Planificación en Servidores de Información Multimedia

Servidores de Información Multimedia

2º Ingeniero de Telecomunicación (Esp. Sonido e Imagen)

Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid

2 Índice

- Recordatorio
- Estados de los procesos
- Conceptos en planificación de procesos
- Algoritmos de planificación de procesos

3 Bibliografía

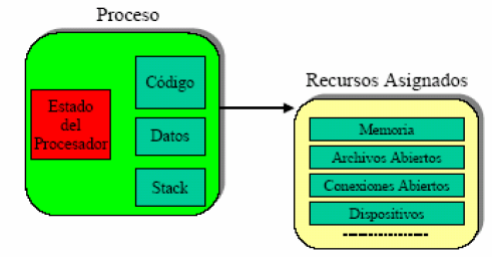
- A. Silberschatz, P. B. Galvin. sistema Operativos. 5ª ed. [Tema 4 y 5 (Hilos)]
- • W. Stallings. sistema Operativos, 4ª ed. [Tema 3, 4 y 9]

4 Recordatorio

- Concepto de Proceso:
  - Primeros sistemas sólo permitían la ejecución de un programa a la vez que tenía disponibles todos los recursos del ordenador
  - Hoy, los sistemas operativos permiten cargar varios programas en memoria y ejecutarlos concurrentemente
- **proceso**
  - Programa en Ejecución
  - Unidad de trabajo de un SO
  - Puede haber dos procesos asociados al mismo programa: 2 copias en ejecución del mismo programa → Son secuencias de ejecución independientes

5 Recordatorio

- Componentes necesarios para gestionar un proceso:
  - Contenido del programa en ejecución
    - Sección de Código: sección de texto
    - Sección de Datos: variables globales
    - Sección de Pila: variables locales, paso de parámetros y dirección de retorno
  - Estado de la CPU: PC y registros de datos/direcciones
  - Recursos del sistema: ficheros, dispositivos E/S

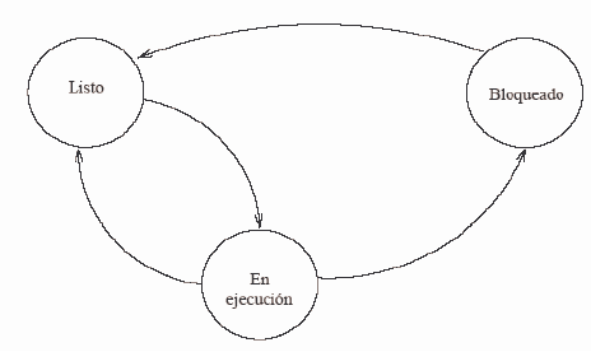


6 Estados de un proceso

- Un proceso, a lo largo de su existencia, pasa por diferentes estados
- El estado actual de un proceso depende de la actividad realizada hasta ahora y del entorno
- A medida que se ejecuta un proceso, cambia su estado
- Estados de un proceso (modelo básico):
  - **Ejecutándose**: Proceso ejecutando instrucciones (tiene asignado la CPU)
  - **Listo**: El proceso está listo para recibir el procesador para iniciar o continuar su ejecución
  - **En Espera**: El proceso deja de competir por el procesador, esperando un evento externo (p.e. terminación de una operación de E/S, sincronización con otro proceso, una señal, etc.)

7 Estados de un proceso

- Modelo básico:



- Cada vez que un proceso cambia de estado se implica al **planificador**.

8 Transiciones entre estados

- **En ejecución → Bloqueado:** Se produce cuando un proceso no puede continuar su ejecución porque debe esperar un evento
- **En ejecución → Listo:** El planificador decide que el proceso ya ha agotado el tiempo de ejecución que le corresponde y lo sustituye
- **Listo → En ejecución**: El planificador selecciona al proceso para ser cargado en la CPU y ejecutado
- **Bloqueado → Listo:** El evento que estaba esperando el proceso bloqueado ha sucedido. El proceso pasa al estado listo para continuar su ejecución cuando lo indique el planificador

9 Procesos suspendidos

- Los 3 estados principales (Listo, Ejecución, Bloqueado) pueden no ser suficientes
- Justificación:
  - Si todos los procesos están en bloqueados esperando un suceso y no hay memoria disponible para nuevos procesos → el procesador estará desocupado
- Una solución:
  - Procesos suspendidos

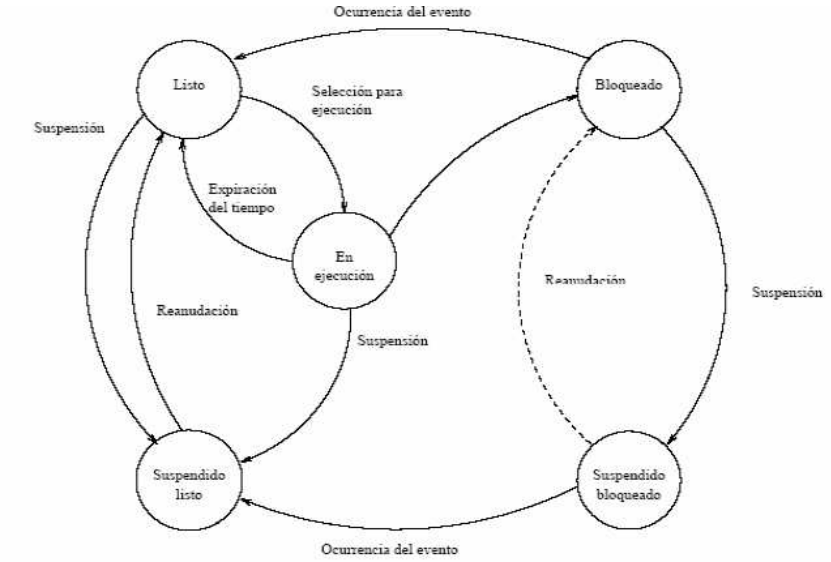
10 Procesos suspendidos (I)

