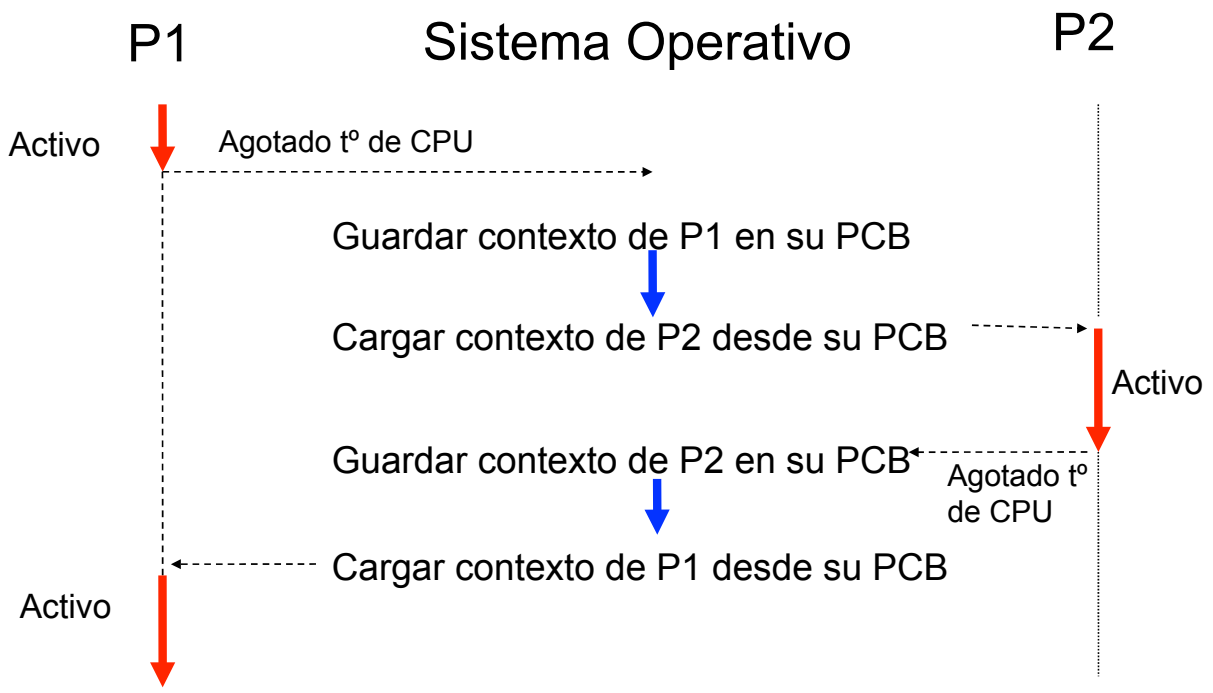
1. GENERALIDADES SOBRE PROCESOS E HILOS

Nos situamos en la materia abordada en Fundamentos del Software sobre procesos e hilos, la ampliamos y profundizaremos en algunos de sus contenidos.

Cambio de contexto

- Cuando un proceso esta ejecutándose, su PC, puntero a pila, registros, etc., están cargados en la CPU (es decir, los registros hardware contienen los valores actuales).
- Cuando el SO detiene un proceso ejecutándose, salva los valores actuales de estos registros (*contexto*) en el PCB de ese proceso.
- La acción de conmutar la CPU de un proceso a otro se denomina ***cambio de contexto***. Los sistemas de tiempo compartido realizan de 10 a 100 cambios de contexto por segundo. Este trabajo es sobrecarga.

Cambio de contexto (y II)



