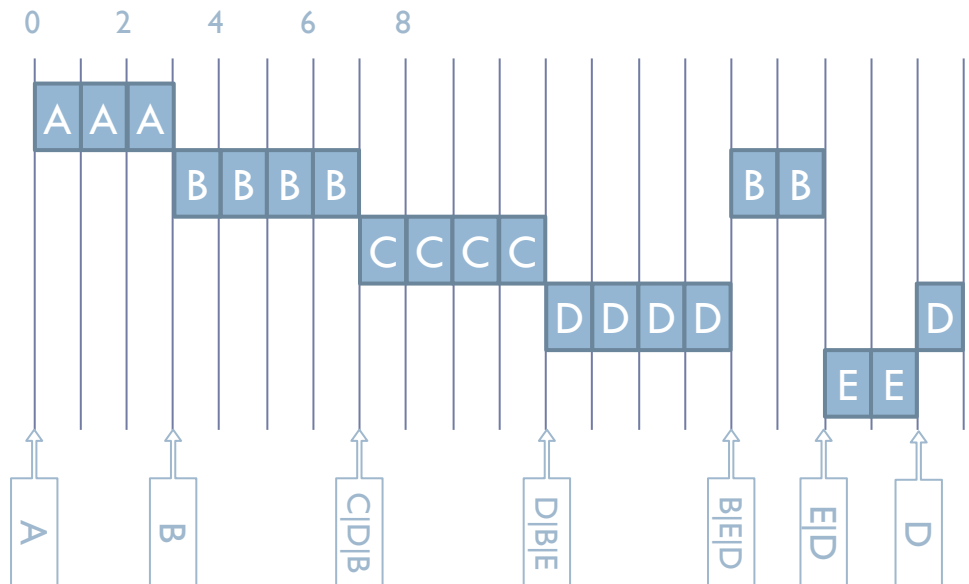
Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno normalizado
A	0	3	0	5	4	1	$4/3=1.33$
B	2	6	2	17	16	10	$16/6=2.66$
C	4	4	5	13	13	9	$13/4=3.25$
D	6	5	9	20	14	9	$14/5=2.8$
E	8	2	13	15	7	5	$7/2=3.5$

- ▶  $T_e$ : Tiempo medio de espera: ~~4.6~~ ~~3.6~~ **6**
- ▶  $T_n$ : Tiempo medio de retorno normalizado: ~~2.5~~ ~~1.84~~ **2.54**

# Ejemplo Round-Robin ( $q=4$ )

- ▶ Mientras que haya procesos:
  - ▶ Se selecciona el primer proceso listo para ejecutar de la lista
  - ▶ Se ejecuta dicho proceso durante  $q=4$  (rodaja) o fin de ejecución
    - ▶ Se pone al final de la lista de listos los que lleguen durante rodaja (e instante llegada)
  - ▶ Se pone al final de la lista de listo el proceso ejecutado

Proceso	Llegada	Servicio
<b>A</b>	0	3
<b>B</b>	2	6
<b>C</b>	4	4
<b>D</b>	6	5
<b>E</b>	8	2



# Ejemplo Round-Robin (q=4)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno normalizado
A	0	3	0	3	3	0	$3/3=1$
B	2	6	3	17	15	9	$15/6=2.5$
C	4	4	7	11	7	3	$7/4=1.75$
D	6	5	11	20	14	9	$14/5=2.8$
E	8	2	17	19	11	9	$11/2=5.5$

- ▶  $T_e$ : Tiempo medio de espera: ~~4.6~~ ~~3.6~~ **6**
- ▶  $T_n$ : Tiempo medio de retorno normalizado: ~~2.5~~ ~~1.84~~ **2.71**

# Asignación por prioridades

---

- ▶ **Iniciales:** Prio
- ▶ **Nombre:** Por prioridades
- ▶ **Funcionamiento:** El de mayor prioridad
  - ▶ Cada procesos tiene asignada una prioridad
  - ▶ Hay una cola FIFO por cada prioridad
  - ▶ Se selecciona el primer proceso que haya de la cola de más prioridad de entre todas.
- ▶ **Apropiativo:** NO
- ▶ **Desventajas:**
  - ▶ Si prioridad fija entonces problema de inanición
  - ▶ Solución: aplicar mecanismos de envejecimiento de forma que los de menor prioridad con más tiempo “crezcan” de prioridad temporalmente.