- Solución al problema anterior: Permitir la ejecución de más procesos
- Requisito: Ampliar la memoria principal o suspender procesos a disco duro
- Implementación práctica:
  - **Intercambio** de procesos entre memoria y disco
  - Nuevos estados de un proceso: **Listo suspendido y Bloqueado suspendido**
- **Listo suspendido**: el proceso está suspendido, pero se encuentra listo para ejecutarse
- **Bloqueado suspendido**: el proceso está suspendido y además está esperando que suceda un evento

11 Procesos suspendidos (II)

- Método (intercambio o swapping):
  - El sistema operativo puede suspender un proceso y transferirlo a disco duro
  - El espacio liberado en la memoria principal es usado para traer otro proceso
  - ¿Qué proceso elegir para cargar en memoria?
    - Uno proceso nuevo
    - Uno previamente suspendido (debemos elegir los que se encuentran en listo suspendido y no en bloqueado suspendido)

12 Estados de un proceso



13 Nuevas transiciones

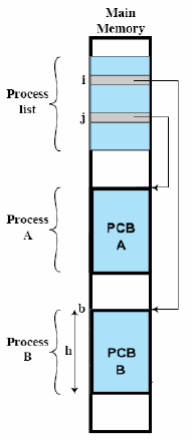
- **En ejecución → Suspendido listo:** Se suspende el proceso actual en ejecución
- **Listo → Suspendido listo:** Se produce la suspensión del proceso en estado listo por parte del sistema operativo
- **Bloqueado → Suspendido bloqueado**: Se suspende el proceso que se encontraba a la espera de un evento
- **Suspendido listo → Listo:** El proceso es reanudado por el sistema operativo y pasa a la cola de procesos listos para ejecución
- **Suspendido bloqueado → Bloqueado**: El proceso es reanudado antes de que el evento que espera suceda. Esta transición no se da en algunos sistemas, cuando un proceso bloqueado se suspende no se reanuda hasta que el evento suceda, y entonces pasa a Suspendido listo
- **Suspendido bloqueado → Suspendido listo**: El evento que se esperaba sucede. El proceso no se reanuda sino que se queda en Suspendido listo. Cuando se reanude ya no estará bloqueado

14 Otras razones para suspender un proceso

- Otras razones por las que un proceso puede pasar a suspendido:
  - El sistema está en riesgo de fallo. El sistema suspende todos los procesos activos para poder corregir errores y volver a activarlos cuando el sistema funcione correctamente
  - Un proceso sospechoso de mal funcionamiento puede ser suspendido hasta verificar su correcto funcionamiento
  - El planificador puede suspender los proceso de baja prioridad en momento de carga excesiva del sistema

15 Contexto de un proceso (I)

- **Tabla de procesos y Bloque de control de Proceso**
  - El sistema operativo gestiona los procesos a través de una tabla que contiene información sobre todos los procesos en el sistema
  - Esa tabla es una estructura de datos localizada en una zona de memoria perteneciente al núcleo del sistema operativo
  - Cada proceso se representa mediante una estructura de datos llamada (en Linux) **PCB (Bloque de Control de Proceso)**
    - **En Unix se llama process structure que apunta a un user structure entre otros datos que también contiene datos del PCB**
    - En Windows se tienen varios objetos:
      - Executive Process Block (**EPROCESS**)
      - Kernel Process Block (**KPROCESS**)
      - Process Environment Block (**PEB**)

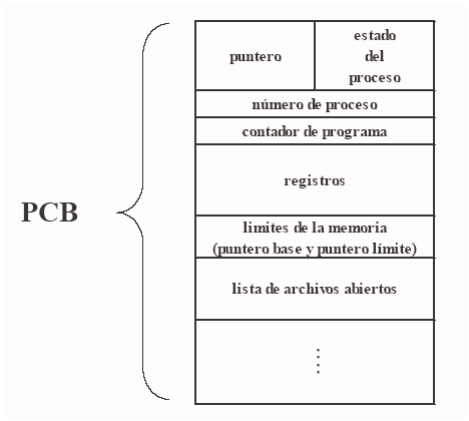


16 Contexto de un proceso (II)

- **Bloque de control de proceso (PCB):**
  - **Identificación del Proceso** (número único: PID)
  - **Estado del Proceso** (Ejecutándose, listo, bloqueado, etc.)
  - **Contador de programa PC** (Dir. próxima instrucción a ejecutar)
  - **Registros de CPU** (para guardar los registros de datos/direcciones)
  - **Planificación de CPU** (prioridades, punteros a colas de planificación y otros parámetros)
  - **Administración de Memoria** (registros base y límite, información sobre como se organiza la memoria, etc.)
  - **Estadísticas** (CPU usada, límites de tiempo, etc.)
  - **Estado de recursos** (Lista de recursos asignados y estado: ficheros, dispositivos de E/S)