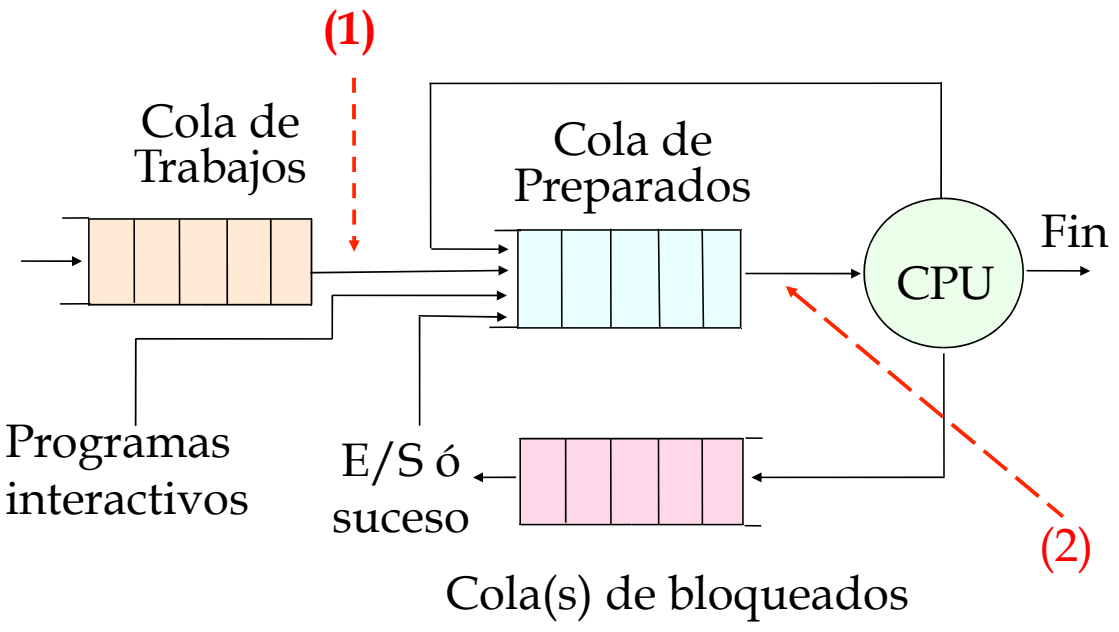
PCB's y Colas de Estados

- El SO mantiene una colección de colas que representan el estado de todos los procesos en el sistema.
- Típicamente hay una cola por estado.
- Cada PCB esta encolado en una cola de estado acorde a su estado actual.
- Conforme un proceso cambia de estado, su PCB es retirado de una cola y encolado en otra.

Colas de estados

- *Cola de trabajos* -- conjunto de los trabajos pendientes de ser admitidos en el sistema (trabajos por lotes que no están en memoria).
- *Cola de preparados* -- conjunto de todos los procesos que residen en memoria principal, preparados y esperando para ejecutarse.
- *Cola(s) de bloqueados* -- conjunto de todos los procesos esperando por un dispositivo de E/S particular o por un suceso.

Migración entre colas



Planificación de procesos
Tipos de planificadores

- Planificador ➔ Parte del SO que controla la utilización de un recurso.
- Tipos de planificadores de la CPU:
  - » *Planificador a largo plazo* (planificador de trabajos): selecciona los procesos que deben llevarse a la cola de preparados; (1) en figura anterior.
  - » *Planificador a corto plazo* (planificador de la CPU): selecciona al proceso que debe ejecutarse a continuación, y le asigna la CPU; (2) en figura anterior.
  - » *Planificador a medio plazo*.

Características de los planificadores

- *El planificador a corto plazo* :
  - » trabaja con la cola de preparados.
  - » se invoca muy frecuentemente (milisegundos) por lo que debe ser rápido.
- *El planificador a largo plazo*
  - » Permite controlar el *grado de multiprogramación*.
  - » se invoca poco frecuentemente (segundos o minutos), por lo que puede ser más lento.

Clasificación de procesos

- *Procesos limitados por E/S o procesos cortos* -- dedican más tiempo a realizar E/S que computo; muchas ráfagas de CPU cortas y largos períodos de espera.

- *Procesos limitados por CPU o procesos largos* -- dedican más tiempo en computación que en realizar E/S; pocas ráfagas de CPU pero largas.

Mezcla de trabajos

Es importante que el planificador a largo plazo seleccione una buena mezcla de trabajos, ya que:

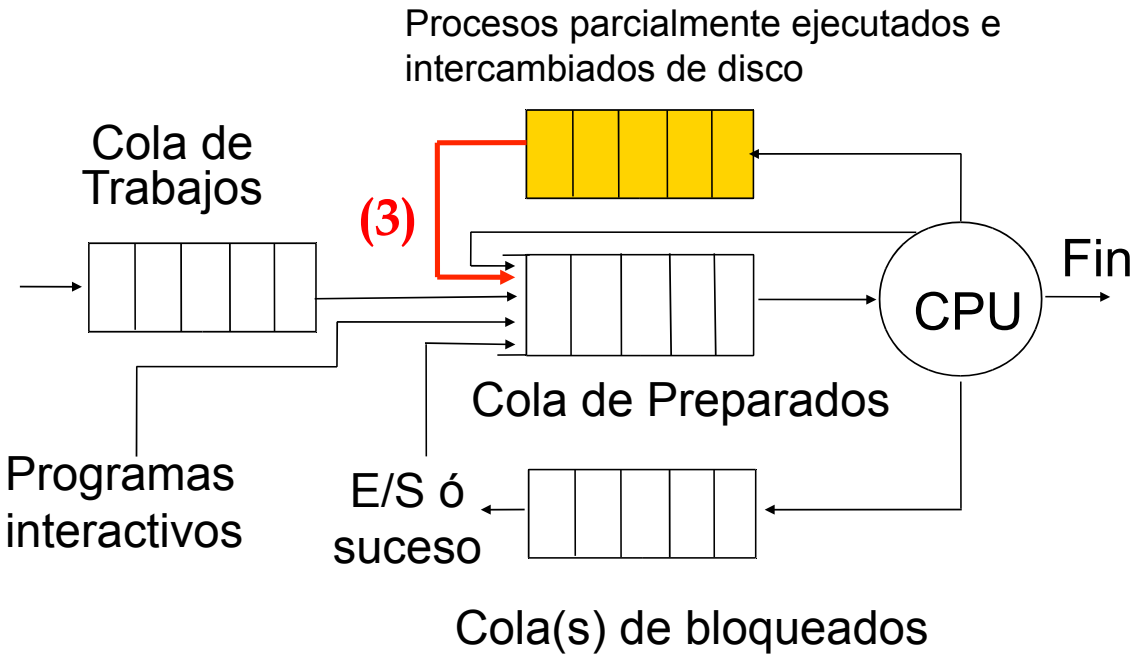
- Si todos los trabajos están limitados por E/S, la cola de preparados estará casi siempre vacía y el planificador a corto plazo tendrá poco que hacer.