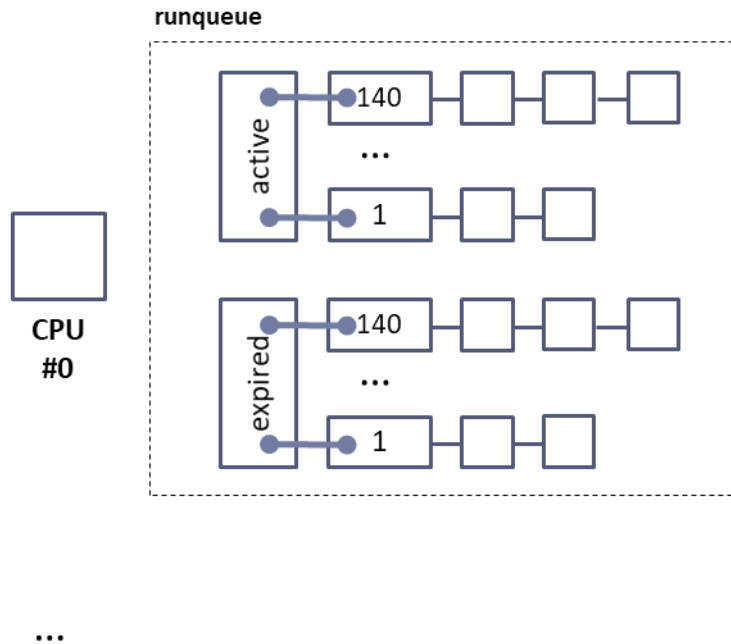
# Contenidos

---

1. Conceptos básicos de planificación de sistemas operativos.
2. Planificación y activación
3. Algoritmos de programación más comunes
  - ▶ FIFO, SJF, RR y PRIORIDAD.
4. **Estructuras de datos de planificación en el núcleo.**
  - ▶ **Planificación en LINUX: envejecimiento.**
5. Llamadas de planificación de procesos.

# Planificación: estructuras de datos

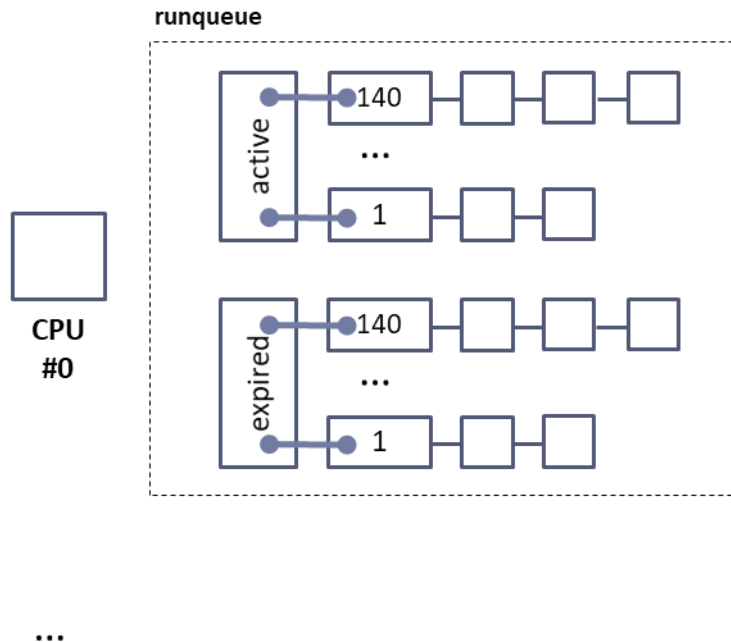
## Linux



- ▶ Kernel/sched.c
- ▶ Cada procesador tiene su propio *runqueue*
- ▶ Cada *runqueue* tiene dos vectores de prioridad:
  - ▶ Activo y Expirado
- ▶ Cada vector de prioridad tiene 140 listas:
  - ▶ Una por nivel de prioridad
  - ▶ Incluye 100 niveles de tiempo real