17 Contexto de un proceso (III)

- PCB



18 Contexto de un proceso (IV)

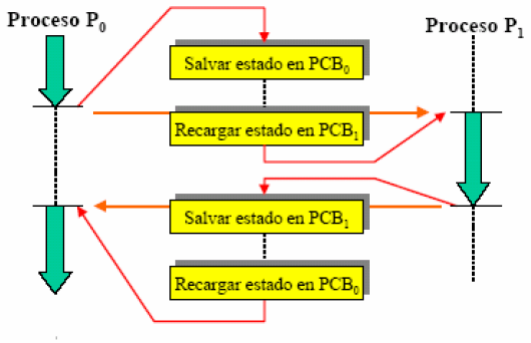
- En Linux por ejemplo se tiene definido el PCB en el fichero sched.h:

```
/* memory management info */
struct mm_struct *mm;
/* open file information */
struct files_struct *files;
/* tss for this task */
struct thread_struct tss;
int pid;
volatile long state; /* -1
unrunnable, 0 runnable, >0 stopped
*/
long priority;
unsigned short uid,euid,suid,fsuid;
#ifdef __SMP__
int processor;
#endif
struct task_struct *p_opptr, *p_pptr,
*p_cptr, *p_ysptr, *p_osptr;
/* limits */
struct rlimit rlim[RLIM_NLIMITS];
long utime, stime, cutime, cstime,
start_time;
```

19 Cambios de contexto (recordemos)

- Cuando un proceso esta ejecutándose, los valores del PC, otros registros de datos/direcciones, puntero a pila, etc., es decir, su contexto, están cargados en los registros de la CPU
- Cuando el SO detiene la ejecución de un proceso, salva su contexto en su PCB
- La acción de conmutar la CPU de un proceso a otro se denomina cambio de contexto
- Los sistemas de tiempo compartido realizan de cientos/miles de cambios de contexto por segundo.
  - Un cambio de contexto suele tardar de 1 a 1000 [µseg]
  - El cambio de contexto no correponde a trabajo productivo, sino que es burocracia (overhead)

20 Cambios de contexto (recordemos)



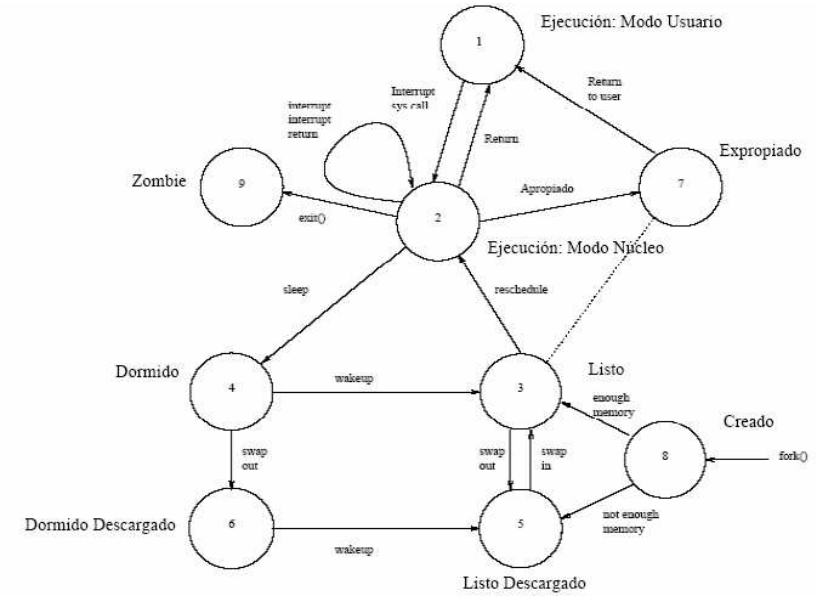
21 Tipos de cambio de contexto

- Cambio de contexto "voluntario" :
  - Proceso realiza llamada al sistema que implica esperar por evento
  - Transición de *en ejecución* a *bloqueado*
  - Ejemplos: iniciar una operación de E/S
  - Motivo: Eficiencia en el uso del procesador
- – Cambio de contexto "involuntario" :
  - SO le quita la CPU al proceso
  - Transición de *en ejecución* a *listo*
  - Ejemplos: fin del tiempo de ejecución
  - Motivo: Reparto del procesador

22 Ejemplo - UNIX

- **Estados de un Proceso en Unix**
- Se amplían los posibles estados a nueve:
  - Ejecutando en modo usuario
  - Ejecución en modo Kernel
  - Listo (tras dejar CPU de forma voluntaria)
  - Expulsado/Expropiado (deja la CPU de forma NO voluntaria)
  - Dormido: (Bloqueado)
  - Listo Descargado (listo suspendido)
  - Dormido Descargado: (Bloqueado suspendido)
  - Creado
  - Zombie

23 Ejemplo - UNIX



24 Cambios de estado en UNIX

- **Ejecutando en modo Usuario → Ejecutando en modo Kernel**: El proceso ejecuta una llamada al Sistema (se produce un *trap*) o se produce un interrupción
- **Ejecutando en modo Kernel → Ejecutando en modo Usuario**: Tras la llamada al sistema se continua ejecutando el proceso en modo Usuario
- **Ejecutando en modo Kernel → Expropiado**: Tras el fin de una llamada al sistema el planificador puede decidir que el proceso actual no debe continuar ejecutandose. Es un estado similar a listo para ejecutar.
- **Ejecutando en modo Kernel → Zombie**: Un proceso quiere finalizar con la llamada *exit().* Pasa a modo kernel y se finaliza el proceso
- **Ejecutando en modo Kernel → Dormido**: El proceso queda a la espera de un evento (p.e. Operación de E/S)

25 Cambios de estado en UNIX

- **Ejecutando en modo Kernel → Ejecutando en modo Kernel**: Durante la ejecución del proceso en modo Kernel se puede producir una interrupción. No es un cambio de estado sino un cambio de proceso
- **Listo → Ejecutando en modo Kernel**: Un proceso listo para ejecutar vuelve a tomar el control del procesador y lo hace en modo Kernel para terminar la llamada al sistema que estaba realizando y fue interrumpida por la espera de un evento.
- **Listo → Listo Descargado**: El sistema tiene demasiados proceso y todos no caben en memoria. Lo descarga a disco (swapping)
- **Dormido → Listo**: El evento que esperaba el proceso se ha producido
- **Dormido → Dormido Descargado**: Un proceso que espera un evento es descargado a disco para liberar espacio en memoria para otros procesos
- **Listo Descargado → Listo**: Se carga el proceso descargado en la memoria