- Si todos los procesos están limitados por CPU, la cola de E/S estará casi siempre vacía y el sistema estará de nuevo desequilibrado.

Planificador a medio plazo

- En algunos SO's, p. ej. SO de tiempo compartido, a veces es necesario sacar procesos de la memoria (reducir *el grado de multiprogramación*), bien para mejorar la mezcla de procesos, bien por cambio en los requisitos de memoria, y luego volverlos a introducir (intercambio o *swapping*).
- El *planificador a medio plazo* se encarga de devolver los procesos a memoria. Transición (3) en la siguiente figura.

Planificador a medio plazo (y II)



NOTA: Algunos autores utilizan el término “despachador” como equivalente a “planificador a corto plazo”, otros lo definen como sigue.....

- El despachador (*dispatcher*) es el módulo del SO que da el control de la CPU al proceso seleccionado por el planificador a corto plazo; esto involucra:
  - » Cambio de contexto (se realiza en modo kernel).
  - » Conmutación a modo usuario.
  - » Salto a la posición de memoria adecuada del programa para su reanudación.

En principio utilizaremos “despachador” como equivalente a “planificador a corto plazo”

Activación del Despachador

- El despachador puede actuar cuando:
  1. Un proceso no quiere seguir ejecutándose (finaliza o ejecuta una operación que lo bloquea)
  2. Un elemento del SO determina que el proceso no puede seguir ejecutándose (ej. E/S o retiro de memoria principal)
  3. El proceso agota el quantum de tiempo asignado
  4. Un suceso cambia el estado de un proceso de bloqueado a ejecutable

Políticas de planificación de la CPU

- Objetivos:
  - » Buen rendimiento (productividad)
  - » Buen servicio
- Para reflexionar sobre el comportamiento de las políticas de planificación se definen ciertas medidas asociadas a cada proceso:

(t es el tiempo de CPU del proceso)

⇨ *Tiempo de respuesta (T):* tiempo total transcurrido desde que se crea el proceso hasta que termina.

⇨ *Tiempo de espera (E)*: tiempo que el proceso ha estado esperando en la cola de preparados: T - t

⇨ *Penalización (P)* : T / t

Políticas de planificación de la CPU (y II)

- Otras medidas interesantes son:

⇨ tiempo perdido por el SO tomando decisiones que afectan a la planificación de procesos y haciendo los cambios de contexto necesarios.

En un sistema eficiente debe representar entre el 10% y el 30% del total del tiempo del procesador

⇨ tiempo en el que la cola de ejecutables está vacía y no se realiza ningún trabajo productivo

Políticas de planificación de CPU (y III)

- Las políticas de planificación se comportan de distinta manera dependiendo de la clase de procesos
  - » Ninguna política de planificación es completamente satisfactoria, cualquier mejora en una clase de procesos es a expensas de perder eficiencia en los procesos de otra clase.
- Podemos clasificarlas en:
  - » **No apropiativas (sin desplazamiento)**: una vez que se le asigna el procesador a un proceso, no se le puede retirar hasta que éste voluntariamente lo deje (finalice o se bloquee)
  - » **Apropiativas (con desplazamiento)**: al contrario, el SO puede apropiarse del procesador cuando lo decida

Políticas de planificación de la CPU (y IV)

- FCFS
- El más corto primero:
  - » no apropiativo
  - » apropiativo
- Planificación por prioridades
- Por turnos (*Round-Robin*)
- Colas múltiples
- Colas múltiples con realimentación.

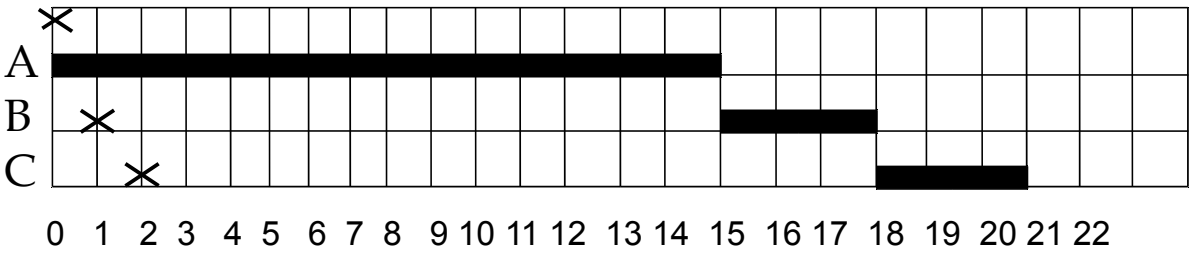
FCFS (*First Come First Served)*

- Los procesos son servidos según el orden de llegada a la cola de ejecutables.
- Es *no apropiativo*, cada proceso se ejecutará hasta que finalice o se bloquee.
- Fácil de implementar pero pobre en cuanto a prestaciones.
- Todos los procesos pierden la misma cantidad de tiempo esperando en la cola de ejecutables independientemente de sus necesidades.
- Procesos cortos muy penalizados.
- Procesos largos poco penalizados.

