Tema 2

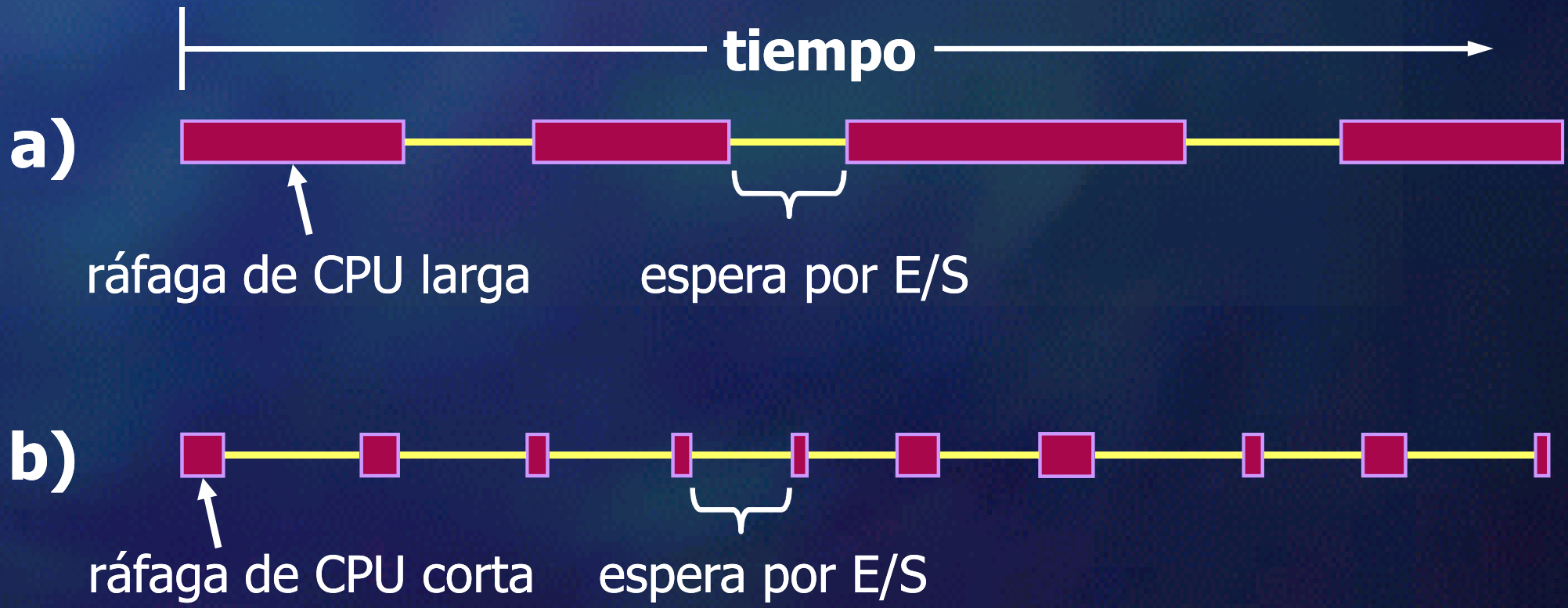
Planificación del Procesador

Contenidos

- Conceptos básicos
  - Multiprogramación. Ráfagas de CPU y de E/S
  - Niveles de planificación
- Criterios y objetivos de la planificación
- Algoritmos de planificación
  - Planificación expropiativa y no expropiativa
  - Orden de llegada, tiempo de ejecución, prioridades, ...
  - Planificación multinivel
- Ejemplos reales de planificación
  - Windows NT/XP
  - Linux

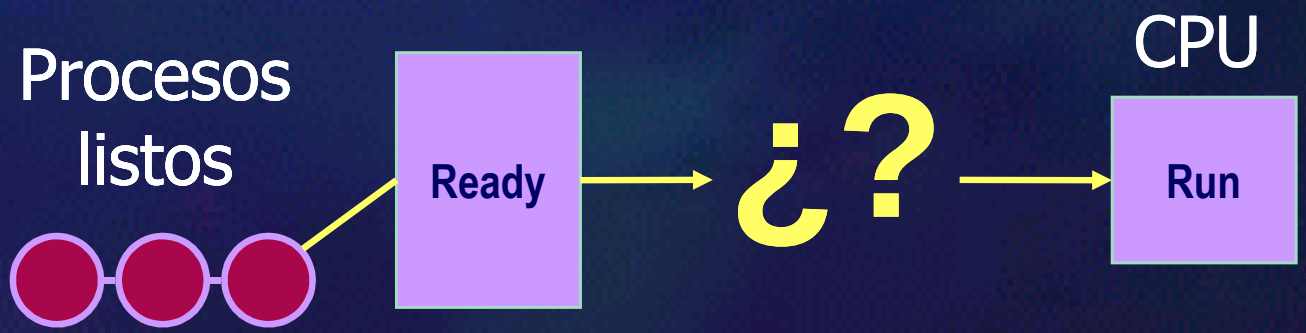
Ciclos de CPU – Entrada/Salida

- Procesos alternan fases de cómputo con esperas por E/S
- Clasificación de procesos
  a) Intensivos en CPU (ráfagas de CPU largas)
  b) Intensivos en E/S (ráfagas de CPU cortas)



Necesidad de planificar

- Con sólo un proceso, la CPU permanecería ociosa en las esperas por entrada/salida
- Multiprogramación
  - Permite ejecutar varios procesos concurrentemente
  - Aprovechar al máximo la CPU
- **Función del planificador**
  - Cada vez que la CPU esté libre:
    decidir qué proceso accede a ella



¿qué es planificar?

- Una política de planificación responde a:
  a) Cuándo y en qué condiciones expulsamos a un proceso en ejecución
     - Preemption (adelantamiento, expropiación)
  b) Qué proceso de la cola listos despachamos
     - Scheduler (planificador)
- Resumiendo:
  - La planificación son las políticas y mecanismos que deciden el orden en que se completan los procesos.

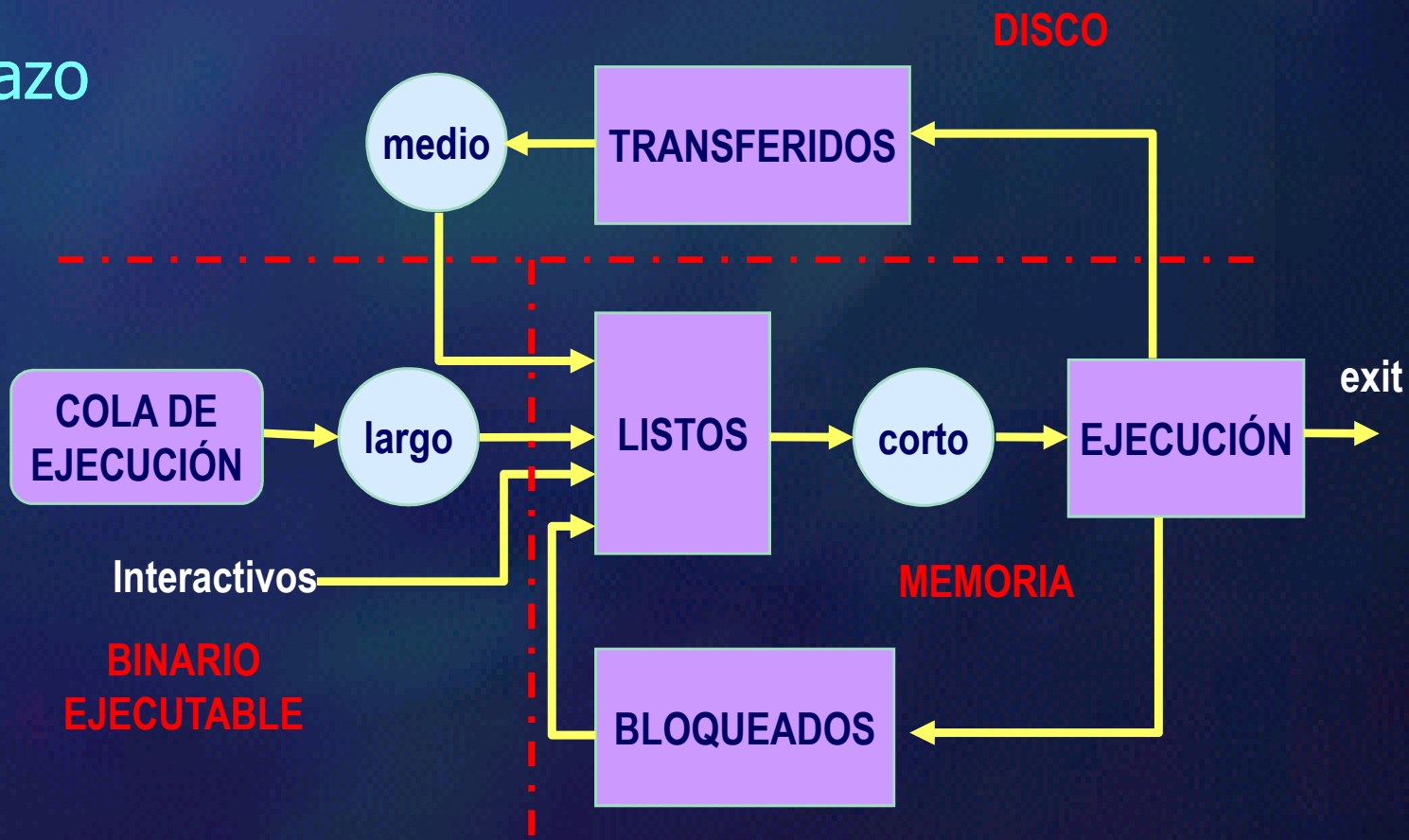
Criterios generales del planificador

- Objetivo del planificador
  - Maximizar las prestaciones del computador según ciertos criterios

- Criterios generales
  - Maximizar el rendimiento
    - Uso de la CPU, n° de trabajos terminados por u. tiempo,....
  - Ser estadísticamente predecible
    - Posibilidad de estimar cuanto va a tardar un trabajo
  - Ser imparcial
    - No debe discriminar injustamente a unos proc. sobre otros
  - Aprovechar los recursos
    - Mantener en uso los distintos subsistemas del HW

Niveles de planificación

- Tipos de planificadores
  - Largo plazo
  - Medio plazo
  - Corto plazo



Niveles de planificación

- Largo plazo
  - Poca frecuencia de ejecución
  - Para procesos por lotes (batch o jobs)
  - Lanza programas a ejecución (crea procesos)
  - Objetivos
    - Mantener constante el grado de multiprogramación
      - Si acaba un proceso, lanza otro
    - Mantener una buena mezcla de procesos
      - Tantos procesos intensivos en CPU como intensivos en E/S
  - No siempre existe un planificador de este tipo
- Los interactivos pasan direct. a *Listos*
  - Filtrado de otra naturaleza
    - Por el número de terminales disponible (limita los usuarios)
    - Usuarios que abandonan si el sistema está muy cargado

Niveles de planificación

- Medio plazo
  - Puede controlar
    - La ocupación de memoria (si hay poca, descarga procesos)
    - La carga del sistema (si es mucha, suspende procesos)
    - Puede mejorar la mezcla de procesos
  - No es relevante en SO modernos con Mem. Virtual
    - Los procesos que no referencian sus páginas, las van perdiendo porque otros procesos las reemplazan
- Corto plazo
  - Mantiene ocupada la CPU
  - Se invoca muy frecuentemente
    - Por tanto no debe consumir mucho tiempo de CPU
    - Ejemplo: Quantum de 10ms. Planificador consume 1ms
      - 9% del tiempo de CPU ocupado por el planificador

Contenidos

- Conceptos básicos
  - Ráfagas de CPU y de E/S. Multiprogramación
  - Niveles de planificación
- Criterios y objetivos de la planificación
- Algoritmos de planificación
  - Planificación expropiativa y no expropiativa
  - Orden de llegada, tiempo de ejecución, prioridades, ...
  - Planificación multinivel
- Ejemplos reales de planificación
  - Windows NT/XP
  - Linux

Criterios cuantitativos

- El planificador debe optimizar valores objetivos y cuantitativos
  - Utilización de la CPU: 0%-100%
    - Porcentaje de tiempo con la CPU ocupada
    - Descontar el tiempo de CPU consumido por el SO
  - Productividad (throughput): procesos/seg.
    - Trabajos terminados por unidad de tiempo
  - Tiempo de retorno (t. real o elapsed): Tr
    - Tiempo desde que lanzas un programa hasta que termina.
  - Tiempo de espera: Te
    - Parte de Tret en que el proceso está esperando en Ready
  - Tiempo de respuesta: Ta
    - En sist. multitarea mide la interactividad: tiempo desde que se lanza un proceso y éste responde al usuario.