# Planificación: gestión

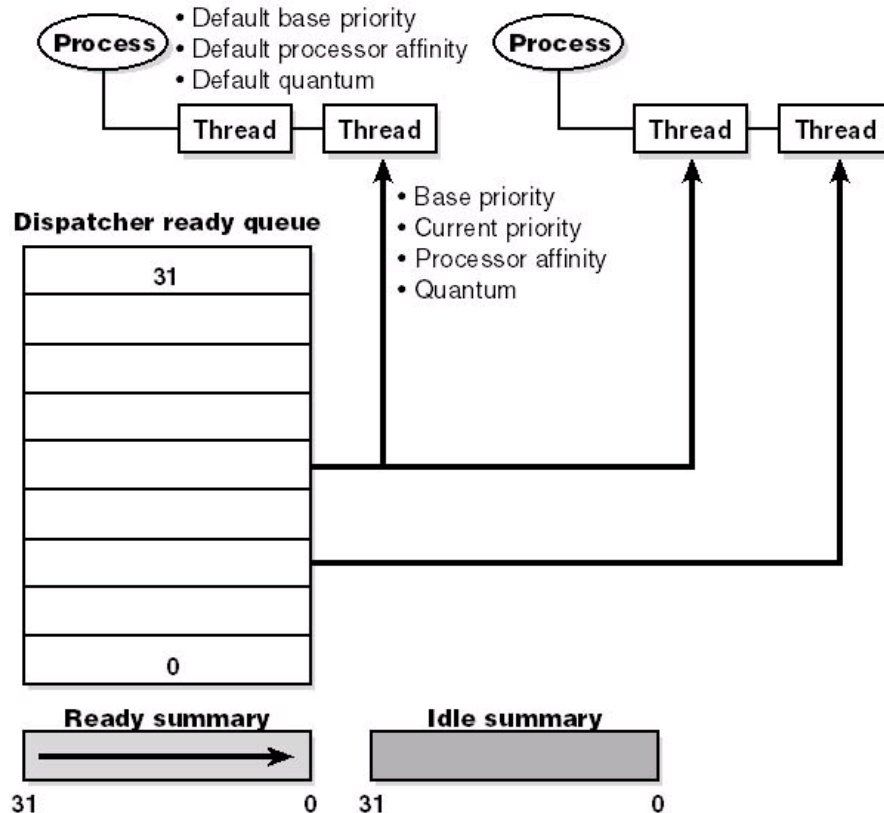
## Linux



- ▶ El planificador elige los procesos de la lista de activos de acuerdo a su prioridad
- ▶ Cuando expira la rodaja de un proceso, lo mueve a la lista de Expirado
  - ▶ Se recalcula prioridad y rodaja
- ▶ Cuando la lista de activos está vacía, el planificador intercambia las listas de activo y expirados
- ▶ Si un proceso es suficientemente interactivo permanecerá en la lista de activos

# Planificación: estructuras de datos

## Windows 2000



- ▶ *Dispatcher database:*
  - ▶ base de datos de hilos esperando para ejecutar y a qué proceso pertenecen
- ▶ *Dispatcher ready queue*
  - ▶ Una cola por nivel de prioridad
- ▶ *Ready summary*
  - ▶ Un bit por nivel
  - ▶ Si  $\text{bit}_i = 1 \rightarrow$  un hilo en ese nivel
  - ▶ Aumenta velocidad de búsqueda
- ▶ *Idle summary*
  - ▶ Un bit por procesador
  - ▶ Si  $\text{bit} = 1 \rightarrow$  procesador libre

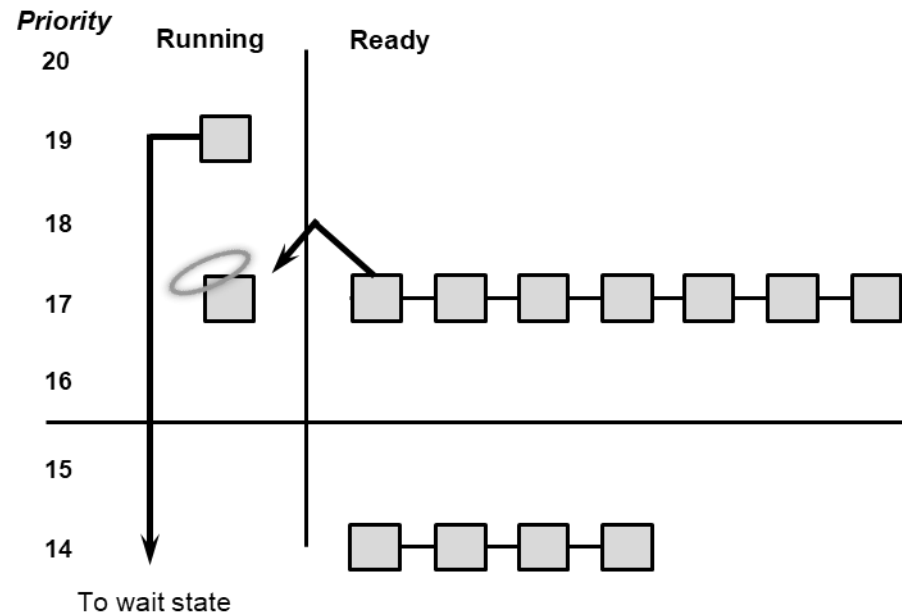


# Planificación: escenarios (1 / 3)

## Windows 2000

### ► Cambio de contexto voluntario:

- Entra en el estado de espera por algún objeto:
  - evento, mutex, semáforo, operación de E/S, etc.
- Al terminar pasa al final de la cola de listos + *temporary priority boost*.
- Rodaja de T: se mantiene