FCFS (y II)

Procesos	llegada	tiempo CPU	T (t.respuesta)	E (t. espera)
A	0	15	15-0=15	0
B	1	3	18-1=17	14
C	2	3	21-2=19	16

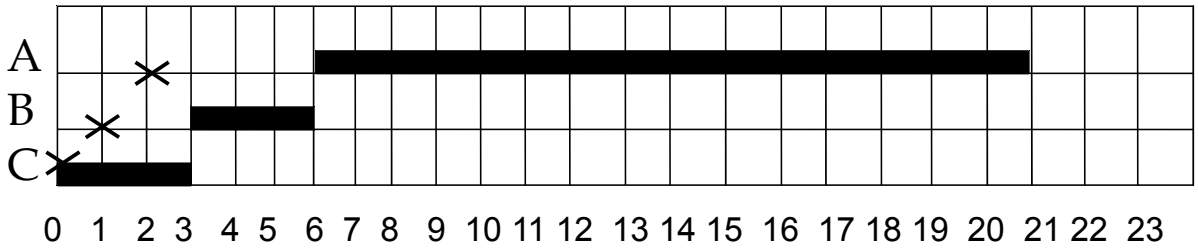
Diagrama de ocupación de la CPU



FCFS (y III)

Procesos	llegada	tiempo CPU	T (t.respuesta)	E (t. espera)
A	2	15	21-2 = 19	4
B	1	3	6-1 = 5	2
C	0	3	3-0 = 3	0

Diagrama de ocupación de la CPU



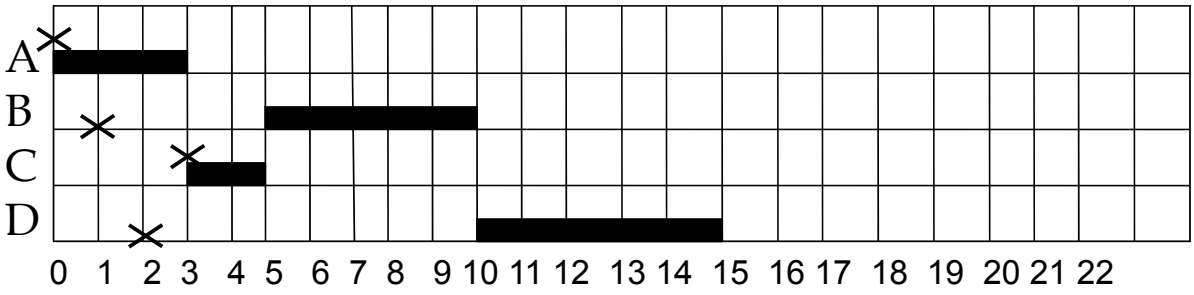
El más corto primero (SJF)

- Es *no apropiativo*.
- Cuando el procesador queda libre, selecciona el proceso que requiera un tiempo de servicio menor.
- Si existen dos o más procesos en igualdad de condiciones, se sigue FCFS.
- Necesita conocer explícitamente el tiempo estimado de ejecución (tº servicio) ¿Cómo? Es necesaria una estimación
- Disminuye el tiempo de respuesta para los procesos cortos y discrimina a los largos.
- Tiempo medio de espera bajo.

El más corto primero (y II)

Procesos	llegada	Ráfaga real	Ráfaga estimada
A	0	3	3
B	1	5	5
C	3	2	2
D	2	5	5

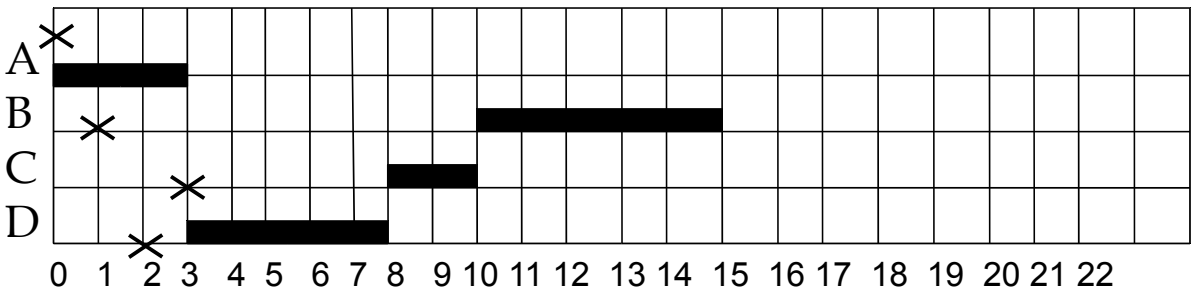
Diagrama de ocupación de la CPU



El más corto primero (y III)

Procesos	llegada	Ráfaga real	Ráfaga estimada
A	0	3	3
B	1	5	6
C	3	2	5
D	2	5	5

Diagrama de ocupación de la CPU



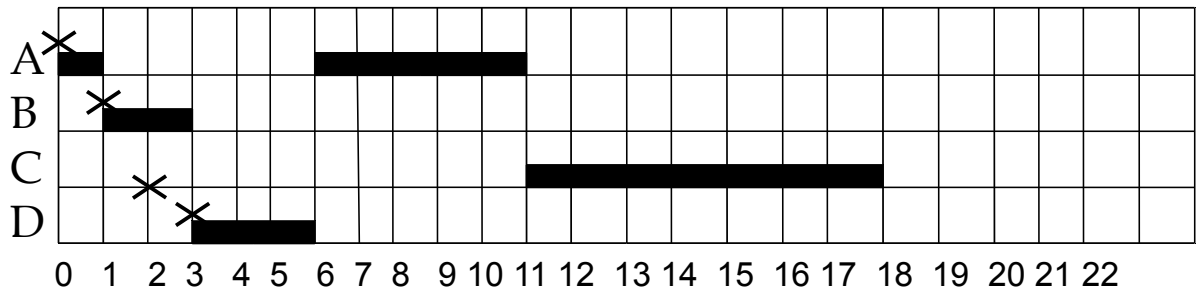
El más corto primero apropiativo (SRTF)

