

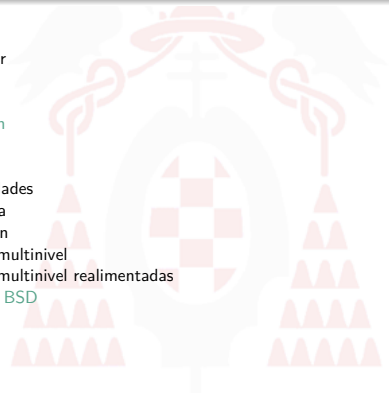
# Planificación del uso de la CPU

*Departamento de Automática*  
Universidad de Alcalá



/gso>

# Índice

- 
- 1 Concepto de planificación
    - Introducción
    - Planificador / dispatcher
    - Tipos de planificadores
  - 2 Criterios de evaluación
  - 3 Algoritmos de planificación
    - Planificación FIFO
    - Planificación SJF
    - Planificación con prioridades
    - Planificación con requisa
    - Planificación round-robin
    - Planificación con colas multinivel
    - Planificación con colas multinivel realimentadas
  - 4 Planificación en UNIX 4.4 BSD
    - Generalidades
    - Colas
    - Cálculo de prioridades
  - 5 Planificación en W2K
    - Generalidades
    - Quantum
    - Ejecución del planificador
    - Escenarios de planificación
    - Ajuste de prioridad

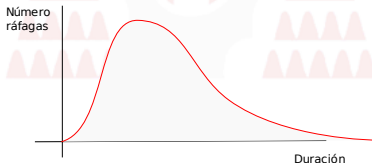
# Concepto de planificación

## Introducción (I)

- La ejecución de un trabajo se compone de secuencias de uso del procesador y de espera
  - La planificación utiliza los periodos de espera para alternar la utilización del procesador entre distintos procesos
- Los objetivos de la planificación son:
  - Equidad.
  - Eficiencia del propio planificador.
  - Bajo tiempo de respuesta (importante en sistemas interactivos y tiempo real).
  - Rendimiento alto.
  - Minimizar el tiempo de espera.
- Todos estos objetivos no se pueden conseguir simultáneamente.

## Introducción (II)

- Los procesos se ejecutan por ráfagas.
- En función del tamaño predominante de la ráfaga se distinguen:
  - Procesos intensivos en CPU.
  - Procesos intensivos en E/S.



# Concepto de planificación

## Planificador / dispatcher (I)

### Planificador (o *scheduler*)

Componente del SO que determina quién es el siguiente trabajo en ocupar la CPU (implementa políticas)

### *Dispatcher* (o repartidor)

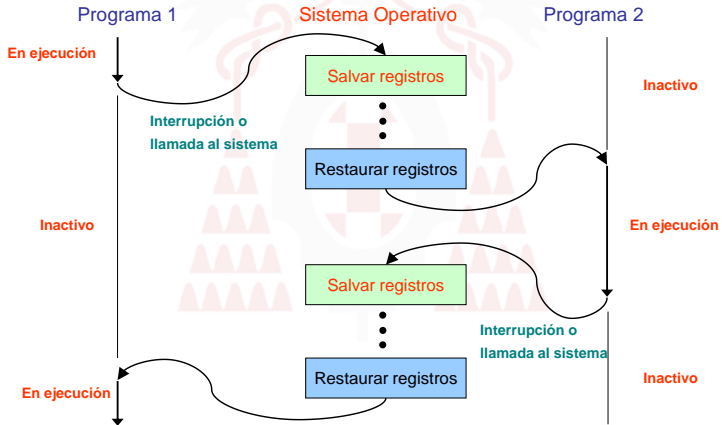
Componente del SO que conmuta el procesador de un trabajo a otro (implementa mecanismos)

Un proceso está ejecutándose en la CPU y se produce un *evento* ...