26 Cambios de estado en UNIX

- **Dormido Descargado → Listo Descargado**: El evento que esperaba un proceso descargado se ha producido, pasa a listo, pero todavía sigue descargado
- **Expropiado → Ejecutando en modo Usuario**: El planificador decide devolver el control a un proceso expropiado
- **Creado → Listo**: Cuando el proceso se crea y hay memoria para cargarlo en memoria pasa a Listo
- **Creado → Listo Descargado**: Si cuando se crea un proceso no hay memoria suficientes este se crea descargado

27 Regiones de un proceso en UNIX

- Unix considera que un proceso consta de tres partes o regiones:
  - **Text** (bloque de texto) : contiene instrucciones
  - **Data** (bloque de datos): contiene datos
  - **Stack** (bloque de pila): pila para llamadas a funciones y procedimientos

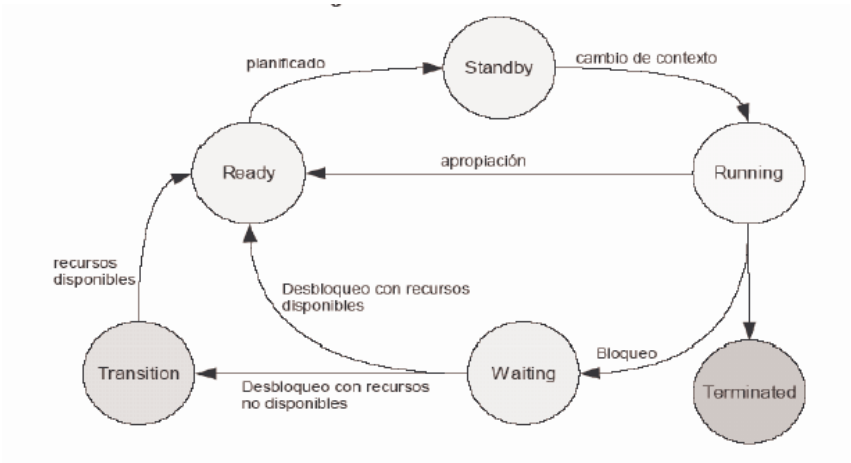
28 Ejemplo - Linux

- Estados de un proceso en Linux

29 Estados de un proceso en Linux

- **TASK_RUNNING:** Proceso listo o en ejecución
- **TASK_ INTERRUPTIBLE:** Proceso "dormido" que puede despertar por alguna señal o interrupción
- **TASK_ UNINTERRUPTIBLE:** Similar al anterior, pero que no puede ser despertado inmediatamente, espera a una interrupción y no puede ser despertado por una señal (el proceso está *suspendido*)
- **TASK_ ZOMBIE:** Proceso-hijo terminado pero que no ha sido liberado por su proceso padre
- **TASK_ STOPPED:** Proceso detenido, generalmente por una señal (SIGSTOP). Útil para depuración

30 Ejemplos – Windows 2000



31 Estados de un hilo de ejecución en Win2K

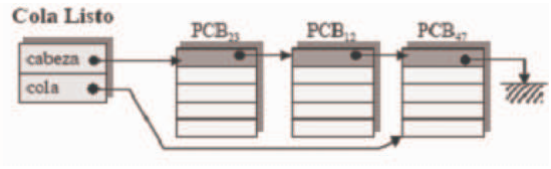
- **Ready:** El hilo listo para ejecutarse
- **Running:** El hilo se está ejecutando
- **Standby:** El hilo ha sido seleccionado para ser ejecutado en un procesador particular
- **Waiting:** Se encuentra bloqueado esperando un evento
- **Transition:** El hilo se encuentra preparado para ejecutarse, pero los recursos necesarios no están disponibles
- **Terminated**: Finalización del hilo

32 Planificación de procesos

- Objetivos de la Planificación de Procesos:
  - **Multiprogramación**: Tener siempre un proceso en ejecución con el propósito mejorar utilización CPU y otros recursos
  - **Tiempo Compartido:** Cambiar rápidamente la CPU entre procesos para mantener buena interactividad
- No pueden existir más procesos en ejecución simultánea que el número de procesadores
- Sistemas con más de un procesador permiten tener más de un proceso en ejecución simultánea

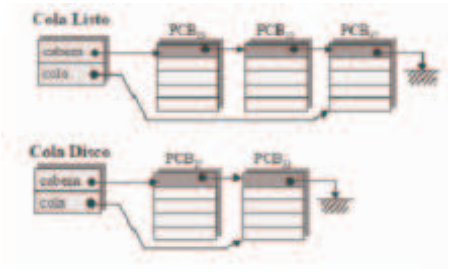
33 Colas de planificación (I)

- **Cola de trabajos**: a medida que los procesos entran en el sistema se le añade a esta cola
- Aquellos procesos en memoria y esperando ejecutarse se mantienen en una cola llamada **Cola de procesos listos**
  - Implementada normalmente como una *Lista Enlazada*: El encabezado contiene punteros al primero y último PCB. Cada PCB contiene un puntero al siguiente PCB de la lista de procesos listos