- Cada vez que entra un proceso a la cola de ejecutables se comprueba si su tiempo de servicio es menor que el tiempo de servicio que le queda al proceso que está ejecutándose. Casos:
  - » **Si es menor**: se realiza un cambio de contexto y el proceso con menor tiempo de servicio es el que se ejecuta.
  - » **No es menor**: continúa el proceso que estaba ejecutándose.
- El tiempo de respuesta es menor excepto para procesos muy largos.
- Se obtiene la menor penalización en promedio
- Mantiene la cola de ejecutables con la mínima ocupación posible.

El más corto primero apropiativo (y II)

Procesos	Tº llegada	Ráfaga real	Ráfaga estimada
A	0	6	6
B	1	2	2
C	2	7	7
D	3	3	3

Diagrama de ocupación de la CPU



Planificación por prioridades

- Asociamos a cada proceso un número de prioridad (entero).
- Se asigna la CPU al proceso con mayor prioridad (enteros menores = mayor prioridad)

- **Problema**: Inanición -- los procesos de baja prioridad pueden no ejecutarse nunca.
- **Solución**: Envejecimiento -- con el paso del tiempo se incrementa la prioridad de los procesos.

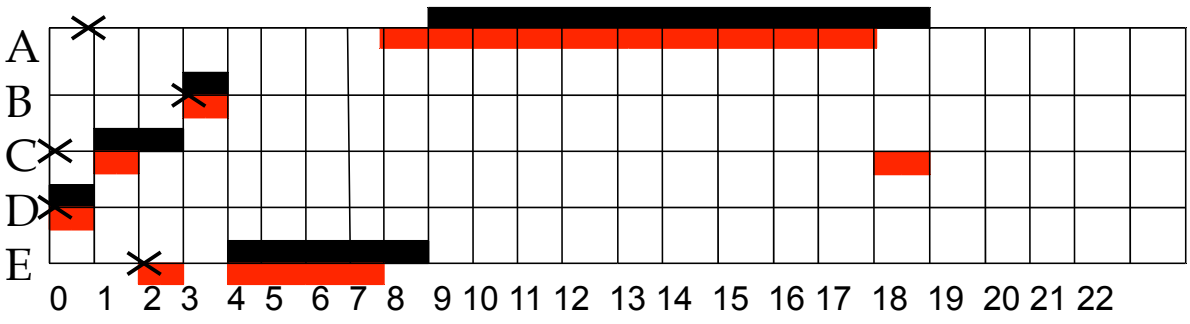
Planificación por prioridades (y II)

Procesos	Tº llegada	Ráfaga	Prioridad
A	1	10	3
B	3	1	1
C	0	2	3
D	0	1	2
E	2	5	2

No apropiativo

Apropiativo

Diagrama de ocupación de la CPU



Por Turnos (Round-Robin)

- La CPU se asigna a los procesos en intervalos de tiempo (quatum).
- Procedimiento:
  - » Si el proceso finaliza o se bloquea antes de agotar el quantum, libera la CPU. Se toma el siguiente proceso de la cola de ejecutables (la cola es FIFO) y se le asigna un quantum completo.
  - » Si el proceso no termina durante ese quantum, se interrumpe y se coloca al final de la cola de ejecutables.
- Es apropiativo.

***Nota**: En los ejemplos* supondremos que si un proceso *A* llega a la cola de ejecutables al mismo tiempo que otro *B* agota su quantum, la llegada de *A* a la cola de ejecutables ocurre antes de que *B* se incorpore a ella.

Por Turnos (y II)

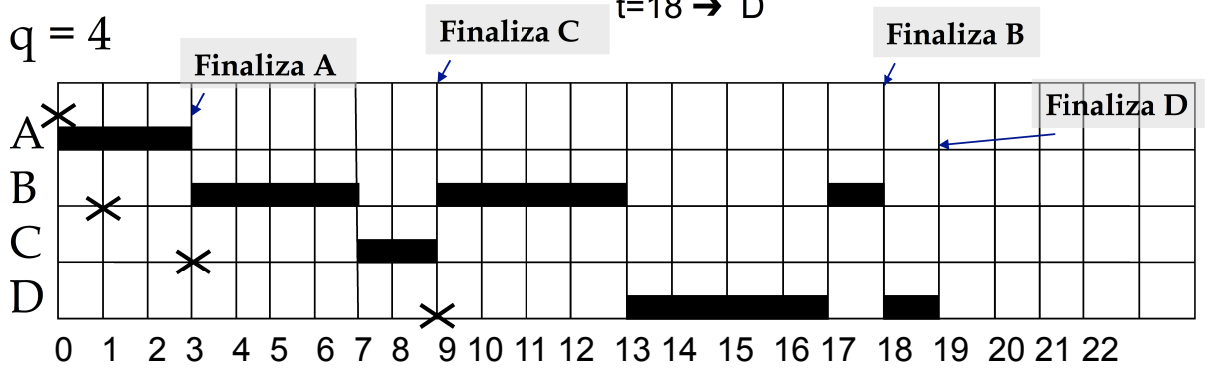
Procesos	Tº llegada	Ráfaga
A	0	3
B	1	9
C	3	2
D	9	5

