Sistemas operativos
2ª edición

Capítulo 4
Planificación del procesador
(extracto de las transparencias del libro)

Contenido

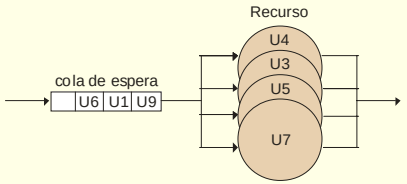
- Introducción
- Caracterización de los procesos
- Objetivos de la planificación
- Algoritmos de planificación expulsivos y no expulsivos
- Planificación en multiprocesadores
- Estudio de un ejemplo: la planificación en Linux
- Planificación en tiempo real

Introducción

- SO planifica recursos
  - UCP recurso más importante
- Planificación del procesador
  - Aparece con multiprogramación
  - Sistemas por lotes: aprovechamiento de UCP
  - Tiempo compartido: reparto equitativo entre usuarios
  - PC: interactividad; buen tiempo de respuesta
  - Multiprocesadores: aprovechar paralelismo
  - Tiempo real (no crítico): cumplir plazos

El problema general de la planificación

- Recurso con múltiples ejemplares utilizado por varios usuarios
- Planificación: qué ejemplar se asigna a qué usuario



Aspectos generales de la planificación

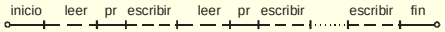
- Objetivos generales:
  - Optimizar uso
  - Minimizar tiempo de espera
  - Ofrecer reparto equitativo
  - Proporcionar grados de urgencia
- Tipos de planificación: expulsiva versus no expulsiva
- Afinidad a subconjunto de ejemplares de recurso
  - Estricta: pedida por el usuario
  - Natural: favorece rendimiento

Caracterización de los procesos

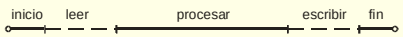
- Perfil de uso del procesador
  - Mejor primero proceso con ráfaga de UCP más corta
- Grado de interactividad
  - Asegurar buen tiempo de respuesta de interactivos
- Nivel de urgencia
  - Especialmente importante en sistemas de tiempo real

Uso intensivo de la E/S versus la UCP

```
Programa

inicio();

Repetir
    leer(fichero_entr, dato);
    /* procesado sencillo */
    res=pr(dato);
    escribir(fichero_sal, res);
hasta EOF(fichero_entr);

fin();
```




```
Programa

inicio();

leer(fichero, matriz);

/* procesado complejo */
procesar(matriz);

escribir(fichero, matriz);

fin();
```



Objetivos de la planificación del procesador

- Optimizar el comportamiento del sistema.
- Diferentes parámetros (a veces contrapuestos)

- Parámetros por entidad (proceso o hilo):
  - **Tiempo de ejecución** (Te)
  - **Tiempo de espera**
  - **Tiempo de respuesta** (Ta)
- Parámetros globales:
  - **Uso del procesador** (C)
  - **Tasa de trabajos completados** (P)

Puntos de activación

- Puntos del SO donde puede invocarse el planificador :
  1. Proceso en ejecución finaliza
  2. Proceso realiza llamada que lo bloquea
  3. Proceso realiza llamada que desbloquea proceso más urgente
  4. Interrupción desbloquea proceso más urgente
  5. Proceso realiza llamada declarándose menos urgente
  6. Interrupción de reloj marca fin de rodaja de ejecución
- Dos tipos de algoritmos:
  - no expulsivos: sólo C.C. voluntarios (1 y 2)
  - expulsivos: además C.C. involuntarios (3, 4, 5 y/o 6)
- No confundir con núcleo expulsivo o no expulsivo

Algoritmos de planificación

- Primero en llegar primero en ejecutar (FCFS)
- Primero el trabajo más corto (SJF/SRTF)
- Planificación basada en prioridades
- *Round robin* (turno rotatorio)
- Colas multinivel

- Algoritmo real es mezcla de teóricos + ajustes empíricos

Primero en llegar primero en ejecutar (FCFS)

- Selección: Proceso que lleva más tiempo en cola de listos
- Algoritmo no expulsivo
- Fácil de implementar:
  - Cola de listos gestionada en modo FIFO
- Satisfacción de objetivos generales:
  - ✓ Optimizar uso
  - ↓ Minimizar tiempo de espera
  - ↓ Ofrecer reparto equitativo
  - ↓ Proporcionar grados de urgencia

Primero el trabajo más corto (SJF)

- Selección: Proceso listo con próxima ráfaga UCP más corta
- Versión no expulsiva: Sólo si proceso se bloquea o termina
- Versión expulsiva: 1º el de menor tiempo restante (SRTF)
  - También se activa si proceso pasa a listo (puntos 3 y 4)
- ¿Cómo se conoce a priori la duración de la próxima ráfaga?
  - Estimaciones a partir de las anteriores
- Puede producir inanición
- Punto fuerte:
  - ✓ Minimizar tiempo de espera