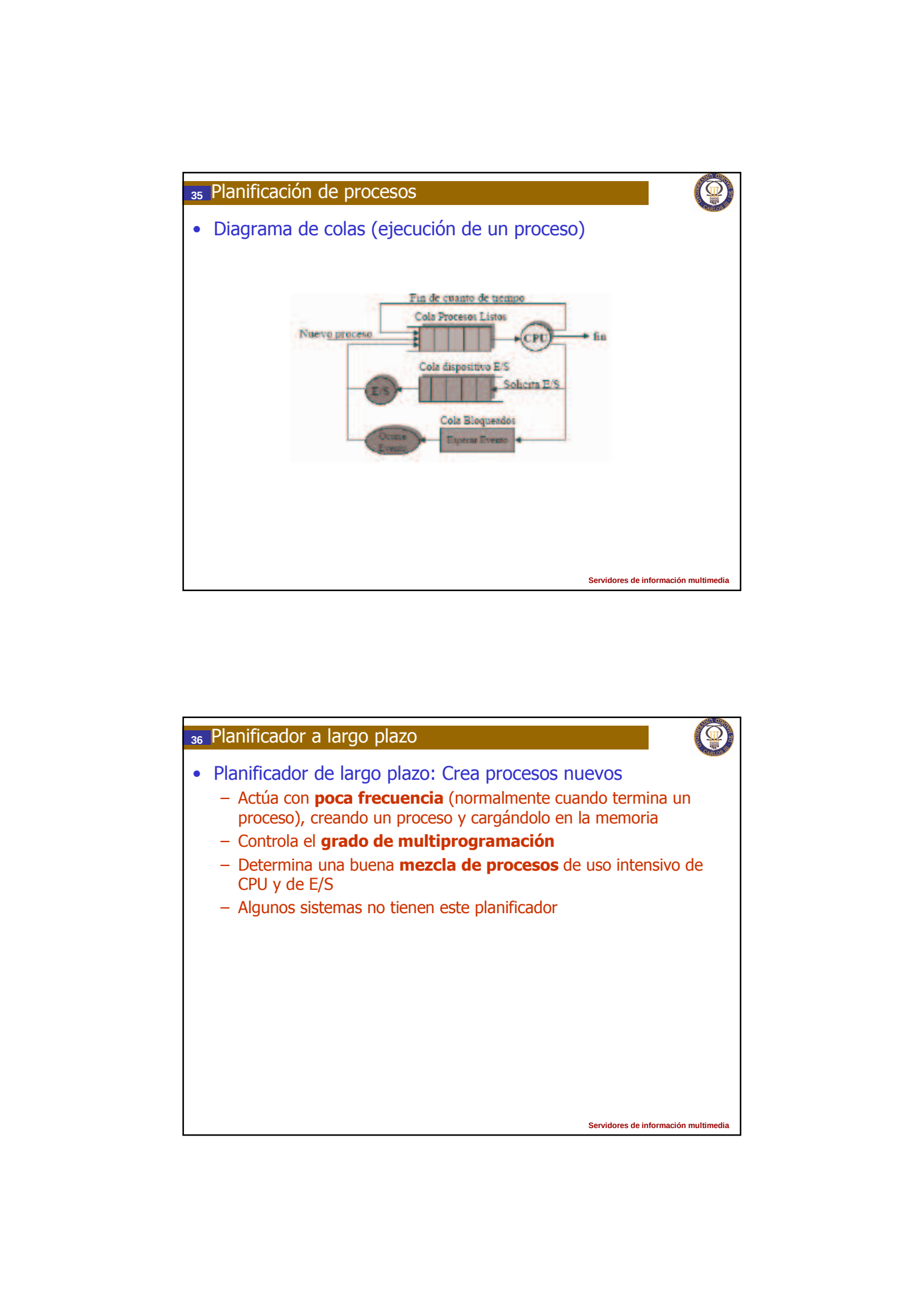


34 Colas de planificación (II)

- Existen otras colas en el sistema
- Para la gestión de dispositivos de E/S se implementa una cola para evitar que varios procesos accedan a la vez a un dispositivo. Se añaden a la cola y el proceso debe esperar a que el dispositivo se le asigne.
- Estas colas se conocen como **Colas de Dispositivo**
  - hay una cola por cada dispositivo



35 Planificación de procesos



- Diagrama de colas (ejecución de un proceso)

36 Planificador a largo plazo

- Planificador de largo plazo: Crea procesos nuevos
  - Actúa con **poca frecuencia** (normalmente cuando termina un proceso), creando un proceso y cargándolo en la memoria
  - Controla el **grado de multiprogramación**
  - Determina una buena **mezcla de procesos** de uso intensivo de CPU y de E/S
  - Algunos sistemas no tienen este planificador

37 Planificador a corto plazo

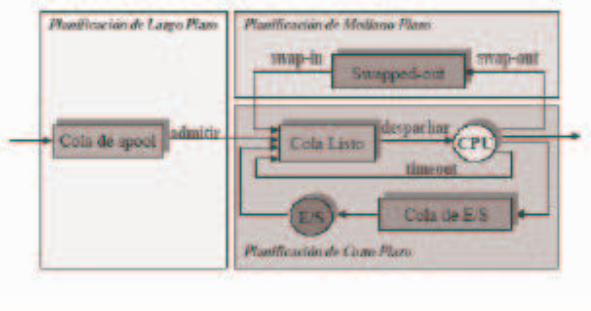
- Planificador de corto plazo: Asegura la Interactividad (Planificador de la CPU)
  - Selecciona procesos de la cola de procesos listos para asignarles la CPU
  - Se ejecuta con alta frecuencia, cada vez que ocurre un suceso:
    - Expiración de ranuras (timer) de CPU
    - Interrupciones de E/S
    - Llamadas al sistema operativo
    - Por terminación de un proceso (exit)
  - Existen diferentes algoritmos de asignación de CPU