Ejemplo y optimización de criterios

- Ejemplo: `time ls -lR /var`
  - El comando `time` devuelve inf. temp. del `ls`

```
0,17s user, 0,22s system, 7,84s real (4%)
```

    - Tr: desde que empieza hasta que termina 7.84s
    - Tproceso: 0.17 + 0.22 = 0.39s
      - 0.17s en ejecución del proceso (modo user)
      - 0.22s en ejec. de syscalls llamadas por el proceso (núcleo)
- Optimizar criterios
  - Maximizar la utilización y productividad
  - Minimizar los tiempos (retorno, espera, resp.)
  - En general se max. o min. valores medios
  - A veces se optimizan valores max. o min.
    - Ejemplo: minimizar el valor max. del tiempo de respuesta

Diferentes entornos = diferentes objetivos

- Política de planificación
  - Conjunto de estrategias y decisiones tomadas para diseñar la planificación y conseguir los objetivos propuestos
- Sistemas por lotes
  - Capacidad de procesamiento / rendimiento
  - Tiempo de retorno
  - Utilización de la CPU
- Sistemas interactivos
  - Tiempos de respuesta (varianza)
  - Proporcionalidad complejidad/tiempo respuesta
- Sistemas en tiempo real
  - Cumplir tiempos límite de servicio
  - Predictibilidad o consistencia

Contenidos

- Conceptos básicos
  - Ráfagas de CPU y de E/S. Multiprogramación
  - Niveles de planificación
- Criterios y objetivos de la planificación
- Algoritmos de planificación
  - Planificación expropiativa y no expropiativa
  - Orden de llegada, tiempo de ejecución, prioridades, ...
  - Planificación multinivel
- Ejemplos reales de planificación
  - Windows NT/XP
  - Linux

¿Donde se planifica? (preemption)

- Situaciones DONDE planificar:
  1. Siempre que el proceso en ejecución abandona la CPU voluntariamente
     - el proceso en ejecución se bloquea (fin ráfaga CPU)
     - el proceso en ejecución termina

     Planificación sólo en (1) $\Rightarrow$ **no expropiativa (no expulsiva)**
  2. Otras situaciones donde el proceso no abandona voluntariamente la CPU
     - Cuando llega otro proceso a la cola de listos (nuevo o procedente de un bloqueo)
     - Mediante una interrupción programada cada cierto tiempo para lanzar el planificador

     Planificación también en (2) $\Rightarrow$ **expropiativa (expulsiva)**

¿Cuál es el siguiente proceso a ejecutar? (scheduler)

- CÓMO se elige el siguiente proceso a ejecutar
  1. Por orden de llegada
  2. Por tiempo de ejecución
  3. Por prioridades
  4. Mezcla de las anteriores

- Algoritmos de planificación
  - Distintas combinaciones de "preemption" y "scheduler"

Algoritmos de planificación

- Algoritmos de planificación:

	No expropiativa	Expropiativa
por orden de llegada	FCFS	Turno circular (RR)
por tiempo de ejecución	Trabajo más corto primero (SJF)	Tiempo restante más corto primero (SRTN)
por prioridades	Prioridades	Prioridades expropiativas

- Planificación con colas multinivel

FCFS: Por orden de llegada (First-Come, First-Served)

- Características:
  - No expropiativo
    - El proceso en ejecución no abandona la CPU hasta que finaliza o se bloquea.
  - Pasa a ejecución el proceso que más tiempo lleva en la cola de listos
  - Beneficia a los procesos que requieren mucho procesador frente a los intensivos en E/S
  - Te y Tr elevados

FCFS: Por orden de llegada (First-Come, First-Served)

Un ejemplo:

Proceso	Ráfaga de CPU
A	2
B	5
C	2

0 2 7 9

Tiempo medio de espera : (0 + 2 + 7) / 3 = **3**

Tiempo medio de retorno : (2 + 7 + 9) / 3 = **6**

FCFS: Por orden de llegada (First-Come, First-Served)

¡ procesos grandes provocan retenciones !

Proceso	Ráfaga de CPU
B	5
A	2
C	2

Tiempo medio de espera : (5 + 0 + 7) / 3 = **4** (3)

Tiempo medio de retorno : (7 + 5 + 9) / 3 = **7** (6)

Ejemplos de planificación

Proceso	Llegada	Tiempo CPU	Prioridad
T0	0	50	4
T1	10	50	3
T2	20	20	1
T3	30	10	2

- El proceso T0 se bloquea a las 10 unidades de tiempo durante 15 unidades de tiempo

	No expropiativa	Expropiativa
por orden de llegada	FCFS	Turno circular (RR) **Q=10**
por tiempo de ejecución	Trabajo más corto primero (SJF)	Tiempo restante más corto primero (SRTN)
por prioridades	Prioridades	Prioridades expropiativas

SJF: El trabajo más corto primero (Shortest Job First)

- Características:
  - No expropiativo
    - El proceso en ejecución no abandona la CPU hasta que finaliza o se bloquea.
  - Pasa a ejecución el proceso más corto de los listos
    - Se ordena la cola de listos de menor a mayor tiempo de ejecución para la siguiente ráfaga de procesador.
  - Evita largos tiempos de espera en los sistemas interactivos y con E/S
    - Minimiza Te
  - Penaliza trabajos con mayor tiempo de ejecución
    - Intensivos en CPU

SJF: El trabajo más corto primero (Shortest Job First)

Proceso	Ráfaga de CPU
A	2
B	5
C	2

Tiempo medio de espera : (0 + 4 + 2) / 3 = **2**

Tiempo medio de retorno : (2 + 9 + 4) / 3 = **5**

- Resultado: tiempo medio de espera **óptimo**
- ¿Predecir el tiempo de ejecución?

SJF: Predecir el tiempo de ejecución

- Estimación del tiempo de ráfaga de CPU
  - Promedio exponencial

Estimación ráfaga n+1 | Tiempo ráfaga n | Estimación ráfaga n

$$\tau_{n+1} = \alpha \cdot t_n + (1-\alpha) \cdot \tau_n \qquad 0 \le \alpha \le 1$$

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

  - Es decir, el segundo término contiene un histórico
  - Cuanto más antigua es la ráfaga menos pesa su duración
  - Si $\alpha=0$ no consideras el último tiempo de ráfaga
  - Si $\alpha=1$ no consideras la historia acumulada
  - Típicamente $\alpha=1/2$

Ejemplos de planificación

Proceso	Llegada	Tiempo CPU	Prioridad
T0	0	50	4
T1	10	50	3
T2	20	20	1
T3	30	10	2

- El proceso T0 se bloquea a las 10 unidades de tiempo durante 15 unidades de tiempo

	No expropiativa	Expropiativa
por orden de llegada	FCFS	Turno circular (RR) **Q=10**
por tiempo de ejecución	Trabajo más corto primero (SJF)	Tiempo restante más corto primero (SRTN)
por prioridades	Prioridades	Prioridades expropiativas

RR: Turno circular (Round Robin)

- Características:
  - Expropiativo
    - El proceso en ejecución puede abandonar la CPU cuando finaliza, se bloquea o agota su cuanto (quantum Q) de tiempo (expropiación)
  - Pasa a ejecución el proceso que más tiempo lleva en la cola de listos
    - Versión expropiativa del FCFS
  - Aumenta la interactividad haciendo un reparto equitativo del procesador
  - Aumenta el número de cambios de contexto
    - Sobrecarga CPU

RR: Turno circular (Round Robin)

Proceso	Ráfaga CPU
A	2
B	5
C	2

Cuanto Q =1

0 4 6 9

Tiempo de retorno medio: (4 + 6 + 9) / 3 = **6.33**

Más alto que SJF (5), pero **mejor tiempo de respuesta**

RR: Turno circular (Round Robin)

- Si el grado de multiprogramación es N
  - Cada proceso recibe 1/N del tiempo de CPU
  - Trespuesta ≤ N x Q
    - No garantiza un Tresp muy pequeño para un proceso, pero si minimiza la desviación de Tresp y establece una cota superior
    - Propiedad muy útil en sistemas interactivos: un retardo esporádico no es grave y los usuarios tienen sensación de interactividad
- Decidir tamaño de Quanto
  - Si Q↑ el sistema degenera en FCFS
  - Si Q↓ demasiados cambios de contexto
  - Regla empírica: elegir Q de forma que:
    - El 80% de las ráfagas de CPU sean menores que Q

Ejemplos de planificación

Proceso	Llegada	Tiempo CPU	Prioridad
T0	0	50	4
T1	10	50	3
T2	20	20	1
T3	30	10	2

- El proceso T0 se bloquea a las 10 unidades de tiempo durante 15 unidades de tiempo

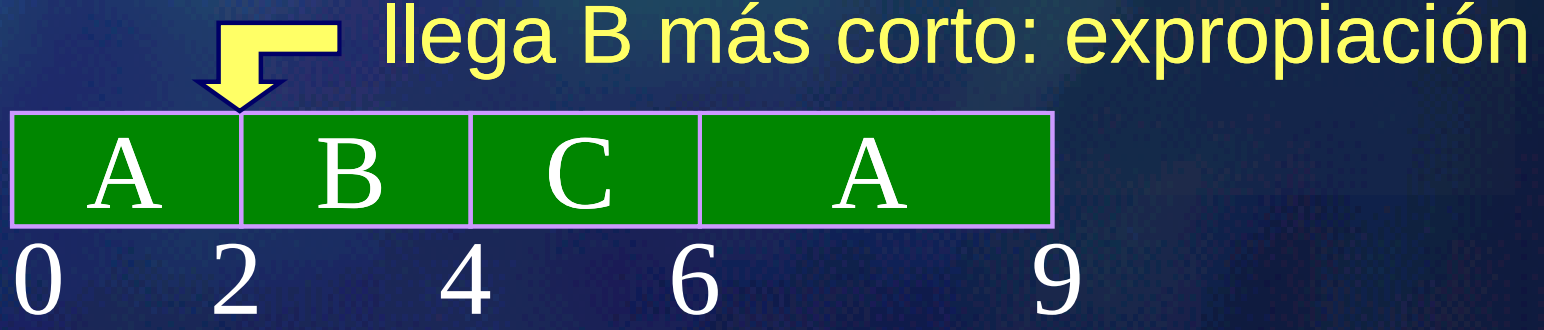
	No expropiativa	Expropiativa
por orden de llegada	FCFS	Turno circular (RR) **Q=10**
por tiempo de ejecución	Trabajo más corto primero (SJF)	Tiempo restante más corto primero (SRTN)
por prioridades	Prioridades	Prioridades expropiativas

SRTN: Tiempo restante más corto primero (Shortest Remaining Time Next)

- Características:
  - Expropiativo
    - El proceso en ejecución puede abandonar la CPU cuando finaliza, se bloquea o entra un proceso con tiempo de próxima ráfaga de CPU menor que el tiempo restante a la ráfaga actual del que se está ejecutando
  - Pasa a ejecución el proceso más corto de los listos
    - Se ordena la cola de listos de menor a mayor tiempo de ejecución para la siguiente ráfaga de procesador.
  - Versión expropiativa del SJF

SRTN: Tiempo restante más corto primero (Shortest Remaining Time Next)

Proceso	Ráfaga de CPU	T. de llegada
A	5	0
B	2	2
C	2	3



Tiempo de espera medio: (4 + 0 + 1) / 3 = 1.66

SJF = (0 + 3 + 4) /3 = 2.33

Ejemplos de planificación

Proceso	Llegada	Tiempo CPU	Prioridad
T0	0	50	4
T1	10	50	3
T2	20	20	1
T3	30	10	2

- El proceso T0 se bloquea a las 10 unidades de tiempo durante 15 unidades de tiempo

	No expropiativa	Expropiativa
por orden de llegada	FCFS	Turno circular (RR) **Q=10**
por tiempo de ejecución	Trabajo más corto primero (SJF)	Tiempo restante más corto primero (SRTN)
por prioridades	Prioridades	Prioridades expropiativas

Planificación por prioridades

- Se asigna un valor numérico (prioridad) a cada proceso
- Se concede la CPU al proceso de mayor prioridad
  - normalmente menor valor ≡ mayor prioridad
  - Expropiativa (cuando llega un proceso más prioritario a la cola de listos) o no expropiativa (sólo cuando termina o se bloquea el proceso en ejecución)
- Procesos con igual prioridad: usar otro algoritmo para decidir cual elegir (FCFS, SJF)
- SJF es un caso particular donde la prioridad es el tiempo previsto de ejecución
- Problema: inanición o postergación indefinida
  - puede que procesos de baja prioridad nunca alcancen la CPU
- Solución: envejecimiento (aging)
  - incrementar la prioridad de los procesos con el avance del tiempo

Planificación por prioridades (ejemplo)

- No es imparcial o equitativa
- Los tiempos medios se sacrifican en favor de los procesos de más prioridad

Proceso	Ráfaga de CPU	Prioridad
A	2	3
B	5	1
C	2	2

0 5 7 9

Tiempo de espera medio: (7 + 0 + 5) / 3 = **4**

Tiempo de retorno medio: (9 + 5 + 7) / 3 = **7**

Ejemplos de planificación

Proceso	Llegada	Tiempo CPU	Prioridad
T0	0	50	4
T1	10	50	3
T2	20	20	1
T3	30	10	2

- El proceso T0 se bloquea a las 10 unidades de tiempo durante 15 unidades de tiempo

	No expropiativa	Expropiativa
por orden de llegada	FCFS	Turno circular (RR) **Q=10**
por tiempo de ejecución	Trabajo más corto primero (SJF)	Tiempo restante más corto primero (SRTN)
por prioridades	Prioridades	Prioridades expropiativas

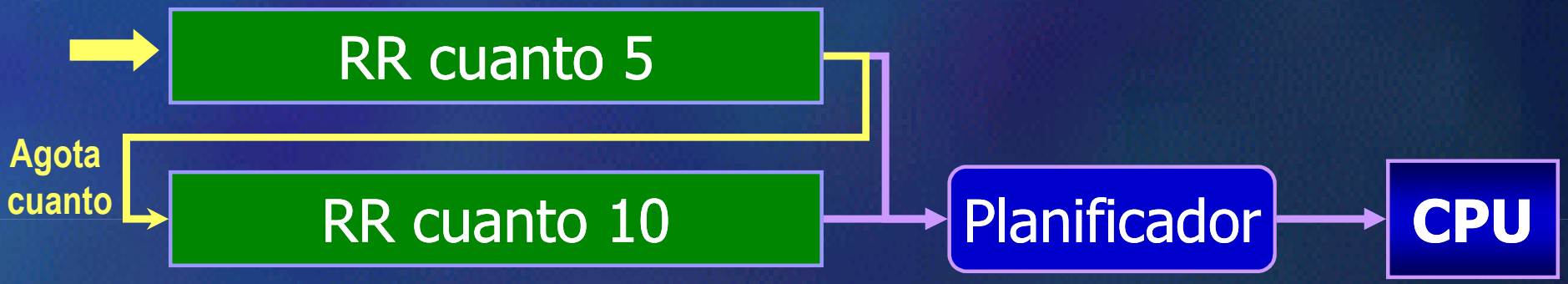
Colas multinivel

- Sistemas reales son heterogéneos
- Varias colas de listos (categorías y políticas específicas)
  - Se planifican las colas por prioridad
    - Problema: inanición o postergación indefinida
  - Repartir CPU entre colas
    - sistema 70%; interactivos 20 %; ...

Colas multinivel realimentadas

- Multinivel pero más flexible
  - no necesitamos clasificar a priori, permite a los procesos migrar de una cola a otra
  - política de cambio de cola
    - Ejemplo: Si no se agota cuanto, promocionar a cola superior y si se agota a inferior
  - se puede evitar inanición
    - promocionando a colas de más prioridad procesos que esperan demasiado tiempo (envejecimiento - *aging*)
  - premia a los interactivos (intensivos en E/S)

Ejemplo colas multinivel con realimentación



	Llegada	CPU	E/S	CPU	Tin	Tout	Te	Tr
P1	0	8	X	X				
P2	2	4	6	2				
P3	4	6	X	X				
P4	7	3	8	3				
P5	9	2	X	X				

SJF

Estimación ráfaga n+1 | Tiempo ráfaga n | Estimación ráfaga n

$$\tau_{n+1} = \alpha \cdot t_n + (1-\alpha) \cdot \tau_n \qquad 0 \le \alpha \le 1$$

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

Proceso	CPU	CPU	CPU	CPU	CPU
T0	5	4	3	2	1
T1	1	2	3	4	5
T2	1	1	1	1	1

- Ráfagas de E/S siempre de 1
- α= 0.5

FSO - Examen septiembre de 2006

8.- En un sistema con planificación de colas de realimentación multinivel de tres niveles se ejecutan tres procesos.