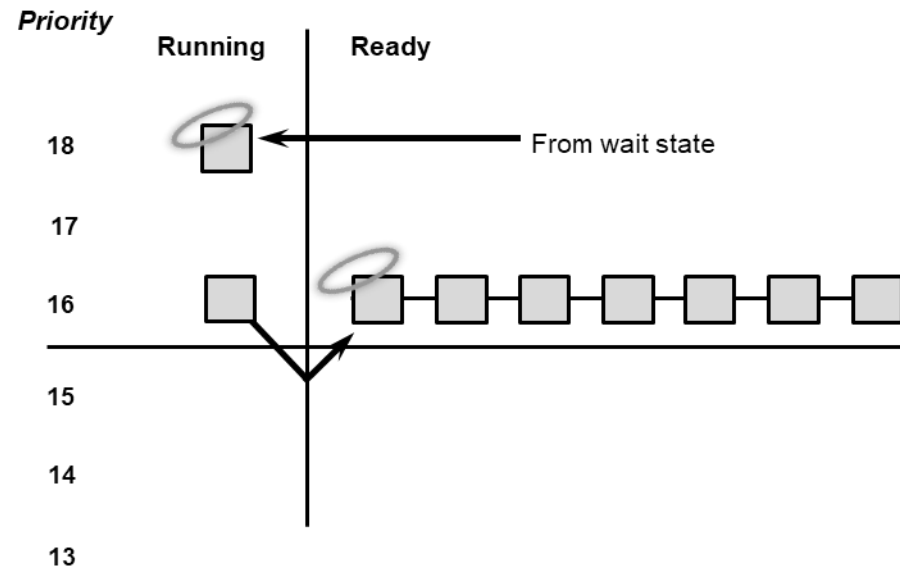


# Planificación: escenarios (2/3)

## Windows 2000

### ► Expulsión:

- Un hilo T de menor prioridad es expulsado cuando otro de mayor prioridad se vuelve listo para ejecutar
- T se pone a la cabeza de la cola de su prioridad
- Rodaja de T: si RT entonces se reinicia en caso contrario se mantiene

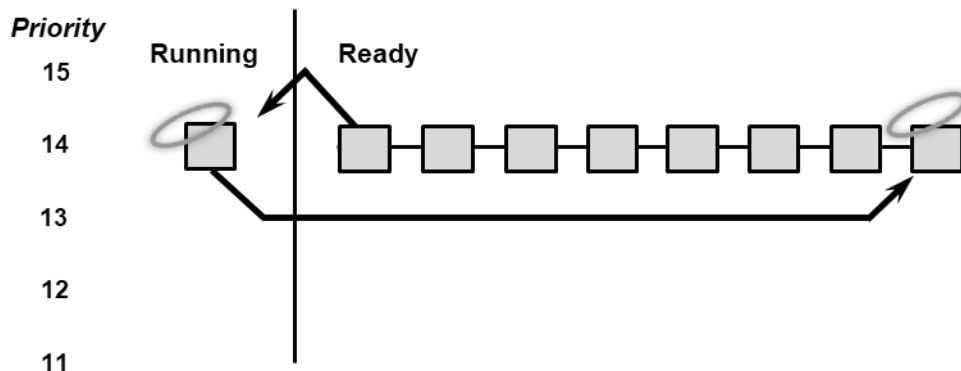


# Planificación: escenarios (3/3)

## Windows 2000

### ► Fin de rodaja:

- Un hilo T agota su rodaja de tiempo (quantum)
- Acciones del planificador:
  - Reducir la prioridad de T → otro hilo pasa a ejecutar
  - No reducir la prioridad → T pasa al último de la cola de su nivel (si vacía, vuelve de nuevo)
- Rodaja de T: se reinicia