Planificación por prioridad

- Cada proceso tiene asignada una prioridad
- Selección: Proceso en cola de listos que tenga mayor prioridad
- Existe versión no expulsiva y expulsiva
  - Si proceso pasa a listo o actual baja su prioridad (3, 4 y 5)
- Las prioridades pueden ser estáticas o dinámicas
- Prioridad puede venir dada por factores externos o internos
- Puede producir inanición:
  - "Envejecimiento": Prioridad aumenta con el tiempo
- Punto fuerte:
  - ✓Proporcionar grados de urgencia

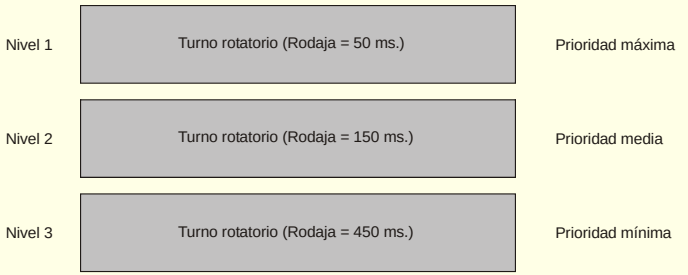
Turno rotatorio (*Round Robin*, RR)

- FCFS + plazo (rodaja o cuanto)
- Algoritmo expulsivo pero sólo con fin del cuanto (punto 6)
- Tiempo de respuesta acotado
- Tamaño rodaja
  - Grande: tiende a FIFO
  - Pequeño: demasiada sobrecarga
  - Igual para todos los procesos o no
  - Para un proceso: fija o dinámica
- Punto fuerte:
  - ✓Ofrecer reparto equitativo

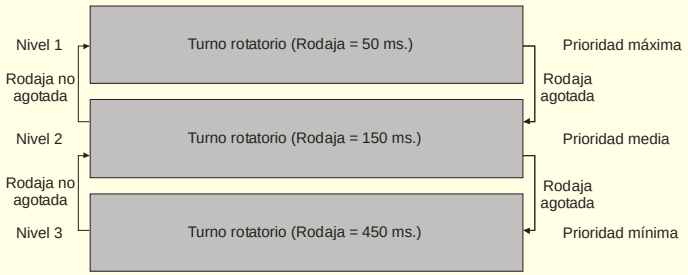
Colas multinivel

- Generalización: Distinguir entre clases de procesos
- Parámetros del modelo:
  - Número de niveles (clases)
  - Algoritmo de planificación de cada nivel
  - Algoritmo de reparto del procesador entre niveles
- Colas con o sin realimentación:
  - Sin: proceso en la misma cola durante toda su vida
  - Con: proceso puede cambiar de nivel

Colas multinivel sin realimentación



Colas multinivel con realimentación



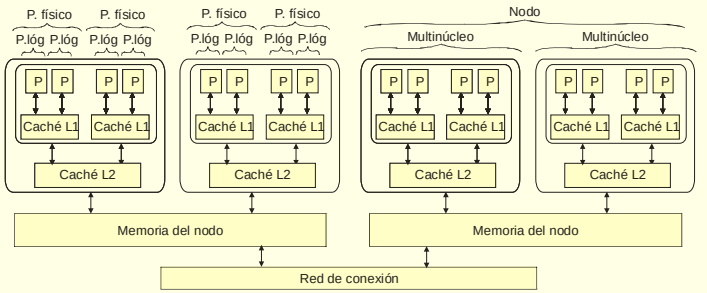
Planificación en multiprocesadores

- Trivial: *N* UCP ejecutan *N* procesos más prioritarios
- Sí, pero hay que tener en cuenta:
  - Afinidad natural y estricta
  - Multiprocesadores jerárquicos (SMT, CMP, NUMA, ...)
    - Compartimiento de recursos entre algunos procesadores
  - Evitar congestión en operación del planificador
  - Además de rendimiento puede haber otros parámetros
    - P.ej. minimizar consumo
    - Linux: `echo 1 > /sys/devices/system/cpu/sched_mc_power_savings`
- Exposición se centra en planificación de procesos independientes
  - No se trata, por ejemplo, *gang scheduling*
- 2 esquemas: Cola única vs. Una cola/procesador