Los tiempos de computación pura de los procesos (total CPU) son los siguientes: P1=60 ciclos, P2=130 ciclos y P3=180 ciclos. Además el proceso P1 realiza una operación de E/S cuando han pasado 20 ciclos de su tiempo de computación. El proceso P2 realiza una operación de E/S cuando han pasado 80 ciclos de su tiempo de computación. El proceso P3 no realiza operaciones de E/S. Cada operación de E/S dura 20 ciclos.

Los cuantos de tiempo asociados a cada cola, a medida que desciende la prioridad de las colas son Cola1: 50, Cola2: 100 y Cola3: 120 ciclos (la Cola1 es la de mayor prioridad). Suponiendo que, inicialmente, todos los procesos llegan la mismo tiempo al sistema de planificación y que el orden inicial en la cola es P1(primero), P2(segundo) y P3 (tercero), completar el diario de ejecución en la tabla siguiente anotando el tiempo en el cual ocurre cada nuevo evento, dónde se encuentra cada proceso en ese momento y la descripción del evento.

Tiempo	Ejecución	Cola1	Cola2	Cola3	Bloqueados	Descripción evento
0	P1	P2,P3				Llegada de los procesos P1,P2, P3 Despacho de P1

Colas multinivel expropiativo

Se ejecutan cuatro procesos en un sistema monoprocesador con planificación de colas multinivel con realimentación de tres niveles .

En la siguiente tabla se muestran los instantes de llegada y las duraciones de las sucesivas ráfagas de CPU y E/S de los procesos. El tiempo avanza de izquierda a derecha.

	Llegada	CPU	E/S	CPU	E/S	CPU
P1	0	3	1	3	1	3
P2	1	5	1	6		
P3	2	10				
P4	4	1	20	6		

La Cola1, es la de mayor prioridad, tratándose de **prioridades expropiativas**. Los cuantos de tiempo asociados a cada cola, se hacen mayores a medida que desciende la prioridad de las colas y son:

	Cuanto
Cola1	2
Cola2	5
Cola3	10

Las reglas de movimiento de procesos entre colas son las siguientes:

- La cola inicial para cualquier proceso es la Cola 1.
- Cuando un proceso agota su cuanto baja de nivel.
- Cuando un proceso no agota su cuanto asciende de nivel.
- Cuando un proceso se despierta del bloqueo y pasa a "listos" obtendrá un cuanto renovado (completo).
- Cuando un proceso es expropiado vuelve al final de la cola desde la que fué despachado, recuperando el valor de cuanto asociado a la misma.

En caso de que en el mismo instante varios procesos lleguen a la vez a la misma cola el orden será:

1. Proceso nuevo
2. Desbloqueo (si son varios los que se despiertan en el mismo instante y van a parar a la misma cola, lo harán en el orden en que se bloquearon)
3. Proceso en CPU que termina su quantum
4. Proceso en CPU expropiado

Contenidos

- Conceptos básicos
  - Ráfagas de CPU y de E/S. Multiprogramación
  - Niveles de planificación
- Criterios y objetivos de la planificación
- Algoritmos de planificación
  - Planificación expropiativa y no expropiativa
  - Orden de llegada, tiempo de ejecución, prioridades, ...
  - Planificación multinivel
- Ejemplos reales de planificación
  - Windows NT/XP
  - Linux

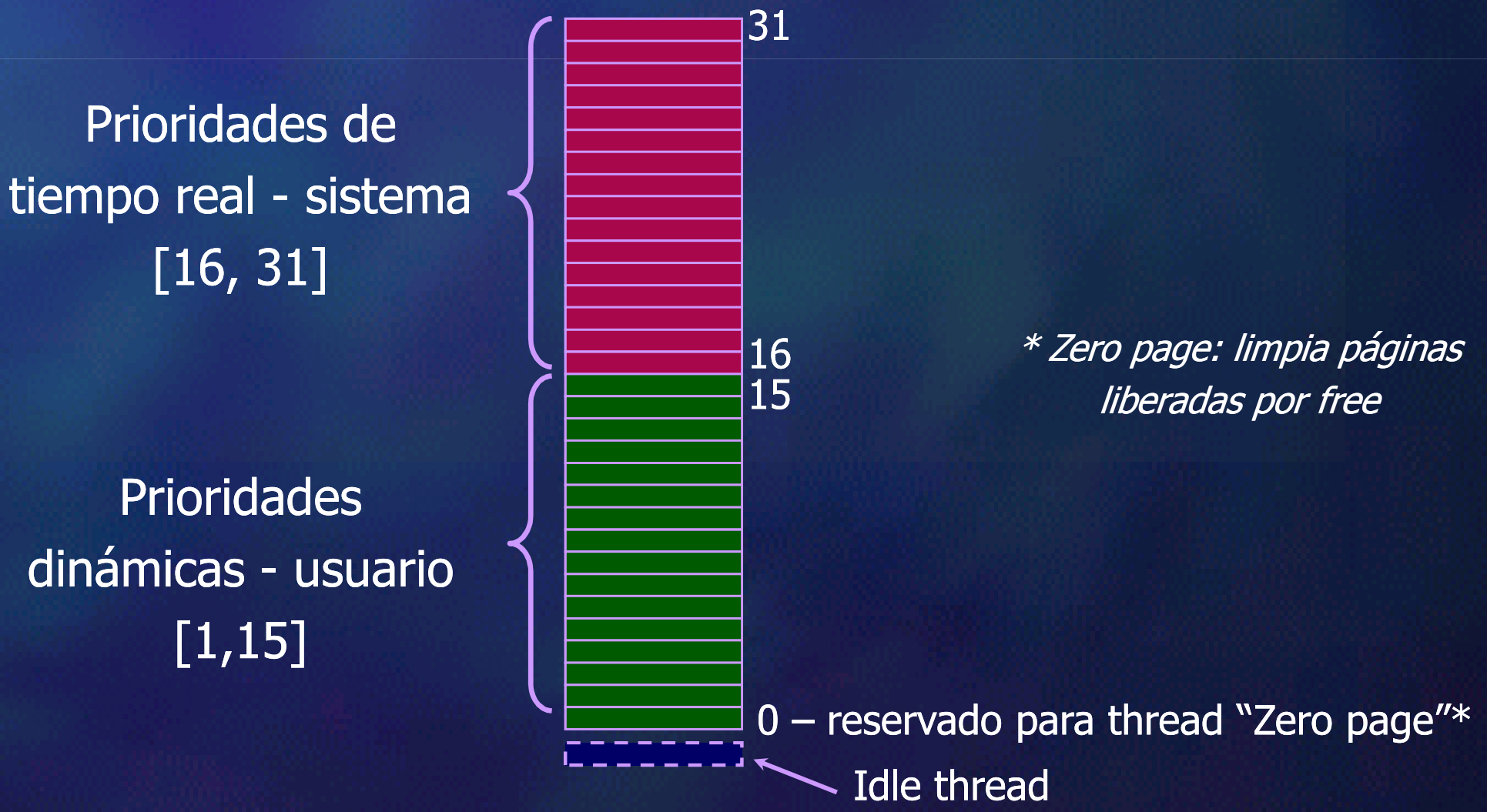
Planificación Windows NT/XP

Planificación Windows NT/XP

- Política de planificación
  - **Planifica threads**, independientemente de los procesos
  - Planificación expropiativa en base a prioridades (dinámicas para usuario – estáticas TR)
  - Tiempo compartido mediante cuantos de tiempo
  - Procesos en tiempo real: tienen mayor prioridad
  - Favorece threads interactivos incrementando su prioridad
  - Determinación del cuanto de tiempo
    - 2 ticks workstation – 12 ticks server (20 – 120 ms)
    - Se puede hacer el cuanto más largo para el thread/ventana en primer plano (interactiva) o para los threads en background (3×)

Planificación Windows NT/XP

- Se planifica por prioridades
- 32 prioridades soportadas por el kernel [0,31]



Planificación Windows NT/XP

- Asignación de prioridades Win32 (API estándar)
  - Cada proceso se asigna a una prioridad base
    - Realtime (24)
    - High (13)
    - Above Normal (10)
    - Normal (8)
    - Below Normal (6)
    - Idle (4)
  - Cada thread tiene una prioridad relativa (±2)
    - Highest (+2)
    - Above normal (+1)
    - Normal (+0)
    - Below Normal (-1)
    - Lowest (-2)

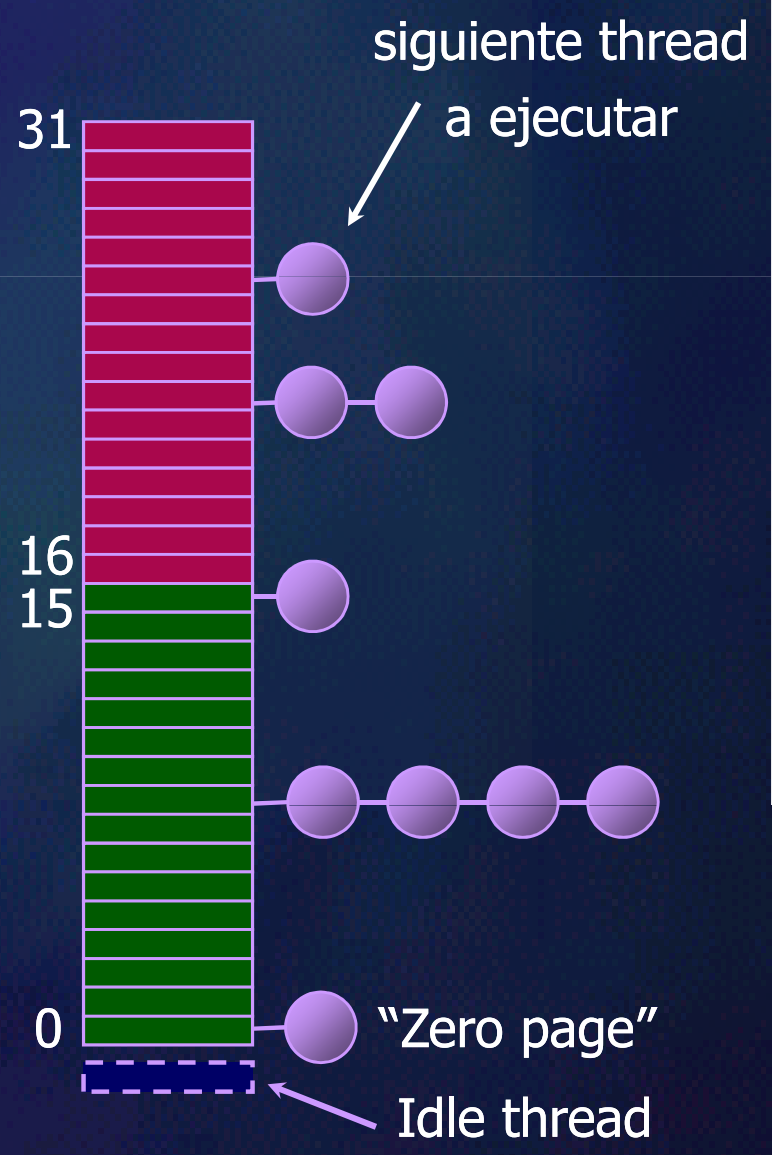
Planificación Windows NT/XP

- Correspondencia prioridades Win32 - kernel

		Win32 process class priorities					
		Realtime	High	Above Normal	Normal	Below Normal	Idle
Win32 thread priorities	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

Algoritmo de planificación Windows NT/XP

- El planificador mantiene 32 colas para threads listos
  - Una cola por cada prioridad
- Se escoge el primer thread listo de la cola de prioridad más alta no vacía
- Tiempo compartido en turno circular (RR) dentro de un mismo nivel de prioridad
- El thread expropiado
  - **Por fin de cuanto** se coloca al final de la cola de su prioridad (RR)
  - **Por un thread de más prioridad** se coloca en la cabeza de la cola de su prioridad (conserva el turno)



Algoritmo de planificación Windows NT/XP

- Ajustes automáticos de las prioridades
  - Los threads de usuario (prioridades dinámicas < 16) pueden experimentar incrementos (boosts) y reducciones (decays) automáticos en su prioridad
  - Para favorecer procesos intensivos en E/S y para evitar inanición
  - Los threads de tiempo real (prioridad > 15) no experimentan estos ajustes
    - La planificación en TR es predecible: respeta la prioridad original de cada thread. Pero esto NO garantiza tiempos de respuesta.