- Funcionamiento del *dispatcher*:
  - 1 Guarda el estado del proceso en el BCP: CP, PSW, registros, ...
  - 2 Restaura el estado de otro proceso.
  - 3 Transfiere el control al nuevo proceso.

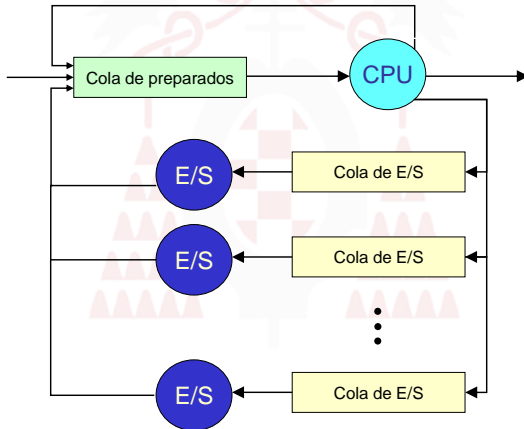
# Concepto de planificación

## Planificador / dispatcher (II)



# Concepto de planificación

## Planificador / dispatcher (III)



# Concepto de planificación

## Tipos de planificadores

- Planificador a largo plazo.
  - Carga el proceso en memoria.
  - Controla el grado de multiprogramación.
  - Es poco utilizado en la actualidad.
- Planificador a corto plazo
  - Selecciona entre los trabajos cargados en memoria, y que están listos para ejecutarse, cuál hará uso del procesador.
  - Debe ser muy rápido.
- Planificador a medio plazo
  - Carga y descarga trabajos activos del disco a memoria y viceversa.
  - Directamente relacionado con la memoria virtual.



## Criterios de evaluación

- *Utilización del procesador.* Porcentaje de tiempo de uso del procesador.
- *Grado de sobrecarga.* Recursos que emplea el planificador.
- *Rendimiento.* Trabajos completados por unidad de tiempo.
- *Tiempo de estancia.* Tiempo desde que se lanza el proceso hasta que se finaliza.
- *Tiempo de espera.* Tiempo que un proceso está en una cola (no está en ejecución).
- *Tiempo de respuesta.* Tiempo en obtener una respuesta del proceso ante un estímulo. Importante en sistemas interactivos o de tiempo real.

# Algoritmos de planificación

## Planificación FIFO (I)

- FIFO: First in first out - primero en entrar, primero en salir.
- Gestiona la cola de procesos listos como una cola FIFO.
- Ventajas
  - Es el algoritmo más sencillo de codificar.
- Inconvenientes:
  - Un proceso puede monopolizar la CPU.
  - **Efecto convoy**: Beneficia a procesos intensivos en CPU.
  - Depende fuertemente del tipo de trabajos y de los instantes de llegada.



# Algoritmos de planificación

## Planificación SJF (I)

- SJF: *Shortest Job First* - primero el más corto.
- Asigna la CPU al trabajo con la siguiente ráfaga más pequeña
- Ventajas:
  - Reduce los tiempos medios de respuesta.
  - Es un algoritmo óptimo.
- Inconvenientes:
  - Exige conocer el futuro, y por lo tanto es irrealizable.
  - Es necesario pronosticar la duración de la siguiente ráfaga.
  - Se puede producir inanición en procesos con ráfagas largas.

# Algoritmos de planificación

## Planificación SJF (II)

- SJF necesita un mecanismo de predicción.
- Un mecanismo es predecir como media exponencial de las longitudes medidas en anteriores ráfagas.

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

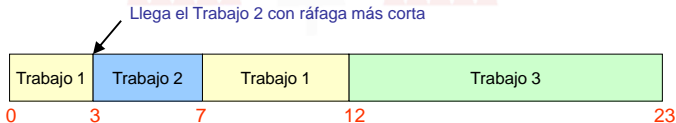
- donde:
  - $t_n$  Longitud de la n-ésima ráfaga de CPU.
  - $\tau_n$  Longitud predicha para la n-ésima ráfaga de CPU.
  - $\alpha$  Parámetro de ajuste.
- Mediante  $\alpha$  se ajusta el peso de la historia reciente.
- Ejemplo:
  - T1: 3 u.t., T2: 12 u.t., T3: 7 u.t., T4: 5 u.t.
  - $\bar{t}_{\text{estancia}} = 13,25$  u.t.