38 Planificador a medio plazo

- Planificador a medio plazo: Mueve procesos entre M.Principal y M.Secundaria
  - Permite regular la carga reduciendo o aumentando el grado de multiprogramación, usando técnica de **swapping** (intercambio)
  - Un factor de decisión importante es la demanda por memoria de los procesos
  - Se usa en sistemas de tiempo compartido

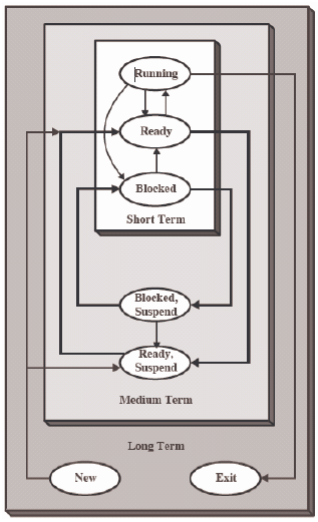
39 Planificacidores

- Todo junto:



40 Niveles de planificación

- Corto Plazo: Blanco
- Medio Plazo: Gris claro
- Largo Plazo: Gris oscuro



41 Planificación de la CPU

- Planificación de la CPU:
  - Realiza la ***Planificación a corto Plazo***
  - Selecciona el próximo proceso/hilo *listo* que debe recibir el procesador
  - Existen diferentes algoritmos de planificación
  - Es invocado cada vez que un proceso/hilo abandona el procesador (voluntariamente o no)

42 Ráfagas de CPU

- Ráfagas de CPU
  - Podemos considerar que la vida activa de un proceso es una sucesión de:
    - ráfagas de CPU -> el proceso ejecuta instrucciones
    - ráfagas de E/S -> el proceso utiliza o espera por la E/S
  - Según la utilización de los recursos, se observan:
    - Procesos intensivos en CPU: se pasan la mayor parte de su existencia en CPU
    - Procesos intensivos en E/S: se pasan la mayor parte de su existencia realizando/esperando operaciones de E/S

43 Objetivos de la planificación

- Los más importantes son:
  - Justicia en el reparto de procesador
  - Maximizar la **productividad**
  - Ser predecible (minimizar la varianza)
  - Balancear el uso de los recursos
  - Lograr un equilibrio entre **tiempo de respuesta** y **utilización**
  - Evitar inanición (o postergación indefinida)
  - Asegurar prioridades de los procesos
  - Favorecer a procesos/hilos que utilizan recursos claves
  - Degradar suavemente el rendimiento ante cargas pesadas

44 Políticas de planificación

- El planificador de la CPU puede intervenir si:
  - 1. Un proceso/hilo pasa voluntariamente a *Bloqueado* (p.e. E/S, sincronización, etc...)
  - 2. Pasa a estado *Listo* por expiración de tiempo (timeout)
  - 3. Cuando un proceso/hilo pasa de estado *Bloqueado* a *Listo*
  - 4. Cuando un proceso/hilo termina
- Es decir, siempre que un proceso/hilo abandona la CPU, o se inserta un proceso/hilo en la cola de *Listos*

45 Políticas de planificación

- Planificación **expropiativa**:
  - El SO puede quitar (expropiar) la CPU al proceso/hilo
- Planificación **no expropiativa**:
  - No se puede retirar el proceso/hilo de la CPU, este la libera voluntariamente al bloquearse o finalizar

- La expropiación *(preemption)* nos asegura que un trabajo no bloquea a otro igualmente importante
- La planificación no expropiativa *(non preemption)* requiere que los trabajos devuelvan el control al SO (Al planificador)

46 Políticas de planificación

- **Despachador (Dispatcher)**
  - Módulo que cede el control de la CPU al proceso/hilo seleccionado por el planificador a corto plazo
  - Funciones:
    - Cambia de contexto
    - Cambia a modo usuario
    - Saltar al punto apropiado del programa de usuario
  - Debe ser muy rápido para reducir el tiempo de latencia (overhead)

47 Criterios de planificación

- **Utilización**: mantener la CPU tan ocupada posible
- **Productividad**: nº de procesos/hilos que completan su ejecución por unidad de tiempo
- **Tiempo de retorno**: cantidad de tiempo necesaria para ejecutar (completar) un proceso/hilo dado
- **Tiempo de espera**: tiempo que un proceso/hilo ha estado esperando en la cola de *Listos*
- **Tiempo de respuesta**: tiempo que va desde que se remite una solicitud hasta que se produce la primera respuesta (no salida)

48 Algoritmos de planificación

- **Tipos de Algoritmos de planificación**
  - **Expropiación** (preemption). Existen algoritmos con y sin expropiación de la CPU
  - **Intervalos de tiempo**. El proceso/hilo puede recibir la CPU por un cierto lapso de tiempo
  - **Prioridades**. A los procesos/hilos se les pueden asociar prioridades, las cuales pueden ser estáticas o dinámicas
  - **Tiempos límites** (deadlines). Existen tiempo límites para que termine un proceso/hilo. Cuanto más cerca se esté de ese límite, más urgente se hace su planificación

49 Algoritmos de planificación

- **Algoritmos de planificación**
  - Sin Expropiación
    - **FCFS** (First Come, First Serve)
    - **SJF** (Shortest Job First)
    - **Prioridades** (estáticas y dinámicas)
  - Con Expropiación
    - **Round-Robin FCFS** (First Come, First Serve)
    - **SRTF** (Shortest Remaining Time First)
    - **Prioridades** (estáticas y dinámicas)