Estado de la cola de ejecutables

t=0 ➔ A; t=3 ➔ BC; t=7 ➔ CB

t=9 ➔ BD; t=13 ➔ DB; t=17 ➔ BD

t=18 ➔ D



Por Turnos (y III)

- Los valores típicos del quantum están entre 1/60sg y 1sg.
- Penaliza a todos los procesos en la misma cantidad, sin importar si son cortos o largos.
- Las ráfagas muy cortas están más penalizadas de lo deseable.
- ¿valor del quantum?
  - muy grande (excede del tº de servicio de todos los procesos) ➔ se convierte en FCFS
  - muy pequeño ➔ el sistema monopoliza la CPU haciendo cambios de contexto (tº del núcleo muy alto)

Colas múltiples

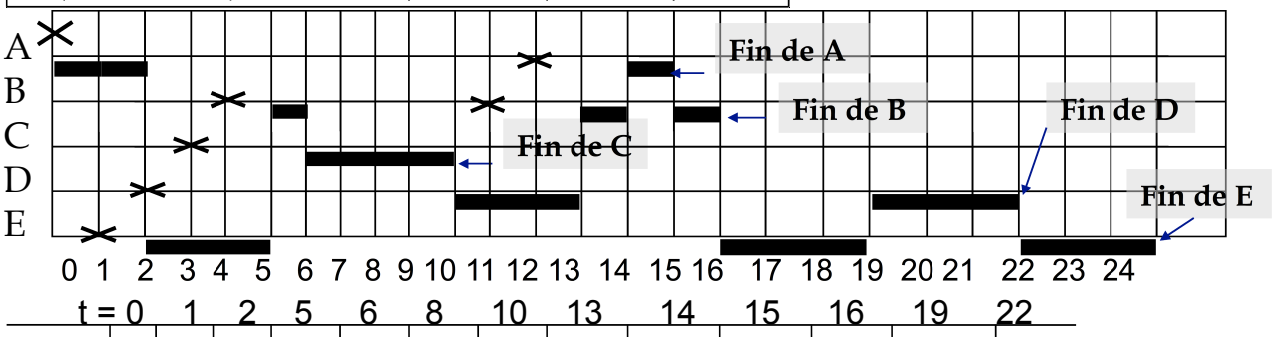
- La cola de preparados se divide en varias colas y cada proceso es asignado permanentemente a una cola concreta P. ej. interactivos y batch
- Cada cola puede tener su propio algoritmo de planificación P. ej. interactivos con RR y batch con FCFS
- Requiere una planificación entre colas
  - » Planificación con prioridades fijas. P. ej. primero servimos a los interactivos luego a los batch
  - » Tiempo compartido -- cada cola obtiene cierto tiempo de CPU que debe repartir entre sus procesos. P. ej. 80% interactivos en RR y 20% a los batch con FCFS

Colas múltiples (y II)

P	TºLlegada	Ráfaga	Bloqueo	Ráfaga	Cola
A	0	2	10	1	1
B	**4**	**1**	**5**	**2**	**1**
C	3	4	-	-	2
D	**2**	**6**	**-**	**-**	**3**
E	1	9	-	-	3

El algoritmo de cada cola es **RR** con **q=1, 2 y 3** ms respectivamente.

El algoritmo entre colas es **prioridades no apropiativo**



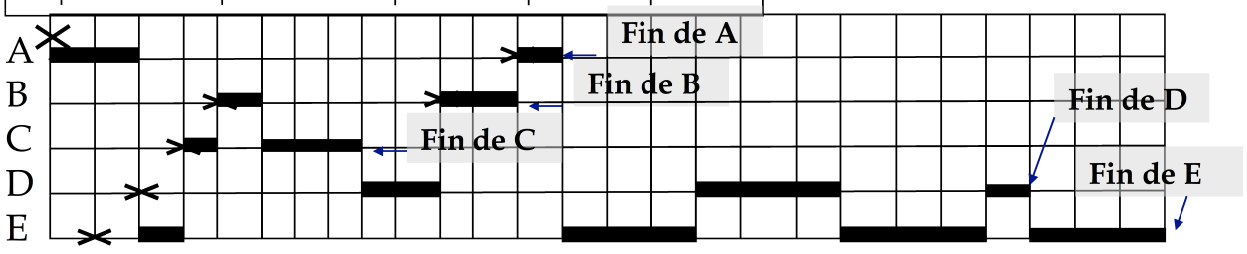
	t = 0	1	2	5	6	8	10	13	14	15	16	19	22
Cola 1	A	A		B				BA	AB	B			
Cola 2				C	C	C							
Cola 3		E	ED	DE	DE	DE	DE	ED	ED	ED	ED	DE	E

Colas múltiples (y III)

P	TºLlegada	Ráfaga	Bloqueo	Ráfaga	Cola
A	0	2	10	1	1
B	**4**	**1**	**5**	**2**	**1**
C	3	4	-	-	2
D	**2**	**6**	**-**	**-**	**3**
E	1	9	-	-	3

El algoritmo de cada cola es **RR** con **q=1, 2 y 3** mlsg. respectivamente.