Algoritmo de planificación Windows NT/XP

- Incrementos en la prioridad de un thread
  - Tienen lugar tras un bloqueo, normalmente cuando se resuelve una petición de E/S
    - Dispositivos lentos = grandes incrementos (p.e. KB + 6)
    - Dispositivos rápidos = pequeños incrementos (p.e. HD + 1)
  - El incremento se aplica sobre su prioridad base
    - El resultado nunca es mayor que 15 (máx. para usuarios)
  - Buen tiempo de respuesta para interactivos, dispositivos de E/S se mantienen ocupados
- Decrementos en la prioridad
  - Después de un incremento por resolución de bloqueo
  - Cada vez que consuma un cuanto de ejecución, se decrementa en 1 la prioridad. Hasta alcanzar la prioridad base del thread

Algoritmo de planificación Windows NT/XP

- Eliminación de la inanición
  - "Balance Set Manager" es un thread con prioridad 16 (de sistema) que se despierta cada segundo
  - Busca threads de usuario que han estado listos 4 ó más segundos sin ejecutar
  - Les incrementa la prioridad y el cuanto
    - Prioridad = 15 (máxima para usuario)
    - Doble cuanto de tiempo
    - hasta 10 threads a la vez
  - El incremento es transitorio: sólo hasta completar el cuanto, después vuelve a su prioridad y cuanto normal

Planificación Windows: limitaciones

- Incrementos automáticos de prioridad algo arbitrarios
  - Favorece los threads que usan dispositivos de E/S
    - Los incrementos están prefijados
    - La tarjeta de sonido tiene máximo incremento +8
  - Si queremos ir rápido hay que tocar mucha música
- El sistema no evita implícitamente la inanición
  - No existe envejecimiento, es necesario vigilar activamente que no ocurra postergación indefinida
- Soporte para tiempo real soft
  - No se garantizan tiempos de respuesta
  - Sólo que la planificación es predecible (prio. estáticas)

Planificación Linux

Planificación Linux

- Objetivos Unix (¿incompatibles?)
  - bajos tiempos de respuesta, buena productividad (trhoughput) para segundo plano, evitar inanición de procesos, conciliar las necesidades de procesos de alta y baja prioridad...
- Política de planificación
  - **Planifica threads**
  - Planificación expropiativa en base a **prioridades** (dinámicas para procesos normales – estáticas TR)
  - Tiempo compartido mediante cuantos de tiempo
  - Procesos en tiempo real: tienen mayor prioridad y planificación diferenciada
  - Favorece procesos interactivos incrementando su prioridad

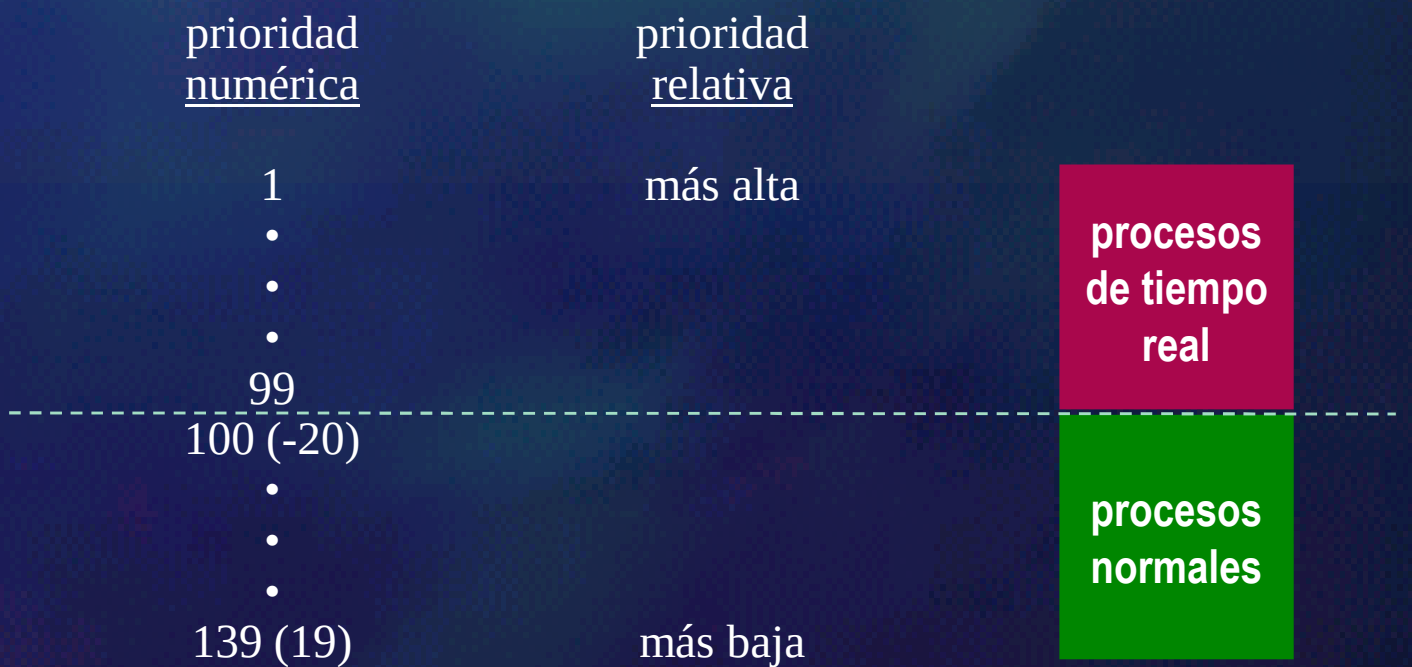
Planificación de **tiempo real** en Linux

- Tiempo real se panifica antes que los procesos normales
- Planificación por prioridad expropiativa
  - Siempre se escoge el proceso TR de prioridad más alta; si hay varios iguales: el primero de la cola
  - prioridades estáticas 1..99
- Dos clases de planificación en tiempo real para procesos con igual prioridad
  - FIFO – por orden de llegada (SCHED_FIFO)
    - Se ejecuta hasta que termine (cuanto ilimitado)
    - Sólo es expropiable por procesos TR de más prioridad
  - Round Robin – turno circular (SCHED_RR)
    - Es desalojado si agota su cuanto
    - Vuelve al final de la cola de listos (misma prioridad)
    - El tiempo se comparte en procesos TR con la misma prioridad

Algoritmo de planificación Linux: O(1)

- Desde kernel 2.6
  - Algoritmo de planificación de tiempo constante
- Colas multinivel
  - Procesos de tiempo real (sin realimentación)
  - Procesos normales (con realimentación)
- Dos clases de procesos – dos políticas
  - Procesos de tiempo real
    - **Siempre se despachan antes** que los procesos normales
    - Prioridades estáticas
    - No equitativo, lo importante: la prioridad absoluta
  - Procesos normales (tiempo compartido)
    - Prioridades dinámicas
    - Intenta ser equitativo
    - Procesos de baja prioridad pueden llegar a alcanzar la CPU
    - Favorecer interactivos

Algoritmo de planificación Linux: O(1)



Planificación de **tiempo compartido** en Linux: O(1)

- Planificación por prioridad expropiativa
  - Siempre se escoge el proceso de prioridad más alta
- Prioridad dinámica
  - Calculada a partir de la prioridad base (100..139) (nice -20..19)
  - Procesos heredan prioridad del padre
  - Usuarios pueden cambiar prioridad base (llamadas nice() y setpriority())
- Esquema de turno rotatorio (RR) para procesos de la misma prioridad
  - Cuanto fijo dependiente directamente de la prioridad estática del proceso

Planificación de **tiempo compartido** en Linux: O(1)

- Cálculo de la prioridad dinámica
  - Cada proceso tiene un prioridad base (nice) asignada en la creación del proceso
  - La dinámica se calcula a partir de la base para favorecer procesos interactivos
    - pdinámica = max (100, min(pestática – bonus +5, 139))
    - bonus = 0..10
      - bonificación (hasta -5) a procesos intensivos en E/S
      - penalización (hasta +5) a procesos intensivos en CPU
  - La bonificación se calcula en base al tiempo medio dormido (TMD en milisegundos)
    - Cuando un proceso "despierta" → TMD += tiempo dormido
    - Cuando un proceso deja la CPU → TMD -= tiempo ejecución CPU
    - 0 ≤ TMD ≤ 1000 ms
    - bonus = floor(TMD / 100)

Planificación de **tiempo compartido** en Linux: O(1)

- Cálculo de la prioridad dinámica
  - Intervalo "dinámico" (±5)
    - suficientemente grande para favorecer procesos interactivos
    - no suficiente para distorsionar totalmente prioridad base
      - Proceso con prioridad 12 nunca más urgente que prioridad 1
  - La prioridad se recalcula cuando el proceso ha agotado su cuanto
  - "Cuanto" fijo y dependiente de la prioridad estática del proceso
    - cuanto =
      - **(140 – prioridad estática) x 20 si prioridad estática < 120**
      - **(140 – prioridad estática) x 5 si prioridad estática ≥ 120**

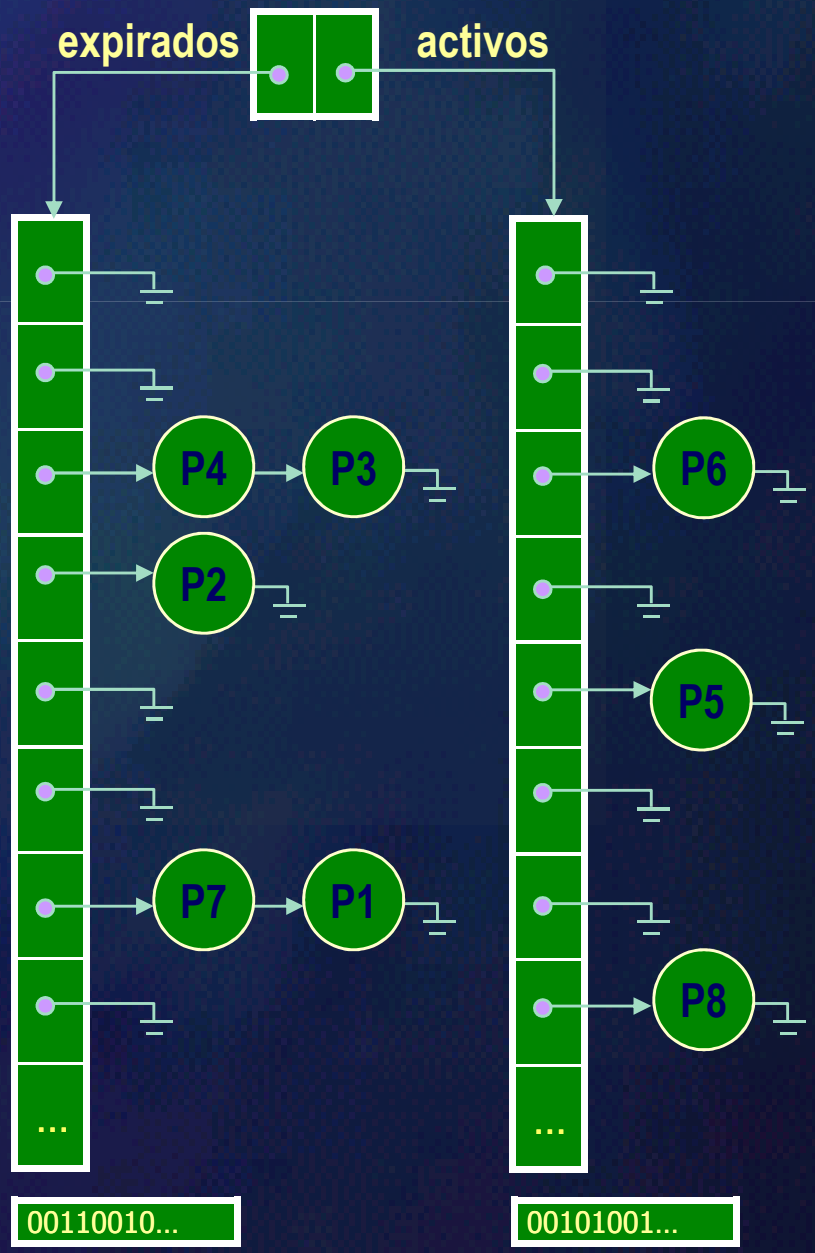
Descripción	Prioridad est.	Valor nice	Cuanto
Máxima	100	-20	800 ms
Alta	110	-10	600 ms
Defecto	120	0	100 ms
Baja	130	10	50 ms
Mínima	139	19	5 ms

Planificación de **tiempo compartido** en Linux: O(1)

- Turno rotatorio (RR) modificado
  - Dos colas de procesos "listos"
    - Cola "listos activos" formada por los procesos que aún no han consumido su cuanto
    - Cola "listos expirados" formada por los procesos que ya han agotado su cuanto
  - El proceso siguiente a ejecutar se elige sólo de los "activos"
  - Si un proceso se bloquea sin consumir su cuanto, cuando se desbloquee vuelve a "listos activos"
  - Si un proceso consume su rodaja, pasa a "listos expirados"

Planificación de **tiempo compartido** en Linux: O(1)

- Turno rotatorio (RR) modificado
  - Evita inanición:
    - Un proceso menos prioritario se ejecutará aunque existan procesos más prioritarios listos (pero que hayan consumido su cuanto)
  - Aparecen "épocas de ejecución"
    - Todos los procesos entran a ejecutarse en una época
    - Una época termina cuando todos los procesos han entrado a ejecución (no hay listos activos)
  - Cambio de época
    - Basta cambiar cola de expirados por cola de activos



Planificación de **tiempo compartido** en Linux: O(1)

- Turno rotatorio (RR) modificado
  - Implementación real un poco más complicada
  - Clasificación de los procesos en Batch e Interactivos
  - Interactivo si:
    - prioridad dinámica ≤ 3 x prioridad estática / 4 + 28
    - Depende del bonus (TMD) del proceso
  - Procesos Batch
    - Cuando agota su cuanto es movido a "expirados"
  - Procesos Interactivos
    - Cuando agota su cuanto normalmente permanece en "activos"
    - Es movido a expirados si:
      - El proceso expirado más viejo lleva mucho tiempo esperando
      - Hay un proceso expirado con mayor prioridad estática que el interactivo

Algoritmo de planificación Linux: CFS

- Desde kernel 2.6.23
  - Completely Fair Scheduler
  - $O(\log_2(n))$
- Dos clases de procesos – dos políticas
  - Procesos de tiempo real
    - **Siempre se despachan antes** que los procesos normales
    - Prioridades estáticas
    - No equitativo, lo importante: la prioridad absoluta
  - Procesos normales (tiempo compartido)
    - Prioridades **estáticas**
    - Intenta ser equitativo
    - Procesos de baja prioridad pueden llegar a alcanzar la CPU
    - Favorecer interactivos

Algoritmo de planificación Linux: CFS

- Idea
  - Repartir el uso del procesador proporcionalmente al valor NICE de cada proceso en ejecución
  - No se asigna un cuanto de tiempo a los procesos si no una porción del procesador
  - El valor NICE actúa como peso para la porción
- NICE
  - Valor entre [-20, 19] (menor valor más peso)
  - Pesos de los los valores NICE en CFS:

Nice	-20	-19	-18	-17	-16	-15	-14	-13	-12	-11
Weight	88761	71755	56483	46273	36291	29154	23254	18705	14949	11916
Nice	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
Weight	9548	7620	6100	4904	3906	3121	2501	1991	1586	1277
Nice	0	1	2	3	4	5	6	7	8	9
Weight	1024	820	655	526	423	335	272	215	172	137
Nice	10	11	12	13	14	15	16	17	18	19
Weight	110	87	70	56	45	36	29	23	18	15

Algoritmo de planificación Linux: CFS

- Latencia Objetivo (sched_latency)
  - Tiempo en el que todos los procesos en ejecución **tienen** que entrar a ejecutarse
  - Valores usados: 20 ms (desktop), 100 ms (server)
  - Reparto proporcional al valor NICE
- Rodaja de tiempo (time_slice)
  - Tiempo en el que se permite ejecutarse a un proceso sin ser reemplazado
  - Proporcional a su NICE

$$time\_slice(P_i) = \frac{weight_{P_i}}{\sum weight_{P_j}} \times sched\_latency$$