50 FCFS (Primero en llegar, Primero en ser servido)

- FCFS (Primero en llegar, Primero en ser servido)
  - Es el algoritmo más simple de implementar
  - La cola de *Listos* se gestiona como una cola FIFO (First Input First Output)
  - Muy sensible al orden de llegada de los procesos/hilos
  - Puede producir grandes tiempos de espera

51 Algoritmo FCFS. Ejemplo

Proceso	Duración
P1	9
P2	4
P3	2

- Calcular el tiempo de espera, tiempo de retorno y tiempo medio de espera si aplicamos el algoritmo FCFS suponiendo que llegan en el mismo instante en el siguiente orden: P1, P2, P3
  - Tiempos de Espera: P1=0; P2=9; P3=13
  - Tiempos de Retorno: P1=9; P2=13; P3=15
  - T. espera Medio: (0+9+13)/3= 7.3
  - Productividad: 3/15=0.2
- Realizar los mismos cálculos suponiendo que llegan en el siguiente orden: P2, P3 y P1
  - T. espera Medio: (0+4+6)/3= 3.3

52 Algoritmo FCFS. *Efecto Convoy*

- Se produce cuando se tiene una mezcla de un proceso/hilo ligado a CPU y varios procesos/hilos ligados a E/S
- Estos últimos tenderán a ubicarse en la cola de *Listos*, donde serán frenados por el proceso/hilo que usa mucha CPU, quedando los recursos de E/S libres
- Por el otro lado, cuando el proceso/hilo ligado a CPU libera el procesador, los demás pasan rápido a E/S, quedando la CPU ociosa
- Por lo tanto, se produce baja utilización de CPU y de los dispositivos de E/S
- Una solución requeriría una planificación no FCFS.

53 SJF (primero el más corto)

- Entra en CPU el proceso con la ráfaga de CPU más breve
- Minimiza el tiempo de espera medio
- Riesgo de **inanición** de los procesos/hilos de larga duración
- Se pueden estimar las próximas ráfagas de CPU de los procesos/hilos según su historia reciente
- Versión expulsiva (SRTF): el proceso en CPU es desalojado si llega a la cola un proceso/hilo con duración más corta (SRTF : Primero se trata el proceso de tiempo restante menor)

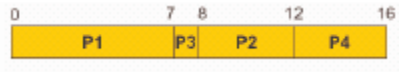
54 SJF (primero el más corto)

- ¿Cómo se estima la duración de la siguiente ráfaga de CPU?
  - La siguiente ráfaga de CPU se predice como una media exponencial de las longitudes medias en anteriores ráfagas
  - Sea:
    - t_n: longitud de la n-ésima ráfaga de CPU
    - τ_n: valor predicho para la n-ésima ráfaga de CPU
    - α : parámetro de ajuste ($0 <= \alpha <= 1$)
      - $\tau_{n+1} = \alpha\, t_n + (1-\alpha)\, \tau_n$
    - Si $\alpha=0$ → La historia reciente no tiene efecto, se supone que las condiciones actuales son transitorias. $\tau_{n+1} = \tau_n$
    - Si $\alpha=1$ → No se tiene en cuenta la historia, sólo tiene importancia la ráfaga más reciente de CPU. $\tau_{n+1} = t_n$
    - Es habitual que $\alpha=1/2$ por lo que la historia reciente y antigua se pondera de igual manera

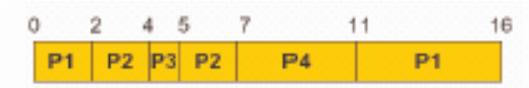
55 Algoritmo SJF. Ejemplo

Proceso	Llegada	Duración
P1	0	7
P2	2	4
P3	4	1
P4	5	4

- Calcular el tiempo medio de espera que resulta de aplicar un algoritmo SJF no expulsivo
  - espera media: (0+6+3+7)/4=4



- Calcular el tiempo medio de espera que resulta de aplicar un algoritmo SJF expulsivo (SRTF)
  - espera media: (9+1+0+2)/4=3



56 SJF (primero el más corto)

- **Ventajas:**
  - SJF es óptimo en cuánto a reducir el tiempo de espera
- **Problemas:**
  - Es necesario conocer a priori o predecir el tiempo de servicio del trabajo
  - Existe posibilidad de inanición para trabajos largos

57 SRTF

- Corresponde a SJF con expropiación
- Se planifica como siguiente el que tenga menor tiempo de proceso hasta su término
- Si llega un trabajo con menor tiempo de proceso que el actual en ejecución, lo saca del procesador (expropiación)
- Se debe recordar el tiempo remanente de proceso para cada trabajo desplazado

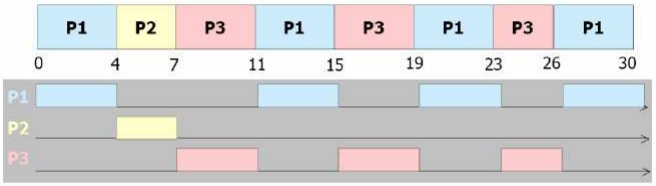
58 Round Robin (Turno rotatorio - circular)

- Es FCFS con expropiación
- Adecuado para implementar tiempo compartido
- Cada proceso/hilo dispone de un *cuanto* de tiempo máximo *q*
- Si cuando expira el *cuanto* de tiempo el proceso continúa en CPU, el planificador lo desaloja y lo ingresa al final de la cola de *Listos*
- La cola de *Listos* se gestiona como una cola FIFO
- Se usa en sistemas interactivos
- Evita o reduce el efecto convoy
- Si hay *n* procesos, cada uno obtiene *1/n* del tiempo de la CPU en intervalos de *q* unidades, como máximo.