Planificación en MP con cola única

- UCP elige qué proceso de la cola ejecuta (autoplanificación)
- Afinidad natural: mejor ejecutar en misma UCP
  - Aprovecha información en caché
- Planificación:
  - UCP queda libre: proceso más prioritario
    - Prioridad matizada por la afinidad natural
  - Proceso *P* pasa a listo:
    1. UCP afín libre
    2. UCP libre
    3. UCP con proceso *Q* tal que prio(*P*)>prio(*Q*)
       - Prioridad matizada por la afinidad natural
  - Uso de **int. SW de planificación** + IPI para forzar CCI
- Además, debe respetar afinidad estricta

Sistema multiprocesador jerárquico



- Jerarquía genérica
  - SMT (*hyperthreading*)
  - CMP (*multicore*)
  - NUMA

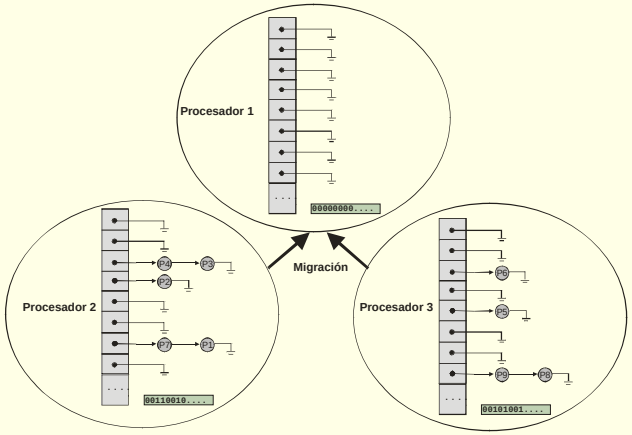
Planificación en multiprocesador jerárquico

- Compartimiento de recursos entre algunos procesadores afecta a:
  - Afinidad: Extensión de afinidad a la jerarquía
  - Prestaciones: 2 UCP comparten → Potencia total < 2*pot./UCP
- Jerarquía de afinidades
  - SMT: Afinidad a núcleo
    - Mejor ejecutar en UCP lógica ∈ mismo núcleo
  - CMP: Afinidad a multinúcleo
    - Mejor ejecutar en núcleo ∈ mismo multinúcleo
  - NUMA: Afinidad a nodo
    - Mejor ejecutar en mismo nodo
- Reparto teniendo en cuenta grado de independencia
  - Mejor ir ocupando UCPs con mayor grado de independencia
  - Crea proc: UCP lógica libre en núcleo libre de multinúcleo libre

Planificación en MP con una cola por UCP

- Cola única:
  - Accesos a cola requieren cerrojo
  - Limitado aprovechamiento de la afinidad natural
- Cola por UCP: UCP se planifica de forma independiente
  - No hay congestión por cerrojo y se aprovecha mejor afinidad
- ¿En qué UCP inicia ejecución nuevo proceso?
  - Procesador menos cargado
  - Aplicando jerarquía: Procesador seleccionado corresponde a
    - Nodo menos cargado (N)
    - Multinúcleo (M) menos cargado de N
    - Procesador físico (F) menos cargado de M
    - Procesador lógico (L) menos cargado de F

Planificación con una cola por procesador



Planificación en Linux

- Muy versátil: desde portátiles a grandes servidores
- Mejoras del planificador en versión 2.6:
  - Menor tiempo de respuesta. Núcleo expulsivo
  - Sobrecarga independiente de número de procesos O(1)
  - Mejor soporte multiprocesador: uso de una cola por UCP
- Planificador: módulo relativamente cambiante

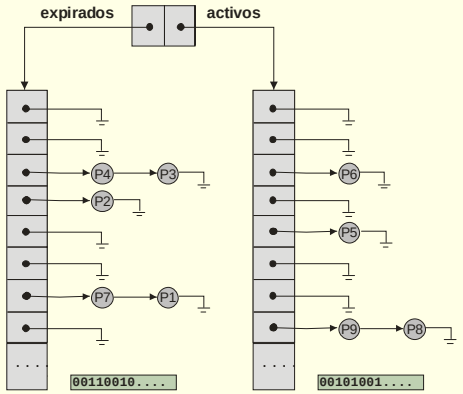
Planificación en Linux: uniprocesador (1/2)

- Cola multinivel sin realimentación: t. real (no crítico) y normal
- Clase tiempo real (+ prioritaria): soporte extensiones t. real POSIX
  - Prioridad estática (SCHED_FIFO o SCHED_RR para iguales)
    - Prioridad y rodaja especificada por el proceso
- Clase normal (SCHED_OTHER): prio dinámica + RR (rodaja fija)
  - Prioridad base estática: -20 (máx.) a 19 (mín.)
  - Ajuste dinámico de -5 (mejora) a +5 (empeora)
    - Depende de uso de UCP y tiempo de bloqueo
    - Favorece procesos interactivos y con E/S y elimina inanición
  - Rodaja con tamaño fijo que depende de la prioridad base
    - 5ms (prio 19); 100ms (prio 0); 800 ms (prio -20)