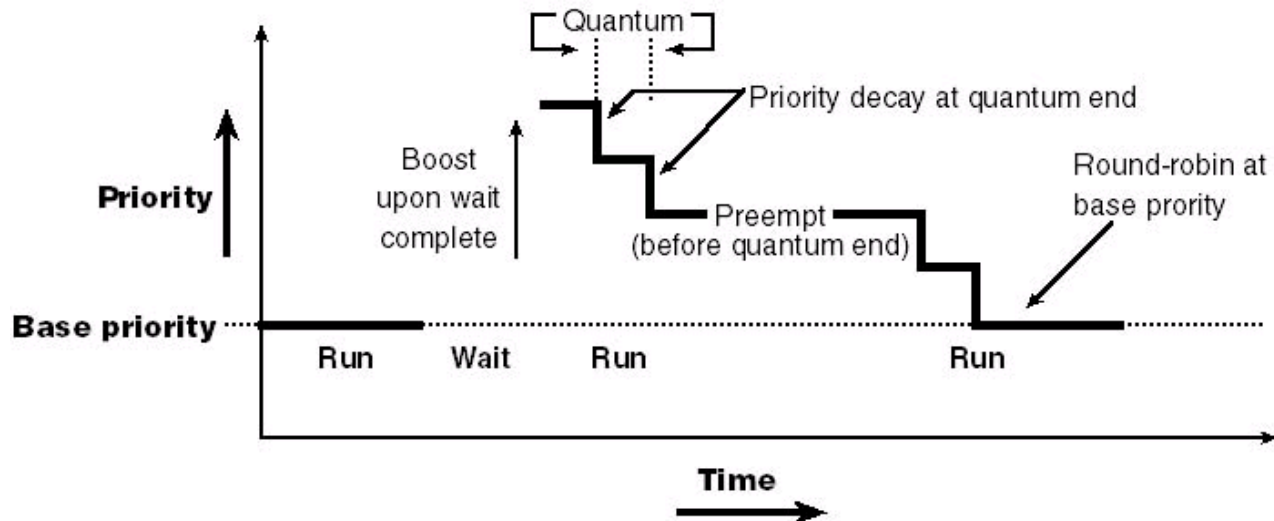


# Planificación: aumento de prioridad

## Windows 2000

### ► *Priority boost:*

- Se aumenta la prioridad en ciertas ocasiones (solo en los niveles 0-15):
  - Cuando se completa una operación de E/S
  - Al salir del estado de una operación *wait*
  - Cuando el hilo lleva «mucho tiempo» en la cola de listo sin ejecutar:
    - El hilo de *kernel balance set manager* aumenta la prioridad por «envejecimiento»
    - Muestra 1 vez por segundo la cola de listos y si  $T.\text{estado}=\text{READY}$  más de 300 ticks (~3 ó 4 segundos) entonces  
 $T.\text{prioridad} = 15$   
 $T.\text{rodaja} = 2 * \text{rodaja\_normal}$



# Planificación Windows

## resumen

---

### □ Principales características:

- ▣ Basado en prioridades y uso de cuantos de tiempo.
- ▣ Planificación apropiativa.
- ▣ Planificación con afinidad de procesador.

### □ Planificación por hilos y no por procesos.

### □ Un hilo puede perder el procesador si hay otro más prioritario que esté listo.

### □ Decisiones de planificación:

- ▣ Hilos nuevos → Listo.
- ▣ Hilos bloqueados que reciben evento → Listo.
- ▣ Hilo deja del procesador si termina cuanto, finaliza o pasa a bloqueado.

# Contenidos

---

1. Conceptos básicos de planificación de sistemas operativos.
2. Planificación y activación
3. Algoritmos de programación más comunes
  - ▶ FIFO, SJF, RR y PRIORIDAD.
4. Estructuras de datos de planificación en el núcleo.
  - ▶ Planificación en LINUX: envejecimiento.
5. **Llamadas de planificación de procesos.**

# Llamadas de planificación de procesos.

---

- ▶ `sched_setscheduler/sched_getscheduler`
  - ▶ Set/return the scheduling policy and parameters of a specified thread.
- ▶ `sched_setparam/sched_getparam`
  - ▶ Set/Fetch the scheduling parameters of a specified thread.
- ▶ `sched_get_priority_max/sched_get_priority_min`
  - ▶ Return the maximum/minimum priority available in a specified scheduling policy.
- ▶ `sched_rr_get_interval`
  - ▶ Fetch the quantum used for threads that are scheduled under the "round-robin"
- ▶ `sched_yield`
  - ▶ Cause the caller to relinquish the CPU, so that some other thread be executed.
- ▶ `sched_setaffinity/sched_getaffinity`
  - ▶ (Linux-specific) Set/Get the CPU affinity of a specified thread.
- ▶ `sched_setaattr/sched_getattr`
  - ▶ Set/Fetch the scheduling policy and parameters of a specified thread. This (Linux-specific) system call provides a superset of the functionality of `sched_set/scheduler` and `sched_set/getparam`.

# Lección 3

## Planificación de procesos

Sistemas Operativos  
Ingeniería Informática