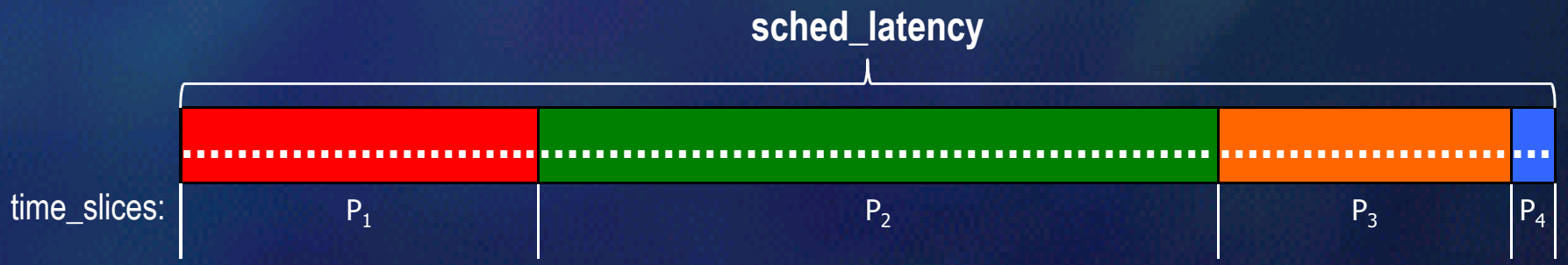
Algoritmo de planificación Linux: CFS

- Ejemplo: 4 procesos

$$time\_slice(P_i) = \frac{weight_{P_i}}{\sum weight_{P_j}} \times sched\_latency$$

- $\Sigma weight_P = 1024+1991+820+110 = 3945$

	P_1	P_2	P_3	P_4
Nice	0	-3	2	10
Weight	1024	1991	820	110
Weight/Σw	0,26	0,50	0,21	0,03



- time_slice(P) siempre ≥ min_granularity
  - Si no se cumple se aumenta sched_latency

Algoritmo de planificación Linux: CFS

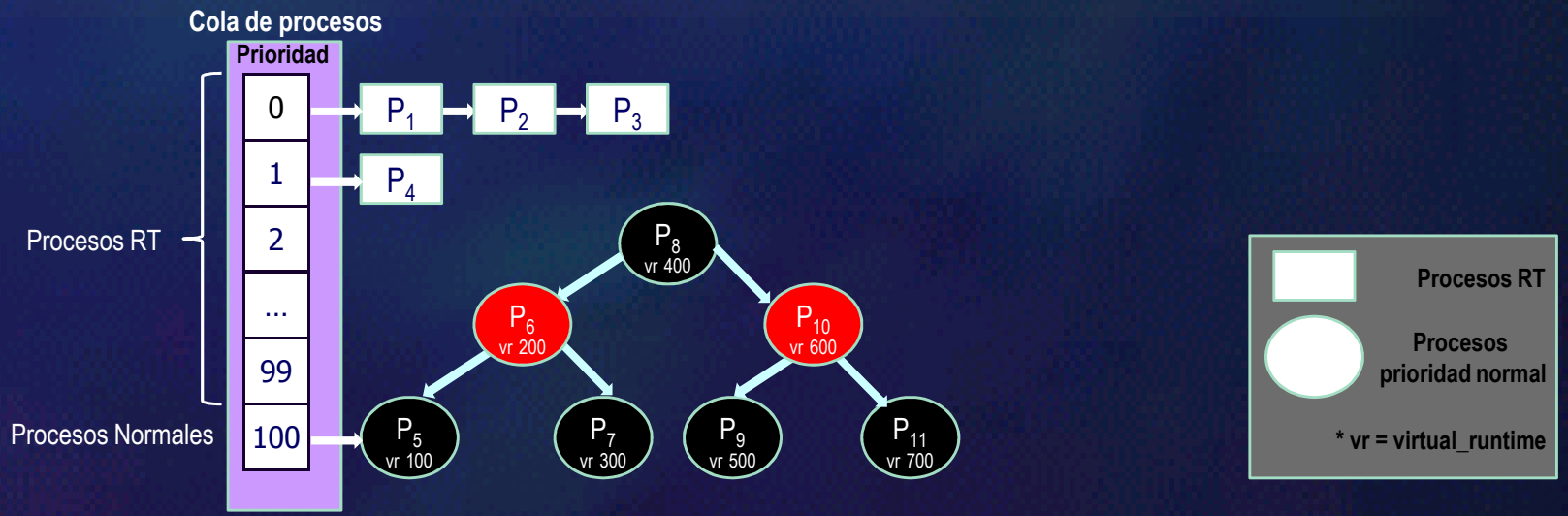
- Virtual runtime
  - No se asigna un "cuanto" sino porcentaje de uso de CPU
  - Hay que contabilizar el tiempo que cada proceso consume de CPU
  - **virtual_runtime:** Tiempo de ejecución acumulado de un proceso ponderado inversamente por su peso

$$virtual\_runtime(P_i,t) += \frac{weight_0}{weight_{P_i}} \times tiempo\_ejec(P_i,t)$$

    - $weight_0$ = peso de nice 0
    - A procesos con nice 0 se les contabiliza exactamente su tiempo de ejecución
    - A procesos con nice < 0 se les contabiliza **menos** de su tiempo de ejecución real
    - A procesos con nice > 0 se les contabiliza **más** de su tiempo de ejecución real

Algoritmo de planificación Linux: CFS

- Cola de procesos
  - Una sola cola READY para todos los procesos (prioridad normal)
  - Ordenada en valor creciente de virtual_runtime
    - Procesos que se han ejecutado "poco tiempo" serán los primeros
  - Implementación mediante un red-black-tree
    - Inserción y borrado en $O(\log_2(n))$
    - Proceso a ejecutar: el más a la izquierda del árbol (O(1))



Algoritmo de planificación Linux: CFS

- Algoritmo de planificación CFS
  - En cada "tick" de planificación
    - Restar al time_slice del proceso actual el periodo de un tick
      - Si time_slice llega a 0, NEED_RESCHED = true
    - Actualizar virtual_runtime del proceso actual

$$virtual\_runtime(P_i,t) += \frac{weight_0}{weight_{P_i}} \times tiempo\_ejec(P_i,t)$$

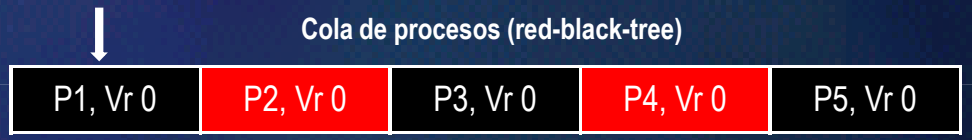
    - Si NEED_RESCHED = true
      - Planificar tarea con el menor virtual_runtime del árbol (la más a la izquierda)

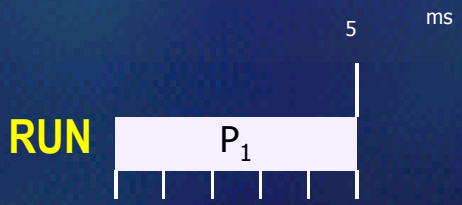
NEED_RESCHED puede pasar a valer true cuando se inserta un proceso en el árbol (nuevo, fin de bloqueo) y si se coloca como el primero (más a la izquierda, con menor vr)

Algoritmo de planificación Linux: CFS

- Ejemplo: 5 procesos, sched_latency = 6, tick = 1ms

	nice	$weight_i$	$weight_i/weight_s$	time_slice
P_1	-10	9548	0,6753	4,0518
P_2	-5	3121	0,2208	1,3248
P_3	0	1024	0,0724	0,4344
P_4	5	335	0,0237	0,1422
P_5	10	110	0,0078	0,0468
Total		14138	1,0000	6



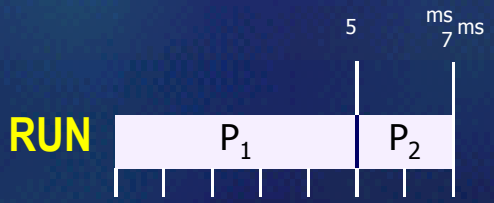


Algoritmo de planificación Linux: CFS

- Ejemplo: 5 procesos, sched_latency = 6, tick = 1ms

	nice	$weight_i$	$weight_i/weight_s$	time_slice
P_1	-10	9548	0,6753	4,0518
P_2	-5	3121	0,2208	1,3248
P_3	0	1024	0,0724	0,4344
P_4	5	335	0,0237	0,1422
P_5	10	110	0,0078	0,0468
Total		14138	1,0000	6

$Vr(P_1)$ = 5 ms * 1024/9548 = 0.536 ms

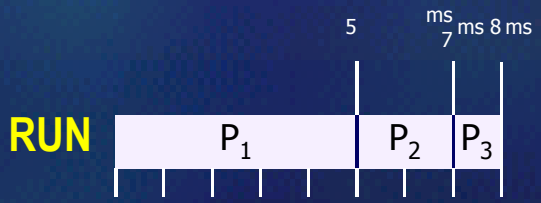


Algoritmo de planificación Linux: CFS

- Ejemplo: 5 procesos, sched_latency = 6, tick = 1ms

	nice	$weight_i$	$weight_i/weight_s$	time_slice
P_1	-10	9548	0,6753	4,0518
P_2	-5	3121	0,2208	1,3248
P_3	0	1024	0,0724	0,4344
P_4	5	335	0,0237	0,1422
P_5	10	110	0,0078	0,0468
Total		14138	1,0000	6

$Vr(P_2) = 2\ ms * 1024/3121 = 0.656\ ms$

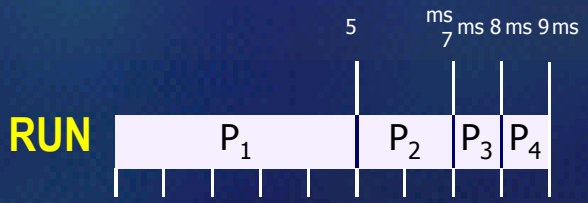


Algoritmo de planificación Linux: CFS

- Ejemplo: 5 procesos, sched_latency = 6, tick = 1ms

	nice	$weight_i$	$weight_i/weight_s$	time_slice
P_1	-10	9548	0,6753	4,0518
P_2	-5	3121	0,2208	1,3248
P_3	0	1024	0,0724	0,4344
P_4	5	335	0,0237	0,1422
P_5	10	110	0,0078	0,0468
Total		14138	1,0000	6

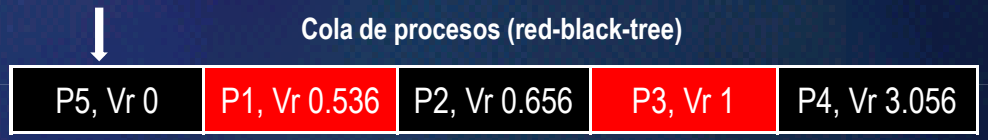
$Vr(P_3) = 1\ ms * 1024/1024 = 1\ ms$



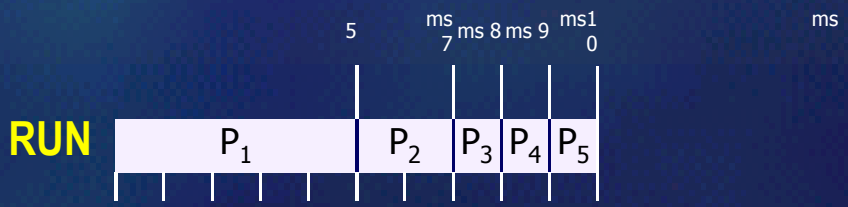
Algoritmo de planificación Linux: CFS

- Ejemplo: 5 procesos, sched_latency = 6, tick = 1ms

	nice	$weight_i$	$weight_i/weight_s$	time_slice
P_1	-10	9548	0,6753	4,0518
P_2	-5	3121	0,2208	1,3248
P_3	0	1024	0,0724	0,4344
P_4	5	335	0,0237	0,1422
P_5	10	110	0,0078	0,0468
Total		14138	1,0000	6



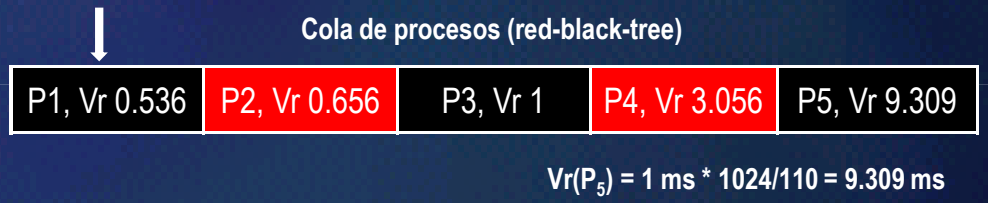
$Vr(P_4)$ = 1 ms * 1024/335= 3.056 ms



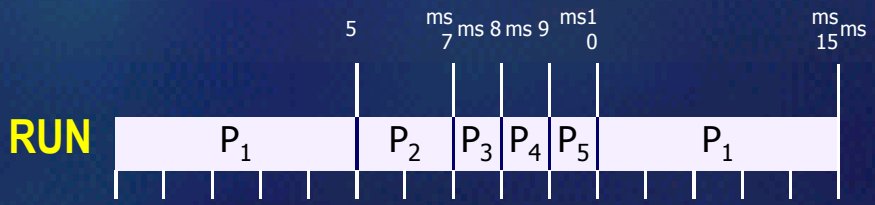
Algoritmo de planificación Linux: CFS

- Ejemplo: 5 procesos, sched_latency = 6, tick = 1ms

	nice	$weight_i$	$weight_i/weight_s$	time_slice
P_1	-10	9548	0,6753	4,0518
P_2	-5	3121	0,2208	1,3248
P_3	0	1024	0,0724	0,4344
P_4	5	335	0,0237	0,1422
P_5	10	110	0,0078	0,0468
Total		14138	1,0000	6