Planificación en Linux: uniprocesador (2/2)

- Aspectos específicos poco convencionales:
  - 2 listas de procesos listos: activos y expirados
  - Planificador selecciona de lista de activos
  - Proceso que agota su rodaja pasa a lista de expirados
    - Excepto si se considera "interactivo" que vuelve a activos
  - Cuando se desbloquea proceso pasa a lista de activos
  - Se crean "rondas de ejecución"
    - Procesos agotan rodaja y terminan todos en lista de expirados
    - En una ronda proceso va gastando su rodaja por tramos
    - "Fin de ronda": intercambio de listas

Planificación en Linux: colas de procesos



Planificación en sistemas de tiempo real

- Cada proceso (**Pi**) se caracteriza por:
  - Periodo de activación: **Ti**
  - Tiempo de cómputo (máximo)/activación: **Ci**
  - Plazo límite (*deadline*): **Di**
  - Grado de uso de UCP: **Ui** = **Ci** / **Ti**
- Modelo simplificado:
  - Procesos periódicos e independientes
  - **T** = **D**
- 2 tipos de planificación:
  - Con prioridades estáticas
    - Monótona en frecuencia (RMS: *Rate-Monotonic Scheduling*)
  - Con prioridades dinámica
    - Por tiempo límite (EDF: *Earliest Deadline First*)

Planificación RMS

- Asignación estática de prioridades
  - Mayor prioridad ← menor plazo
  - Soporte directo por SO con esquema de prioridades estáticas
- Factibilidad de una planificación dada con *n* procesos:
  - No siempre aunque $\sum Ui \leq 1$
  - Garantizada si $\sum Ui \leq [n \cdot (2^{1/n} - 1)]$
    - $n = 2 \rightarrow 0{,}828$; $n = 3 \rightarrow 0{,}780$; $n = 4 \rightarrow 0{,}757$; ...
    - $n \rightarrow \infty \Rightarrow [n \cdot (2^{1/n} - 1)] \rightarrow \ln 2 \approx 0{,}693$
  - Aunque $\sum Ui > [n \cdot (2^{1/n} - 1)]$ no implica infactibilidad
    - Depende de cada caso

Planificación EDF

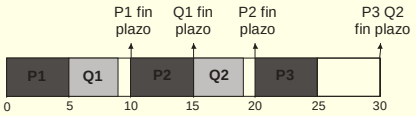
- Asignación dinámica de prioridades
  - Mayor prioridad ← más cercano fin de plazo
  - SO debería conocer plazos de procesos
- Factibilidad de una planificación dada con *n* procesos:
  - Basta con que $\sum Ui \leq 1$

Ejercicio sobre planificación en tiempo real

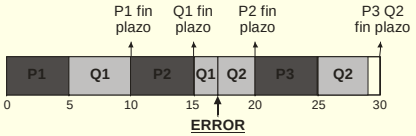
- Aplicar el algoritmo RMS a los 3 siguientes ejemplos:
  - Ejemplo 1:
    - Proceso P: C = 5; T = 10
    - Proceso Q: C = 4; T = 15
  - Ejemplo 2:
    - Proceso P: C = 5; T = 10
    - Proceso Q: **C = 7**; T = 15
  - Ejemplo 3:
    - Proceso P: C = 5; T = 10
    - Proceso Q: **C = 5**; T = 15
- Uso de EDF para ejemplos de RMS no factibles

Ejemplos de aplicación de RMS

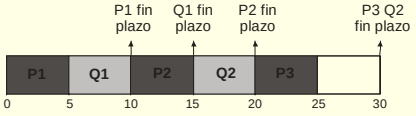
P: C = 5; T = 10; U = 0,5
Q: C = 4; T = 15; U= 0,267
U total = 0,767 < 0,828
T(P) < T(Q) —> PRIO(P) > PRIO(Q)
FACTIBLE



P: C = 5; T = 10; U = 0,5
Q: C = 7; T = 15; U= 0,467
U total = 0,967 > 0,828
T(P) < T(Q) —> PRIO(P) > PRIO(Q)
NO FACTIBLE



P: C = 5; T = 10; U = 0,5
Q: C = 5; T = 15; U= 0,333
U total = 0,833 > 0,828
T(P) < T(Q) —> PRIO(P) > PRIO(Q)
PERO FACTIBLE



Ejemplo de aplicación de EDF

P: C = 5; T = 10; U = 0,5
Q: C = 7; T = 15; U= 0,467
U total = 0,967 < 1
FACTIBLE