El algoritmo entre colas es **prioridades *apropiativo.***



	t = 0	1	2	3	4	5	7	8	10	11	12	13	16	19	22	23
Cola 1	A	A			B				B	B	A					
Cola 2				C	C	C	C									
Cola 3		E	ED	DE	DE	DE	DE	DE	ED	ED	ED	ED	DE	ED	DE	E

Colas múltiples con realimentación

- Un proceso se puede mover entre varias colas
- Requiere definir los siguientes parámetros:
  - » Número de colas
  - » Algoritmo de planificación para cada cola
  - » Método utilizado para determinar cuando trasladar a un proceso a otra cola
  - » Método utilizado para determinar en qué cola se introducirá un proceso cuando necesite un servicio
  - » Algoritmo de planificación entre colas
- Mide en tiempo de ejecución el comportamiento real de los procesos
- Disciplina de planificación más general (Unix, Windows NT)

Ejemplo de colas múltiples con realimentación

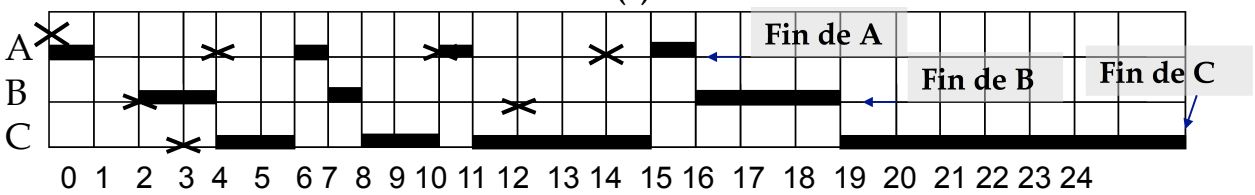
- Tres colas gestionadas mediante **Round Robin**:
  - » Cola 1 con quantum = 2 milisegundos
  - » Cola 2 con quantum = 4 milisegundos
  - » Cola 3 con quantum = 8 milisegundos
- Algoritmo entre colas: **prioridades no apropiativo**, cola 1 mayor prioridad, cola 3 menor prioridad.
- Los procesos entran inicialmente en la cola 1
- Cuando un proceso se bloquea, al regresar a la cola de ejecutables sigue en la misma cola si no hay traspaso
- Un proceso se traspasa a una cola de menor prioridad cuando agota dos quantum de tiempo seguidos, o bien, agota uno y se bloquea en el siguiente

Ejemplo de colas múltiples con realimentación (y II)

P	TºLlegada	TºServicio total	Ráfaga		Bloqueo
A	0	4	1	3	
B	2	6	3	4	
C	3	23	-	-	

(*) Los procesos tienen un comportamiento **cíclico**

(*)



t =	0	1	2	4	6	7	8	10	11	15	16	19	23
Cola 1	A		B	CAB	ABC	BC	C	A		A			
Cola 2								C	C	BC	BC	C	
Cola 3												C	

Operaciones sobre procesos
Creación de procesos

- ¿Qué significa crear un proceso?
  - » Asignarle el espacio de direcciones que utilizará
  - » Crear las estructuras de datos para su administración
- ¿Cuándo se crea? Los sucesos comunes son:
  - » En sistemas batch: en respuesta a la recepción y admisión de un trabajo
  - » En sistemas interactivos: cuando el usuario se conecta, el SO crea un proceso que ejecuta el intérprete de órdenes
  - » El SO puede crear un proceso para llevar a cabo un servicio solicitado por un proceso de usuario
  - » Un proceso puede crear otros procesos formando un árbol de procesos. Hablamos de relación padre-hijo (creador-creado)
- En general, el hijo comparte un subconjunto de los recursos del padre.

Creación de procesos (y III)

- Ejecución (posibilidades):
  - » Padre e hijo se ejecutan concurrentemente (UNIX).
  - » Padre espera al que el hijo termine.
- Ejemplo: En UNIX
  - » La llamada al sistema **`fork`** (bifurcar) crea un nuevo proceso.
  - » El espacio de direcciones del hijo inicialmente es un duplicado del espacio de direcciones del padre
  - » La llamada **`exec`** después de `fork` reemplaza el espacio de direcciones con el programa del nuevo proceso.

Creación de procesos (y IV)

- Por tanto, ¿qué pasos, en general, deben realizarse en una operación de creación?
  - » Nombrar al proceso: asignarle un PID único
  - » Asignarle espacio (en MP y/o memoria secundaria)
  - » Crear el PCB e inicializarlo
  - » Insertarlo en la Tabla de procesos y ligarlo a la cola de planificación correspondiente
  - » Determinar su prioridad inicial

Terminación de procesos

- ¿Qué sucesos determinan la finalización de un proceso?
  - » Cuando un proceso ejecuta la última instrucción, solicita al SO su finalización (**exit**)
    - – Envío de datos del hijo al padre
    - – Recursos del proceso son liberados por el SO
  - » El padre puede finalizar la ejecución de sus hijos (**abort o kill**)
    - – El hijo ha sobrepasado los recursos asignados
    - – La tarea asignada al hijo ya no es necesaria
    - – El padre va a finalizar: el SO no permite al hijo continuar (terminación en cascada)
  - » El SO puede terminar la ejecución de un proceso porque se hayan producido errores o condiciones de fallo

Hebras

(*threads*, hilos o procesos ligeros)

- Una *hebra* (o *proceso ligero*) es la unidad básica de utilización de la CPU. Consta de:
  - » Contador de programa.
  - » Conjunto de registros.
  - » Espacio de pila.
  - » Estado
- Una hebra comparte con sus hebras pares....
  - » Sección de código.
  - » Sección de datos.
  - » Recursos del SO (archivos abiertos, señales,..).
- Un *proceso pesado* o tradicional es igual a una tarea con una hebra.