# Algoritmos de planificación

## Planificación con prioridades

- Cada proceso tiene asociada una prioridad.
- La CPU se asigna al proceso con la mayor prioridad.
- Es necesario un mecanismo para asignar las prioridades.
- Clasificación de métodos de asignación de prioridades.
  - Interno o externo.
  - Estático o dinámico.
- Inconveniente: Los procesos pueden sufrir *inanición*.
- Solución: Introducir mecanismos de envejecimiento.

## Planificación con requisa



# Algoritmos de planificación

## Planificación round-robin (I)

- Es característico de los sistemas de tiempo compartido.
- La CPU se asigna a cada proceso listo durante un quantum de tiempo “q” .
  - Evita la monopolización de uso de CPU.
  - El quantum de tiempo se delimita por medio de una interrupción periódica.
- La cola de procesos listos es FIFO
  - Si ráfaga  $> q \Rightarrow$  Interrupción.
  - Si ráfaga  $< q \Rightarrow$  Liberación voluntaria de la CPU.
- Prestaciones: Dependen fuertemente del tamaño de q.
  - Si  $q \rightarrow \infty$ : Round-robin degenera en FIFO.
  - Si  $q \rightarrow 0$ :  $\frac{CPU}{n}$  (n = número de procesos listos).



# Algoritmos de planificación

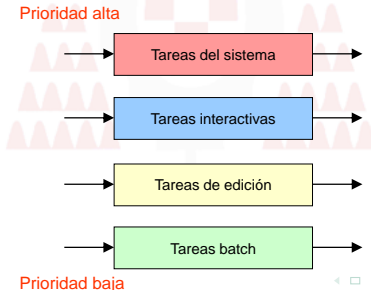
## Planificación round-robin (II)

- El tamaño de “q” es una decisión de diseño.
  - Si “q” es muy pequeño hay muchos cambios de contexto (se pierde eficiencia).
  - Si “q” es grande aumentan los tiempos de respuesta.
- Regla empírica: El 80 % de las ráfagas de CPU deben ser menores que q.
- Problema:
  - Round-robin no distingue entre tipos de trabajos.

# Algoritmos de planificación

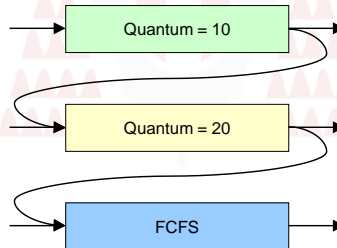
## Planificación con colas multinivel

- Objetivo: Diferenciar entre distintos tipos de trabajos.
- Existen colas separadas en función del tipo de trabajo.
- Cada cola tiene su propio algoritmo de planificación.
- Debe existir otro algoritmo para elegir la cola.



## Planificación con colas multinivel realimentadas

- Los trabajos cambian de prioridad y de cola.
- Nuevos elementos de diseño:
  - Cada cola tiene su propio algoritmo de planificación.
  - Métodos para ascender y descender entre colas.
  - Decisión de dónde incluir inicialmente los trabajos.



# Planificación en UNIX 4.4 BSD

## Generalidades

- Características:
  - Sistema de tiempo compartido.
  - Entorno multiusuario.
  - Aplicaciones interactivas.
- Planificación con prioridades dinámicas [127-0] y desalojo.
  - Procesos en modo supervisor [49-0].
  - Procesos en modo usuario [127-50].
- Existen 32 colas.
  - Cola  $\leftarrow$  Prioridad / 4.
  - Cada cola se planifica con round-robin con  $q = 0, 1s$ .
- El proceso pierde el microprocesador cuando:
  - Llega un proceso con prioridad mayor (requisa).
  - Expira  $q$ .
  - Pasa a espera.