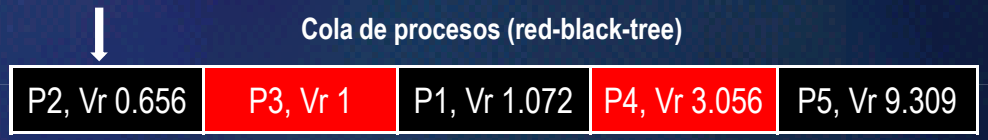
$Vr(P_5)$ = 1 ms * 1024/110 = 9.309 ms



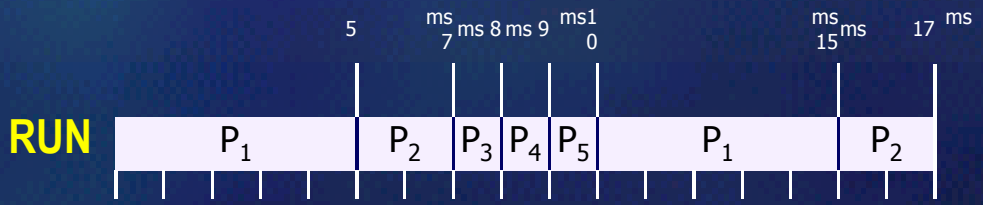
Algoritmo de planificación Linux: CFS

- Ejemplo: 5 procesos, sched_latency = 6, tick = 1ms

	nice	$weight_i$	$weight_i/weight_s$	time_slice
P_1	-10	9548	0,6753	4,0518
P_2	-5	3121	0,2208	1,3248
P_3	0	1024	0,0724	0,4344
P_4	5	335	0,0237	0,1422
P_5	10	110	0,0078	0,0468
Total		14138	1,0000	6



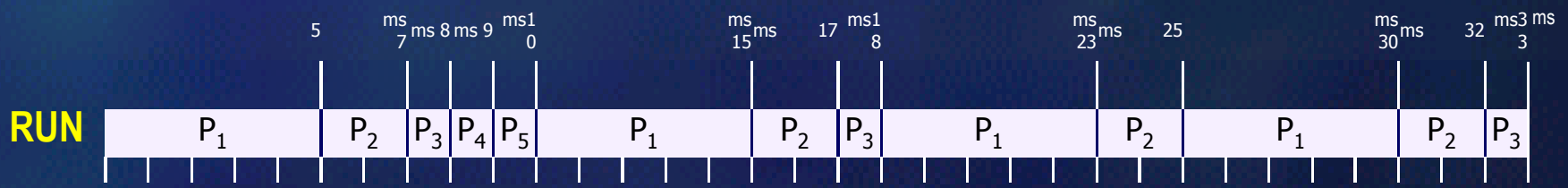
Vr(P1) = 0.536 + 5 ms * 1024/9548 = 1.072 ms



Algoritmo de planificación Linux: CFS

- Ejemplo: 5 procesos, sched_latency = 6, tick = 1ms

	nice	$weight_i$	$weight_i/weight_5$	time_slice	Total time	%CPU
P_1	-10	9548	0,6753	4,0518	20 ms	60,6
P_2	-5	3121	0,2208	1,3248	8 ms	24,2
P_3	0	1024	0,0724	0,4344	3 ms	9
P_4	5	335	0,0237	0,1422	1 ms	3
P_5	10	110	0,0078	0,0468	1 ms	3
Total		14138	1,0000	6	33 ms	1



Ejemplo de planificación

Proceso	Llegada	Tiempo CPU	Prioridad
T0	0	50	4
T1	10	50	3
T2	20	20	1
T3	30	10	2

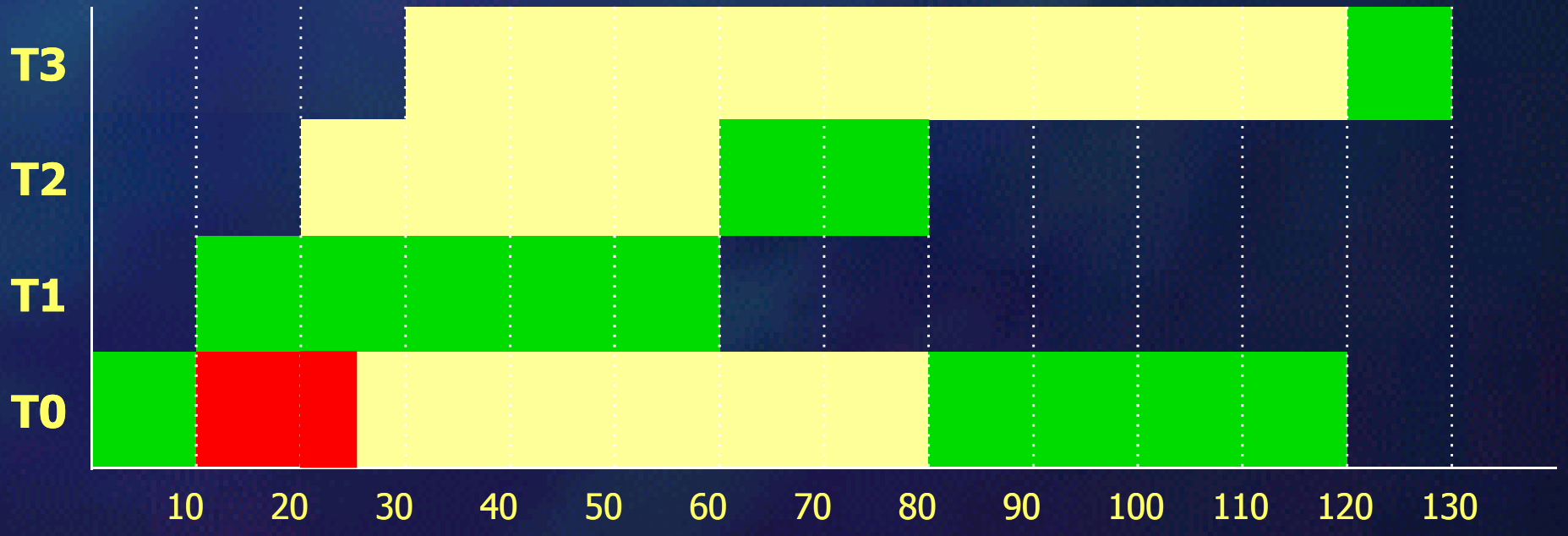
- El proceso T0 se bloquea a las 10 unidades de tiempo durante 15 unidades de tiempo

	No expropiativa	Expropiativa
por orden de llegada	FCFS	Turno circular (RR) **Q=10**
por tiempo de ejecución	Trabajo más corto primero (SJF)	Tiempo restante más corto primero (SRTN)
por prioridades	Prioridades	Prioridades expropiativas

Ejemplo FCFS

Tiempo	Ejecucion	Listos	Bloqueado	
0	T0			LL T0
10	T1		T0	LL T1 BL T0
20	T1	T2	T0	LL T2
25	T1	T2,T0		DBL T0
30	T1	T2,T0,T3		LL T3
60	T2	T0,T3		FIN T1
80	T0	T3		FIN T2
120	T3			FIN T0
130				FIN T3

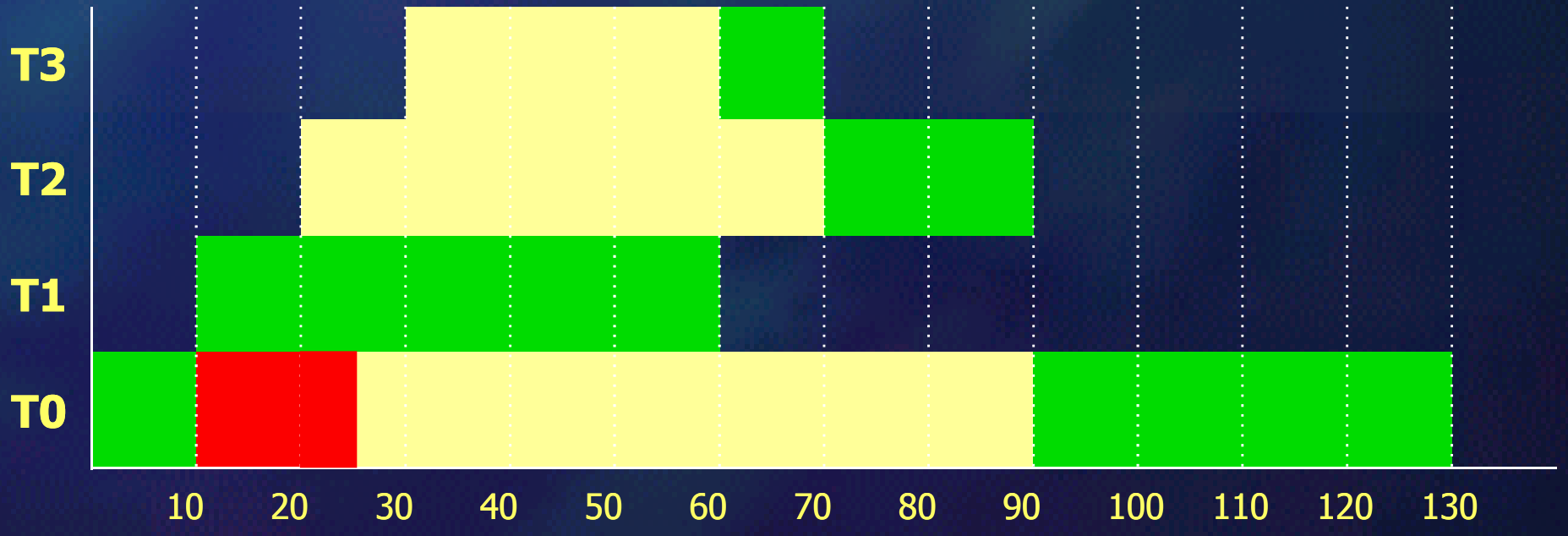
Tr = (120+50+60+100)/4 = 82.5 **Te = (55+0+40+90)/4 = 46.25**



Ejemplo SJF

Tiempo	Ejecucion	Listos	Bloqueado	
0	T0			LL T0
10	T1		T0	LL T1 BL T0
20	T1	T2	T0	LL T2
25	T1	T2,T0		DBL T0
30	T1	T3,T2,T0		LL T3
60	T3	T2,T0		FIN T1
70	T2	T0		FIN T3
90	T0			FIN T2
130				FIN T0

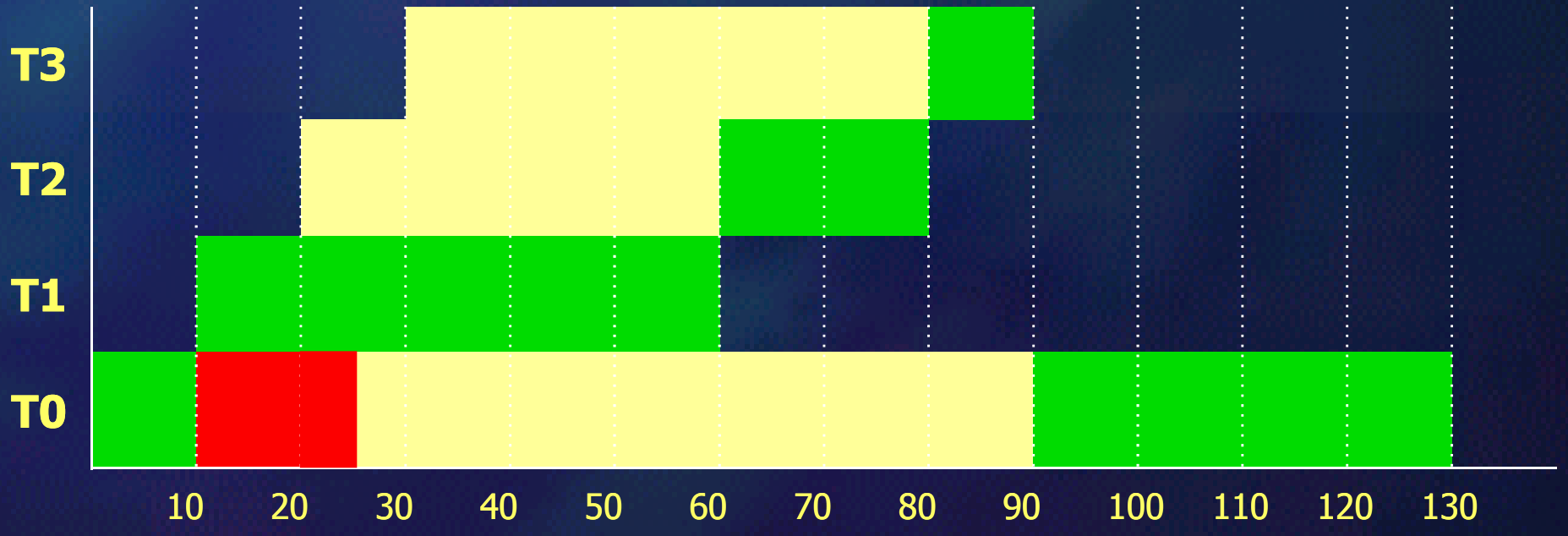
Tr = (130+50+70+40)/4 = 72.5 **Te = (65+0+50+30)/4 = 36.25**



Ejemplo Prioridad

Tiempo	Ejecucion	Listos	Bloqueado	
0	T0			LL T0
10	T1		T0	LL T1 BL T0
20	T1	T2	T0	LL T2
25	T1	T2,T0		DBL T0
30	T1	T2,T3,T0		LL T3
60	T2	T3,T0		FIN T1
80	T3	T0		FIN T2
90	T0			FIN T3
130				FIN T0