Tipos de hebras

Tipos de hebras: *Núcleo (Kernel) , de usuario y enfoques híbridos*

- *Hebras de usuario*
  - » Todo el trabajo de gestión de hebras lo realiza la aplicación, el núcleo no es consciente de la existencia de hebras.
  - » Se implementan a través de una biblioteca en el nivel usuario. La biblioteca contiene código para gestionar las hebras:
    - – crear hebras, intercambiar datos entre hebras, planificar la ejecución de las hebras y salvar y restaurar el contexto de las hebras.
  - » La unidad de planificación para el núcleo es el proceso

Tipos de hebras (y II)

- *Hebras Kernel (Núcleo)*
  - » Toda la gestión de hebras lo realiza el núcleo.
  - » El SO proporciona un conjunto de llamadas al sistema similares a las existentes para los procesos (Mach, OS/2).
  - » El núcleo mantiene la información de contexto del proceso como un todo y de cada hebra.
  - » La unidad de planificación es la hebra.
  - » Las propias funciones del núcleo pueden ser multihebras.

Tipos de hebras (y III)

- Ventajas del uso de las hebras tipo usuario frente a las de tipo núcleo:
  - Se evita la sobrecarga de cambios de modo, que sucede cada vez que se pasa el control de una hebra a otra en sistemas que utilizan hebras núcleo.
  - Se puede tener una planificación para las hebras distinta a la planificación subyacente del SO.
  - Se pueden ejecutar en cualquier SO. Su utilización no supone cambio en el núcleo.

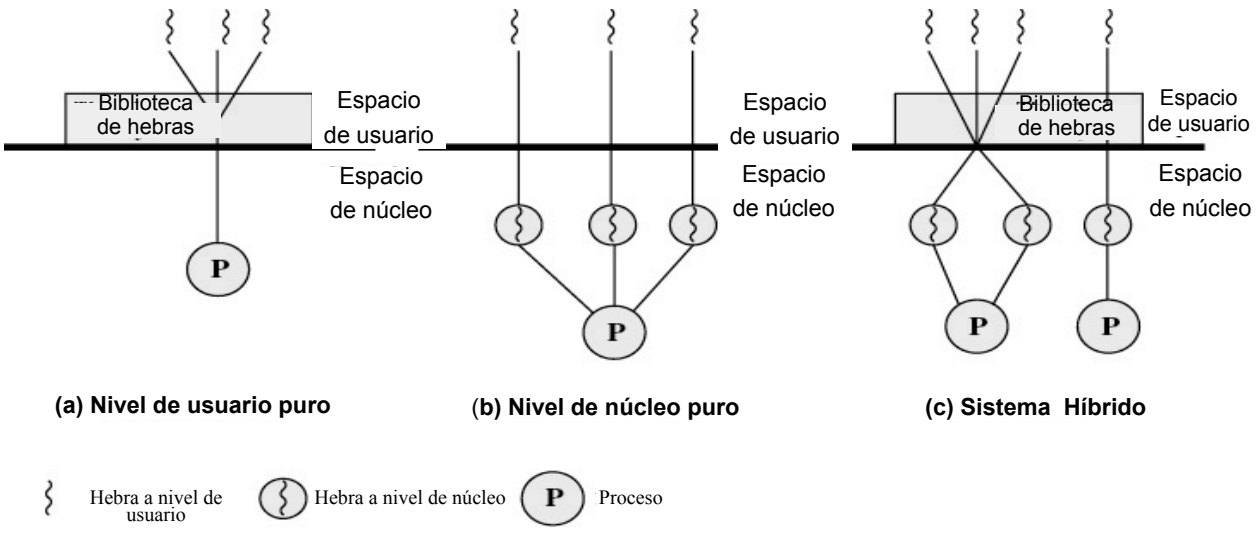
Tipos de hebras (y IV)

- Desventajas del uso de las hebras tipo usuario frente a las de tipo núcleo.
  - Cuando un proceso realiza una llamada al sistema bloqueadora no sólo se bloquea la hebra que realizó la llamada, sino todas las hebras del proceso.
  - En un entorno multiprocesador, una aplicación mutihebra no puede aprovechar la ventajas de dicho entorno ya que el núcleo asigna un procesador a un proceso.

Tipos de hebras (y V)

- *Enfoques híbridos*
  - » Implementan tanto hebras a nivel *kernel* como usuario (p. ej. Solaris 2).
  - » La creación de hebras, y la mayor parte de la planificación y sincronización se realizan en el espacio de usuario.
  - » Las distintas hebras de una aplicación se asocian con varias hebras del núcleo (mismo o menor número), el programador puede ajustar la asociación para obtener un mejor resultado.
  - » Las múltiples hebras de una aplicación se pueden paralelizar y las llamadas al sistema bloqueadoras no necesitan bloquear todo el proceso.

Tipos de hebras (y VI)



Hebras a nivel de usuario y a nivel de núcleo (*Stalling*)