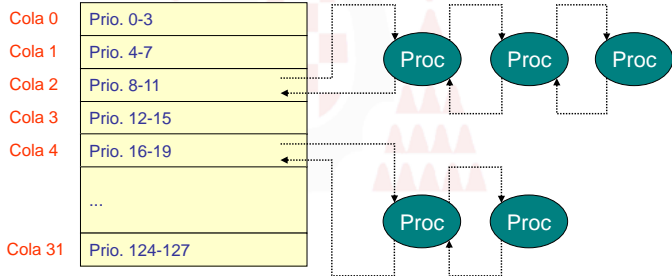
# Planificación en UNIX 4.4 BSD

## Colas

whichqs

0	0	1	0	1	...
---	---	---	---	---	-----

qs



# Planificación en UNIX 4.4 BSD

## Cálculo de prioridades (I)

- Campos relacionados con la planificación:
  - `p_usrpri` = Prioridad de un proceso en modo usuario.
  - `p_estcpu` = Tiempo de procesador acumulado.
  - `p_slptime` = Tiempo bloqueado por algún evento.
  - `p_nice` = Valor que puede modificar el usuario (+-20).

## Planificación en UNIX 4.4 BSD

### Cálculo de prioridades (II)

- Cada cuatro ticks (40 ms) se calcula la prioridad:

$$p\_usrpri = PUSER + \left\lceil \frac{p\_estcpu}{4} \right\rceil + 2 * p\_nice$$

PUSER=50

- Para evitar que  $p\_usrpri = 127$ , cada segundo se ejecuta

$$p\_estcpu = decay * p\_estcpu + p\_nice$$

$$decay = \frac{(2 * carga)}{(2 * carga + 1)}$$

- Cada vez que un proceso se desbloquea:

$$p\_estcpu = decay^{p\_slptime} * p\_estcpu$$

- Por cada segundo bloqueado:  $p\_slptime = p\_slptime + 1$

# Planificación en W2K

## Generalidades

- Características:
  - Sistema de tiempo compartido.
  - Entorno monousuario.
  - Aplicaciones interactivas o como servidor.
  - Planificación de hilos.
- Algoritmo con requisa y prioridades.
- Tareas de tiempo real [16- 31].
- Tareas ordinarias [1 - 15].
  - Prioridades dinámicas:  $\text{base} + \text{offset}$ .
- Hilos con prioridad 0:
  - *Zero page thread*.
  - Proceso inactivo del sistema (*idle thread*).



# Planificación en W2K

## Quantum (I)

- La unidad de medida básica es el *quantum*.
- Un quantum está dividido en unidades lógicas (u.l.).
  - En cada tick se restan 3 u.l. al *quantum* ( $q=q-3$ ).
  - El número de u.l. por *quantum* depende de la versión de Windows.
- W2K Professional:
  - $q = 6u.l. \implies 2ticks$ .
  - Mejor interactividad.
- W2K Server:
  - $q = 36u.l. \implies 12ticks$ .
  - Mejor rendimiento.
- El valor del *quantum* se pueden modificar con el registro de Windows.

# Planificación en W2K

## Quantum (II)

- La duración de un tick depende de la arquitectura.
  - Monoprocesador i80x86: 10 ms por tick.
  - Multiprocesador i80x86: 15 ms por tick.
- Ensanchamiento de q:
  - Para la tarea que se ejecuta en primer plano aumenta el tamaño de q.
  - Pretende mejorar el tiempo de respuesta.
  - El quantum puede aumentar en 2, 4 ó 6 ticks.
  - El ensanchamiento de q no se aplica a Windows Server.