59 Round Robin - Ejemplo

Cuanto: q=4

Procesos	T. Llegada	Duración
P1	0	16
P2	0	3
P3	0	11



Tiempos de Retorno: P1=30; P2=7; P3=26

Tiempo de espera: P1 = 7 +4 +3 = 14 ; P2 = 4; P3 = 7 + 4 + 4 = 15;

T. espera Medio: (14 + 4 + 15)/3= 11

T. espera máximo : 7 (P3 y P1)

60 Influencia del valor del quanto

- Round Robin. Influencia del *cuanto* de tiempo (*q*)
- En general, si *n* es el número de procesos, entonces un proceso debe esperar a lo más:
  - (*n*-1)* *q* para el próximo *cuanto* de tiempo
- Regla para un rendimiento óptimo:
  - E*l 80% de las ráfagas de CPU deben ser menores que el cuanto*

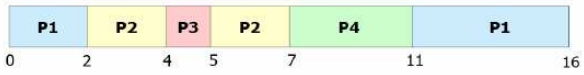
61 Planificación por Prioridades

- Cada proceso tiene una prioridad; entra en CPU aquel con mayor prioridad
  - la política puede ser expulsiva o no
  - Prioridades definidas de forma interna (por el S.O.) o externa (por los usuarios)
  - El SJF/SRTF son casos de planificación por prioridades donde la prioridad se establece en base a la duración estimada de la tarea)
- Riesgo de **inanición** de los procesos con menos prioridad

62 Planificación por Prioridades - Ejemplo

Procesos	T. Llegada	Duración	Prioridad
P1	0	7	5
P2	2	4	10
P3	4	1	15
P4	5	4	10

Ejemplo con Expulsión



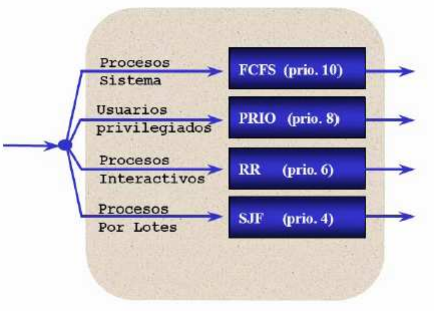
Tiempos de Retorno:
P1 = 16; P2 = 5; P3 = 1;P4 = 6
T. espera Medio:
(9+1+0+2)/4 = 3

63 Planificación con Multicolas

- Los procesos se pueden clasificar fácilmente en distintos grupos interactivos, por lotes, etc.)
- Varias colas de *Listos*, cada una gestionada con una política diferente
- Las colas se reparten la CPU según alguna política:
  - por prioridad absoluta
  - un % de tiempo para cada cola
- Ejemplo:
  - ***Trabajos interactivos***: Round-Robin y máxima prioridad
  - ***Trabajos de fondo*** (lotes): FCFS y mínima prioridad
- El tiempo de CPU se puede dividir en los grupos (p.e. 80% procesos interactivos y 20% proceso por lotes)
- Debe existir un algoritmo de planificación entre colas

64 Multicolas - Ejemplo

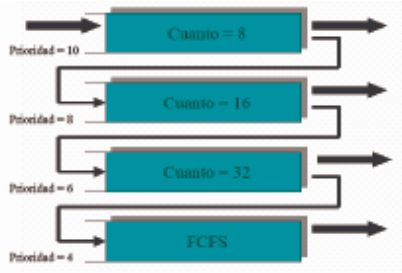
- **Prioridades estrícta**s: Los procesos en cola con menor prioridad no se ejecutan mientras haya procesos en cola con mayor prioridad. Posibilidad de inanición
- **Cuotas de tiempo**: Cada cola está asignada a un porcentaje del tiempo de CPU. Ejemplo: el 80% a trabajos interactivos y el 20% a trabajos por lotes.
- **Prioridades dinámicas**: en función del tamaño de la cola



65 Multicolas con Realimentación

- Existen diferentes colas de procesos preparados.
- Cada cola posee:
  - Política de planificación: para los procesos que están en esa cola
  - Una prioridad asignada: frente a otras colas
- Un proceso puede cambiar de cola de acuerdo con un esquema de actualización de prioridades
  - Los procesos con un tiempo de espera acumulado elevado son **promocionados** a una cola con prioridad superior
  - Los procesos con un tiempo de utilización de la CPU elevado son **degradados** a una cola con prioridad inferior.

66 Multicolas con Realimentación



67 Multicolas con Realimentación

- Multicolas con Realimentación
  - Procesos ligados a E/S tienen alta prioridad (pasan rápido por la CPU)
  - Procesos ligados a CPU disminuyen su prioridad (bajan a colas de mayor cuanto)
  - Para evitar inanición, procesos que esperan mucho pueden subir una cola (envejecimiento)

68 Planificación de la CPU en Linux

- Linux utiliza dos algoritmos de planificación
  - Algoritmo de tiempo compartido con expropiación
  - Algoritmo para tareas en tiempo real por prioridades
- Tiempo Compartido
  - Linux usa un algoritmo de prioridades basado en créditos
  - Cada proceso posee un cierto número de créditos
  - El planificador selecciona el proceso con más créditos de ejecución
  - Cada interrupción el proceso que está en ejecución pierde un crédito
  - Si creditos = 0, entonces el proceso es suspendido
  - Este sistema otorga prioridad a procesos interactivos o limitados por E/S para los que es importante un tiempo de respuesta rápido

69 Planificación de la CPU en Linux

- Tareas de tiempo real
  - Linux implementa dos clases de planificación FIFO y Round Robin
  - Los procesos además tienen una prioridad
  - Siempre corre el proceso con mayor prioridad

Autoría del material

- (c) Mario Muñoz Organero