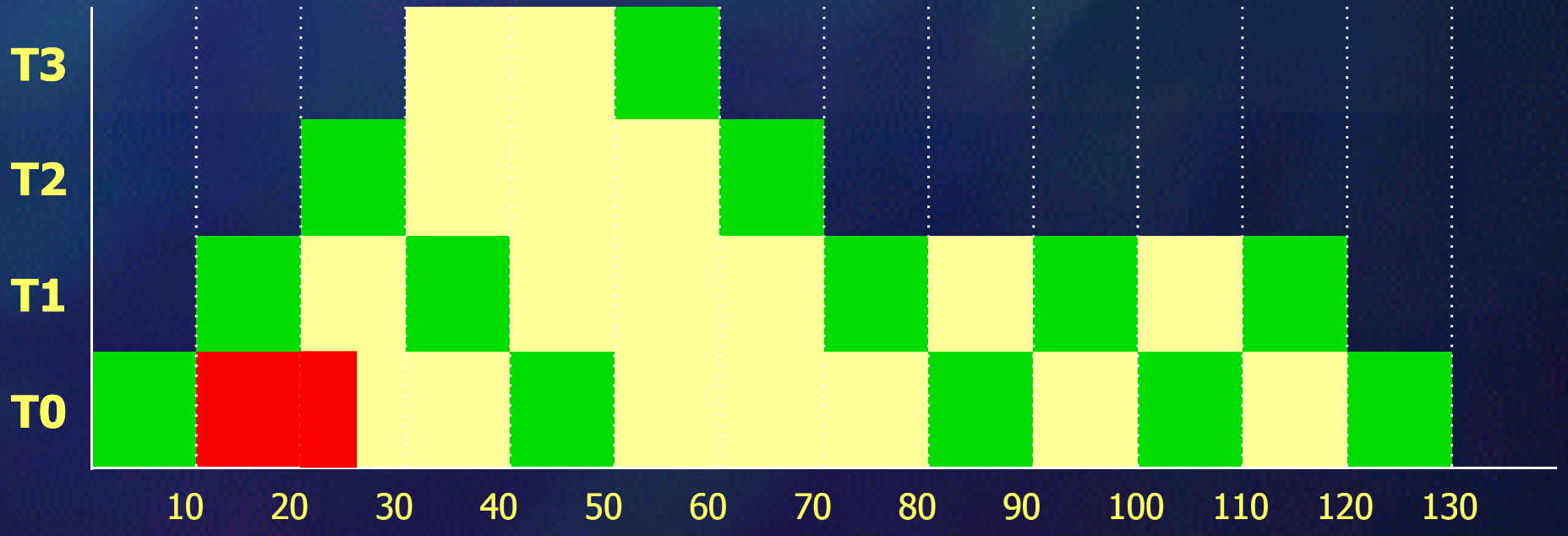
Tr = (130+50+60+60)/4 = 75 Te = (65+0+40+50)/4 = 38.75



Ejemplo RR

Tiempo	Ejecucion	Listos	Bloqueado	
0	T0			LL T0
10	T1		T0	LL T1 BL T0
20	T2	T1	T0	LL T2
25	T2	T1,T0		DBL T0
30	T1	T0,T3,T2		LL T3
40	T0	T3,T2,T1		
50	T3	T2,T1,T0		
60	T2	T1,T0		FIN T3
70	T1	T0		FIN T2
80	T0	T1		
90	T1	T0		
100	T0	T1		
110	T1	T0		
120	T0			FIN T1
130				FIN T0

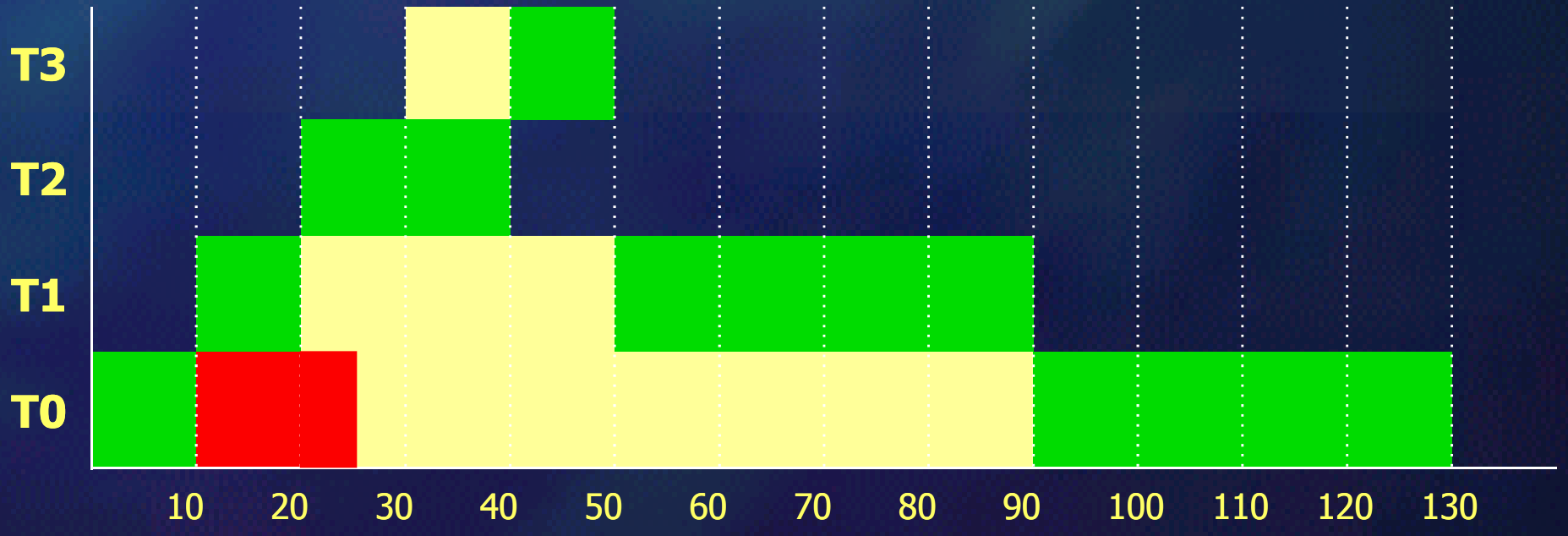
Tr = (130+110+50+30)/4 = 80 **Te = (65+60+30+20)/4 = 43.75**



Ejemplo SRTN

Tiempo	Ejecucion	Listos	Bloqueado	
0	T0			LL T0
10	T1		T0	LL T1 BL T0
20	T2	T1	T0	LL T2
25	T2	T1,T0		DBL T0
30	T2	T3,T1,T0		LL T3
40	T3	T1,T0		FIN T2
50	T1	T0		FIN T3
90	T0			FIN T1
130				FIN T0

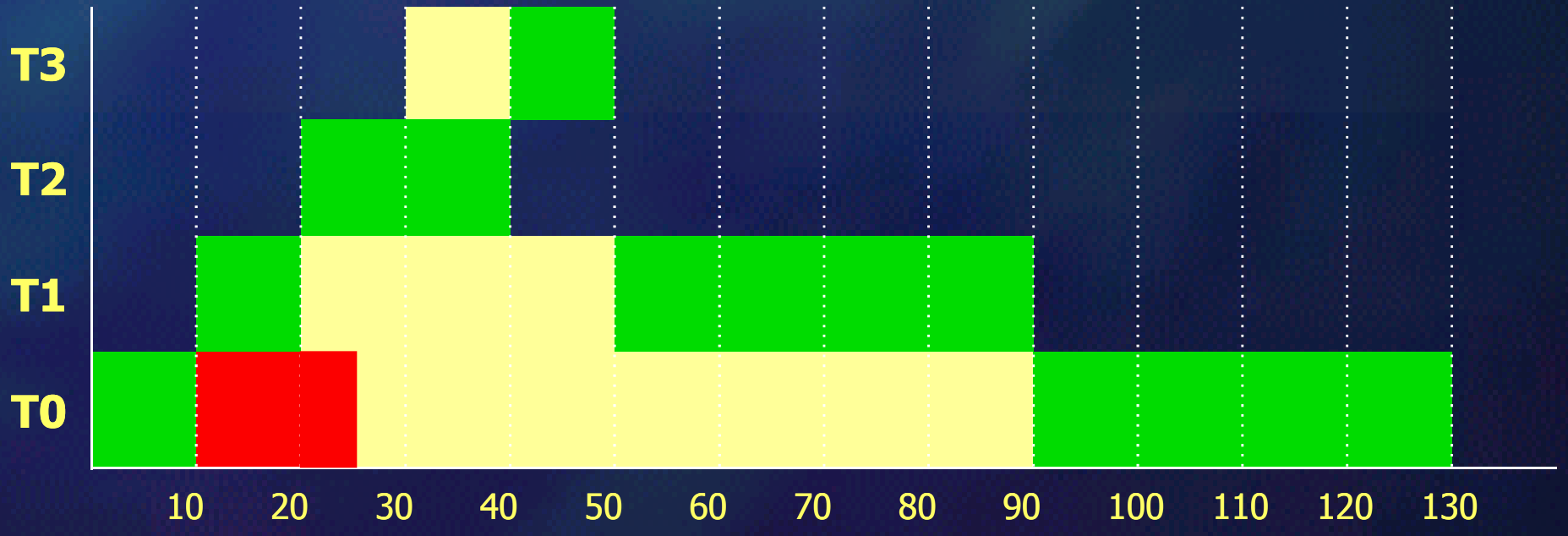
Tr = (130+80+20+20)/4 = 62.5 **Te = (65+30+0+10)/4 = 26.25**



Ejemplo Prioridad expropiativo

Tiempo	Ejecucion	Listos	Bloqueado	
0	T0			LL T0
10	T1		T0	LL T1 BL T0
20	T2	T1	T0	LL T2
25	T2	T1,T0		DBL T0
30	T2	T3,T1,T0		LL T3
40	T3	T1,T0		FIN T2
50	T1	T0		FIN T3
90	T0			FIN T1
130				FIN T0

Tr = (130+80+20+20)/4 = 62.5 **Te = (65+30+0+10)/4 = 26.25**



SJF

Estimación ráfaga n+1 | Tiempo ráfaga n | Estimación ráfaga n

$$\tau_{n+1} = \alpha \cdot t_n + (1-\alpha) \cdot \tau_n \qquad 0 \le \alpha \le 1$$

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

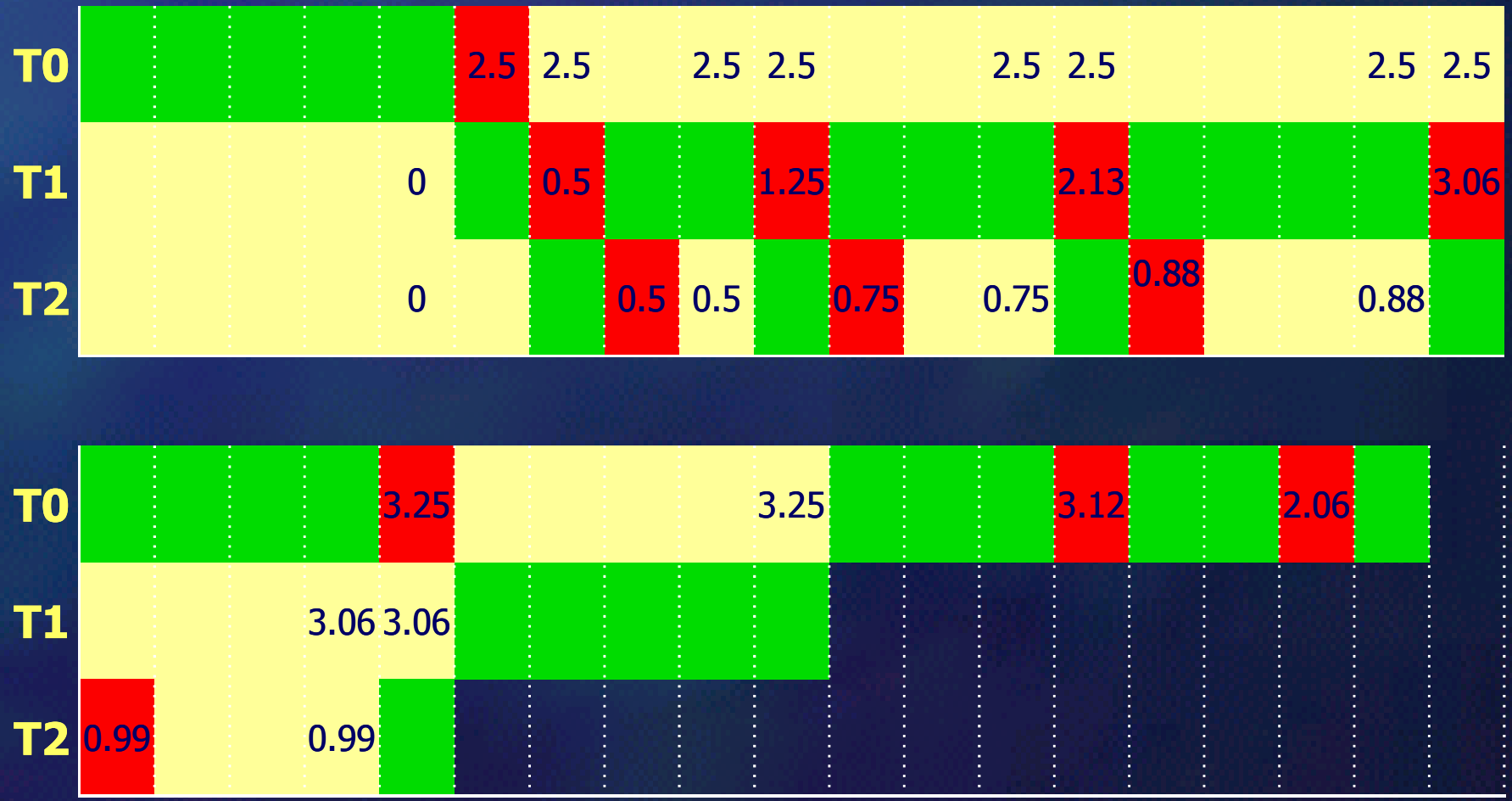
Proceso	CPU	CPU	CPU	CPU	CPU
T0	5	4	3	2	1
T1	1	2	3	4	5
T2	1	1	1	1	1

- Ráfagas de E/S siempre de 1
- α = 0.5

SJF

$$\tau_{n+1} = \alpha \cdot t_n + (1-\alpha) \cdot \tau_n \qquad \alpha = 0.5$$

Proceso	CPU	CPU	CPU	CPU	CPU
T0	5	4	3	2	1
T1	1	2	3	4	5
T2	1	1	1	1	1



FSO - Examen septiembre de 2006

8.- En un sistema con planificación de colas de realimentación multinivel de tres niveles (expropiativo) se ejecutan tres procesos.