# Planificación en W2K

## Ejecución del planificador

- El planificador se ejecuta en las siguientes circunstancias:
  - El hilo libera la CPU:
    - Finalización del quantum.
    - El hilo pasa a espera, por ejemplo, al iniciar una operación E/S.
    - El hilo finaliza su ejecución.
  - Un hilo pasa a estado de listo:
    - El hilo acaba de ser creado.
    - El hilo finaliza una operación de E/S.
  - Cambia la prioridad del hilo.
  - Cambia la afinidad del procesador.

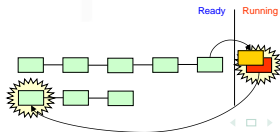
# Planificación en W2K

## Escenarios de planificación (I)

- Conmutación voluntaria:
  - El hilo en ejecución pasa a estado de espera y se elige el primer trabajo de la cola de mayor prioridad.
- Finalización del quantum:
  - Si la prioridad del hilo es la prioridad base.



- Si ha habido aumento de prioridad, se reduce su prioridad





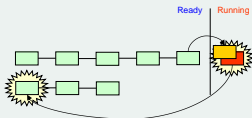
# Planificación en W2K

## Ajuste de prioridad (I)

- Se priorizan los hilos intensivos en E/S.
- Las prioridades pueden aumentar (*boost*) o disminuir (*decay*).

### Priority decay

- La prioridad se reduce en uno si el hilo agota su quantum.
- Un hilo no puede tener menos prioridad que su prioridad base.



### Priority boosting

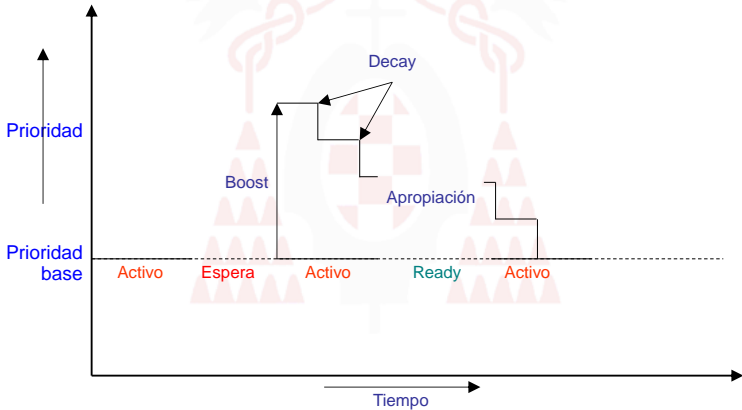
- Se aplica a la prioridad base de los hilos ordinarios que dejan de estar bloqueados.
- La prioridad de un proceso no puede ser superior a 15.
- Se aumenta la prioridad en función de qué causó el bloqueo.

# Planificación en W2K

## Ajuste de prioridad (II)

- El aumento de prioridad depende del motivo del bloqueo.
  - Fin de E/S. Depende del dispositivo:
    - Tarjeta de sonido  $\Rightarrow +8$ .
    - Teclado y ratón  $\Rightarrow +6$ .
    - Red y puerto serio  $\Rightarrow +2$ .
    - Disco, CD-ROM, tarjeta gráfica, puerto paralelo  $\Rightarrow +1$ .
  - Otros motivos (semáforos, eventos, etc.).
- Para evitar la inanición se ejecuta el *Balance Set Manager*.
  - Se ejecuta cada segundo con prioridad 16 (hilo del sistema).
  - Busca hilos con más de 300 ticks sin ejecutarse.
    - Duplica el ancho del quantum.
    - Aumenta la prioridad en +10.

## Planificación en W2K





# Referencias bibliográficas I



[Sánchez, 2005] S. Sánchez Prieto.

*Sistemas Operativos.*

Servicio de Publicaciones de la UA, 2005.



[Tanenbaum, 2009] A. Tanenbaum.

*Sistemas Operativos Modernos.*

Ed. Pearson Education, 2009.



[Stallings, 1999] W. Stallings.

*Organización y arquitectura de Computadores.*

Ed. Prentice Hall, 1999.