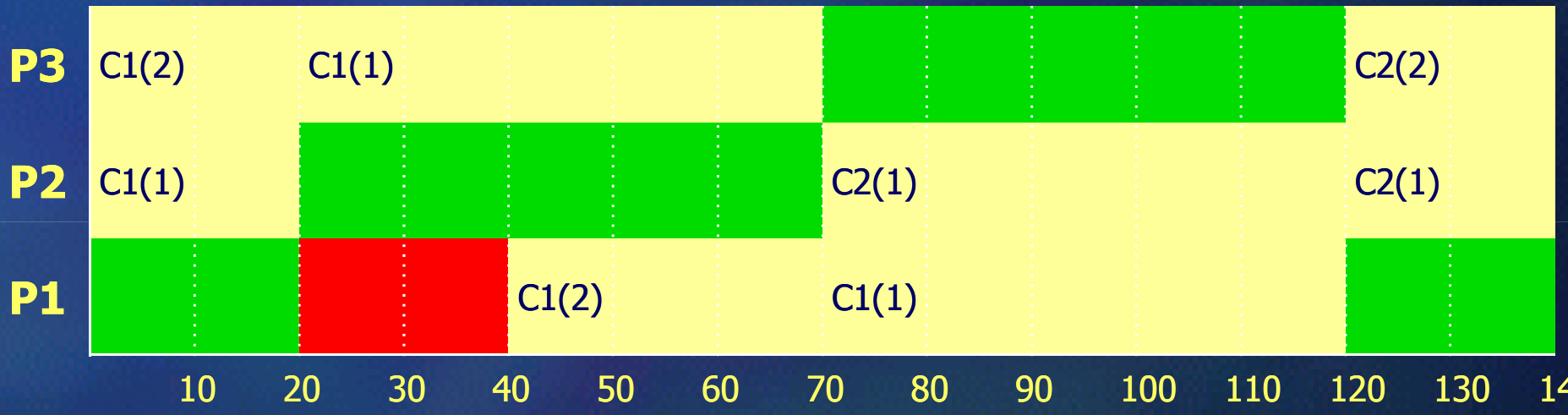
Los tiempos de computación pura de los procesos (total CPU) son los siguientes: P1=60 ciclos, P2=130 ciclos y P3=180 ciclos. Además el proceso P1 realiza una operación de E/S cuando han pasado 20 ciclos de su tiempo de computación. El proceso P2 realiza una operación de E/S cuando han pasado 80 ciclos de su tiempo de computación. El proceso P3 no realiza operaciones de E/S. Cada operación de E/S dura 20 ciclos.

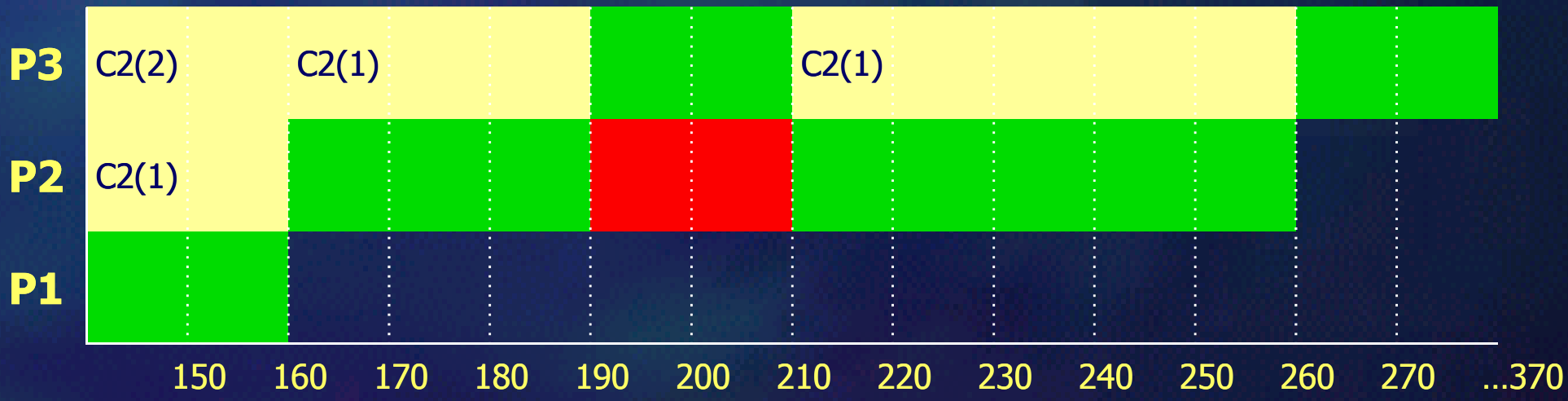
Los cuantos de tiempo asociados a cada cola, a medida que desciende la prioridad de las colas son Cola1: 50, Cola2: 100 y Cola3: 120 ciclos (la Cola1 es la de mayor prioridad).

Suponiendo que, inicialmente, todos los procesos llegan la mismo tiempo al sistema de planificación y que el orden inicial en la cola es P1(primero), P2(segundo) y P3 (tercero), completar el diario de ejecución en la tabla siguiente anotando el tiempo en el cual ocurre cada nuevo evento, dónde se encuentra cada proceso en ese momento y la descripción del evento.

Tiempo	Ejecución	Cola1	Cola2	Cola3	Bloqueados	Descripción evento
0	P1	P2,P3				Llegada de los procesos P1,P2, P3 Despacho de P1

FSO - Examen septiembre de 2006

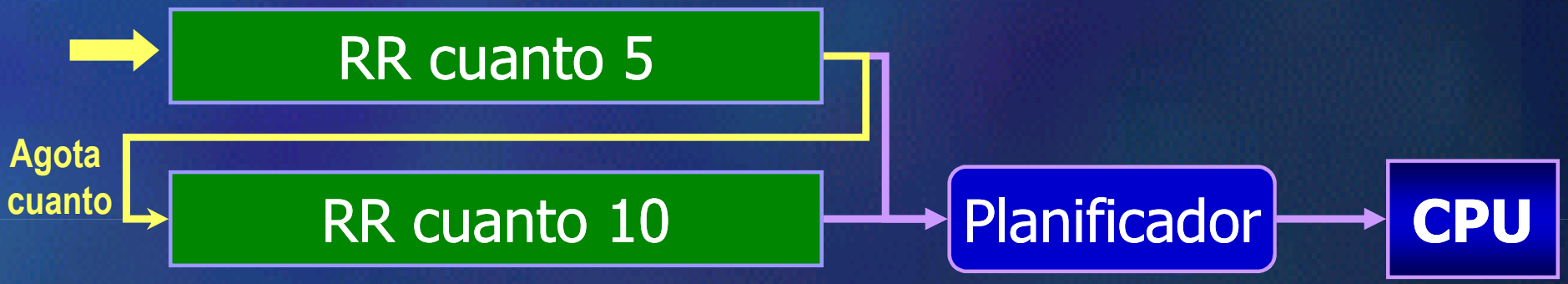




FSO - Examen septiembre de 2006

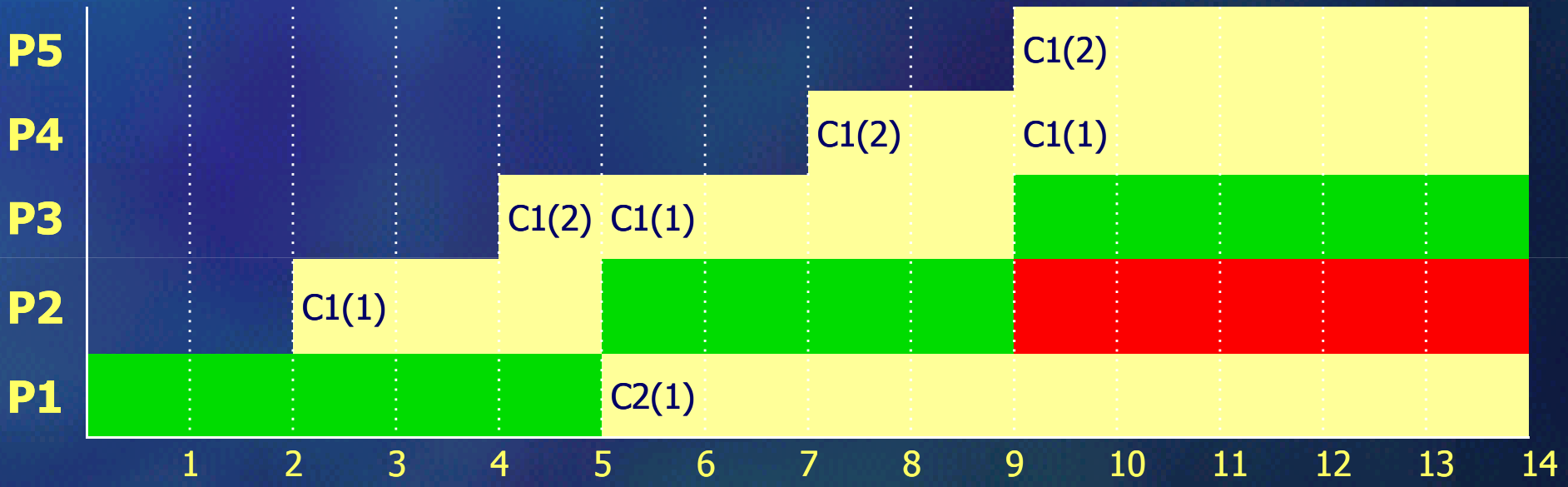
T	Ejec.	C1	C2	C3	Blq	Descripción evento
0	P1(20)[1]	P2(80)[1], P3(180)[1]				Llegada de los procesos P1,P2, P3. Despacho P1
20	P2(70)[1]	P3(180)[1]			P1(40)[1]	Bloqueo de P1, despacho P2
40	P2(70)[1]	P3(180)[1], P1(40)[2]				Desbloqueo de P1
70	P3(120)[1]	P1(40)[2]	P2(30)[1]			P2 a C2, despacho P3
120	P1(160)[2]		P2(30)[1], P3(130)[1]			P3 a C2, despacho P1
160	P2(190)[1]		P3(130)[1]			Fin P1, despacho P2
190	P3(290)[1]				P2(210)[1]	Bloqueo P2, despacho P3
210	P2(260)[2]		P3(110)[1]			Desbloqueo P2, expropiación P3 a C2
260	P3(360)[1]					Fin P2, despacho P3
360	P3(370)[1]					despacho P3
370						Fin P3

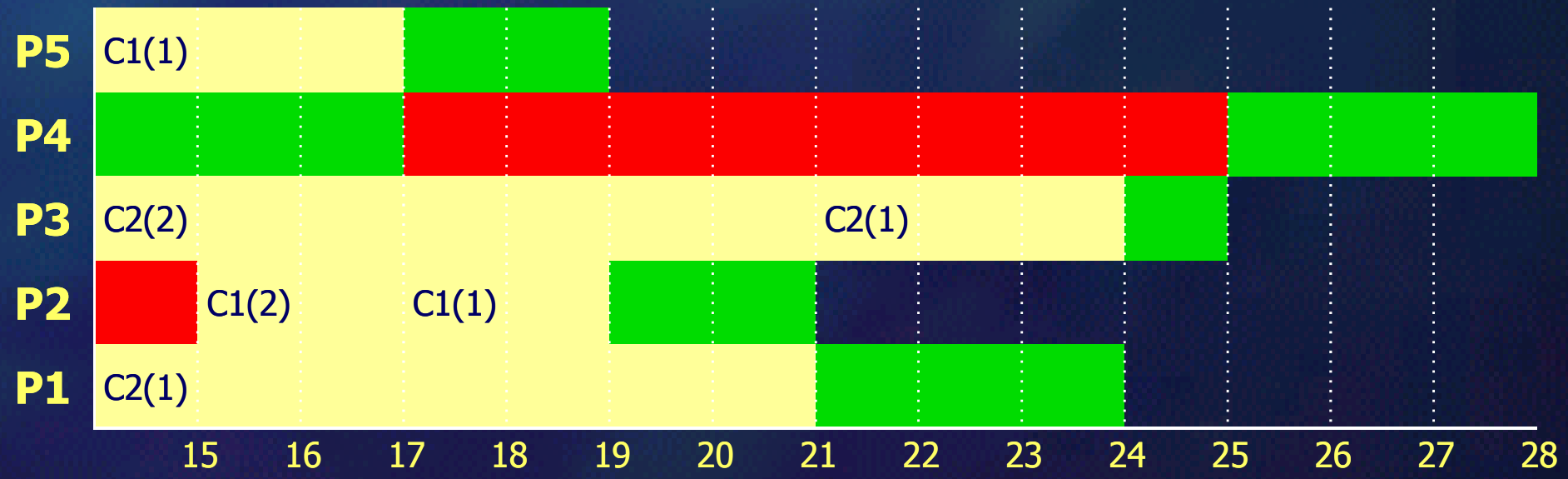
Ejemplo colas multinivel con realimentación



	Llegada	CPU	E/S	CPU	Tin	Tout	Te	Tr
P1	0	8	X	X	0	24	16	24
P2	2	4	6	2	2	21	7	19
P3	4	6	X	X	4	25	15	21
P4	7	3	8	3	7	28	7	21
P5	9	2	X	X	9	19	8	10

Ejemplo colas multinivel con realimentación





Ejemplo colas multinivel con realimentación

Tiempo	Ejecucion	C1	C2	Bloqueado	Descripción
0		P1(8)[1]			LL P1
0	P1(5) [1]				DSP P1
2	P1(5) [1]	P2(4) [1]			LL P2
4	P1(5) [1]	P2(4) [1],P3(6) [1]			LL P3
5	P2(9) [1]	P3(6) [1]	P1(3) [1]		P1 A C2, DSP P2
7	P2(9) [1]	P3(6) [1],P4(3) [1]	P1(3) [1]		LL P4
9	P3(14) [1]	P4(3) [1],P5(2) [1]	P1(3) [1]	P2(15) [1]	BLQ P2, LL P5, DSP P3
14	P4(17) [1]	P5(2) [1]	P1(3) [1], P3(1) [1]	P2(15) [1]	P3 A C2, DSP P4
15	P4(17) [1]	P5(2) [1], P2(2) [2]	P1(3) [1], P3(1) [1]		DSBLQ P2
17	P5(19) [1]	P2(2) [2]	P1(3) [1], P3(1) [1]	P4(25) [1]	BLQ P4, DSP P5
19	P2(21) [2]		P1(3) [1], P3(1) [1]	P4(25) [1]	FIN P5, DSP P2
21	P1(24) [1]		P3(1) [1]	P4(25) [1]	FIN P2, DSP P1
24	P3(25) [1]			P4(25) [1]	FIN P1, DSP P3
25	P4(28) [2]				FIN P3, DSP P4
28					FIN P4

Notación

- Descripción:
  - LL P# : Llegada del proceso P#
  - DSP P# : Despacho del proceso P#
  - BLQ P# : Bloqueo del proceso P#
  - DSBLQ P# : Desbloqueo del proceso P#
  - P# A C# : Proceso P# pasa a cola C#
- Ejecución:
  - $P\#(N)^R$: El proceso P# estará en ejecución (en principio) hasta el instante N (correspondiente a la ráfaga de CPU R)
- Bloqueado:
  - $P\#(N)^R$: El proceso P# estará bloqueado hasta el instante N (correspondiente a la ráfaga de E/S R)
- En Cola C#:
  - $P\#(N)^R$: El proceso P# se encuentra en la cola C# y le quedan N unidades de tiempo correspondientes a su ráfaga de CPU R