Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas

Sistemas Operativos

Unidad I:

Estructura de los sistemas operativos

Planificacion de procesos

M. en C. Hermes Francisco Montes Casiano
hermes.upiicsa@gmail.com

Índice

Introducción

Planificador

- Cuando hay más de un proceso ejecutable, el sistema operativo debe decidir cuál ejecutará primero.
- La parte del Sistema Operativo que toma esta decisión se denomina **planificador**.
- El algoritmo que usa se denomina **algoritmo de planificación**.

Introducción

- En la época de los sistemas de procesamiento por lote el algoritmo de planificación era sencillo, simplemente **se ejecutaba el siguiente trabajo de la cinta**.
- En los sistemas de tiempo compartido, el algoritmo de planificación es más complejo, pues es común que haya varios usuarios, con múltiples procesos cada uno, en espera de ser atendidos.

Ejemplo

En las computadoras personales, puede haber varios procesos iniciados por un usuario compitiendo por la CPU, sin mencionar los trabajos en segundo plano.

Introducción

Política de planificación

El planificador se ocupa de decidir una política de planificación y debe cumplir con los siguientes criterios:

- Equitatividad
- Eficiciencia
- Tiempo de respuesta
- Retorno
- Volumen de producción

Introducción

- Se demostró [Kleinrock, 1975] que cualquier algoritmo de planificación que dé preferencia a una clase de trabajos perjudicará a los trabajos de otras clases.

Tiempo de cómputo

La cantidad de tiempo de CPU es finita. Para darle más a un usuario tenemos que darle menos a otro.

¿Qué se puede predecir de la ejecución de un proceso?

Una complicación que deben enfrentar los planificadores es que cada proceso es **único** e **impredecible**.

Introducción

- Algunos procesos dedican una buena parte del tiempo a esperar E/S de archivos, mientras que otros usarían la CPU si se les permitiera hacerlo.
- Cuando el planificador comienza a ejecutar un proceso, nunca sabe con certeza cuánto tiempo pasará antes de que dicho proceso se bloquee (E/S, espera de un semáforo, etc).
- Para asegurar que ningún proceso se ejecute durante demasiado tiempo, casi todas las computadoras tienen incorporado un cronómetro que genera interrupciones periódicamente.

Introducción

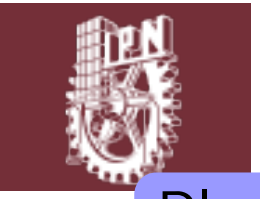
- El sistema operativo se ejecuta en cada interrupción de reloj.

Planificador

Decide si debe permitir que el proceso que se está ejecutando continue o suspenderlo si ya disfrutó de suficiente tiempo de CPU y otorgar a otro proceso la CPU.

Planificación expropiativa

La estrategia de permitir que procesos lógicamente ejecutables se suspendan temporalmente se denomina **planificación expropiativa**.



Introducción

Planificador

Decide si debe permitir que el proceso que se está ejecutando continue o suspenderlo si ya disfrutó de suficiente tiempo de CPU y otorgar a otro proceso la CPU.

Planificación expropiativa

La estrategia de permitir que procesos lógicamente ejecutables se suspendan temporalmente se denomina **planificación expropiativa**.

Consecuencias de la planificación expropiativa

Da pie a condiciones de competencia y requiere semáforos, monitores, mensajes o algún otro método para prevenirlas.

Índice

Planificación round robin

- Es uno de los algoritmos más antigüos, sencillos, equitativos y ampliamente utilizados.
- A cada proceso le asigna un intervalo de tiempo durante el cual se le permite ejecutarse, llamado **cuanto**.
- Si el proceso se está ejecutando al expirar su **cuanto** el sistema operativo se apropia de la CPU y se la da a otro proceso.
- Si el proceso se bloquea o termina antes de expirar su **cuanto**, la conmutación de CPU se efectua naturalmente.

Implementación

Es facíl de implementar, lo que tiene que hacer es mantener una lista de procesos ejecutables. Cuando un proceso gasta su **cuanto**, se coloca al final de la lista.

Planificación round robin

- La cuestión interesante es determinar la duración del **cuanto**.
- La conmutación de un proceso a otro requiere de cierto tiempo para realizar operaciones administrativas
  1. Guardar y cargar registros,
  2. Administradción de los mapas de memoria,
  3. Actualizar diversas tablas y listas, etc

Planificación round robin

Ejemplo

- La conmutación de un proceso requiere 5 *ms*
- El cuanto dura 20 *ms*

Con esos parámetros, despúes de realizar algún trabajo útil durante 20 *ms*, la CPU tendrá que utilizar 5 *ms* en la conmutación de procesos.

Problema

Se desperdiciará el 20 % del tiempo del CPU en gastros administrativos.

- La conmutación de un proceso requiere 5 *ms*
- El cuanto dura 20 *ms*

Planificación round robin

Características del cuanto

- Escoger un cuanto demasiado corto causa demasiadas conmutaciones de procesos y reduce la eficiencia de la CPU.
- Elegir uno demasiado largo puede dar pie a una respuesta deficiente a solicitudes interactivas cortas.

Índice

Planificación por prioridad

- La asignación de prioridades se puede realizar de forma dinámica o estática.
- A cada proceso se le asigna una prioridad y se ejecuta el proceso ejecutable que tenga la prioridad más alta.

Ejemplo

Un proceso demonio que envía correos electrónicos en segundo plano debe tener menor prioridad que uno que muestra video en tiempo real.

Ajuste de la prioridad

Para evitar que los procesos con más alta prioridad se ejecuten indefinidamente, el planificador puede reducir la prioridad de los procesos en cada interrupción de reloj.

Planificación por prioridad

Clases de prioridad

- En muchos casos es conveniente agrupar los procesos en clases de prioridad, usar planificación por prioridad entre las clases y planificación *round robin* dentro de cada clase.
- El algoritmo es el siguiente:
  1. En tanto haya procesos ejecutables en la clase de prioridad **A**, se ejecutará cada uno durante un cuanto, con round robin.
  2. Si la clase de prioridad **A** está vacía, se ejecutan los procesos de la clase B con round robin.
  3. Si tanto la clase **A** como la clase **B** están vacías, se ejecutan los procesos **C** con round robin.

Problema

Si las prioridades no se ajustan ocasionalmente, las clases de baja prioridad podrían morir de inanición.

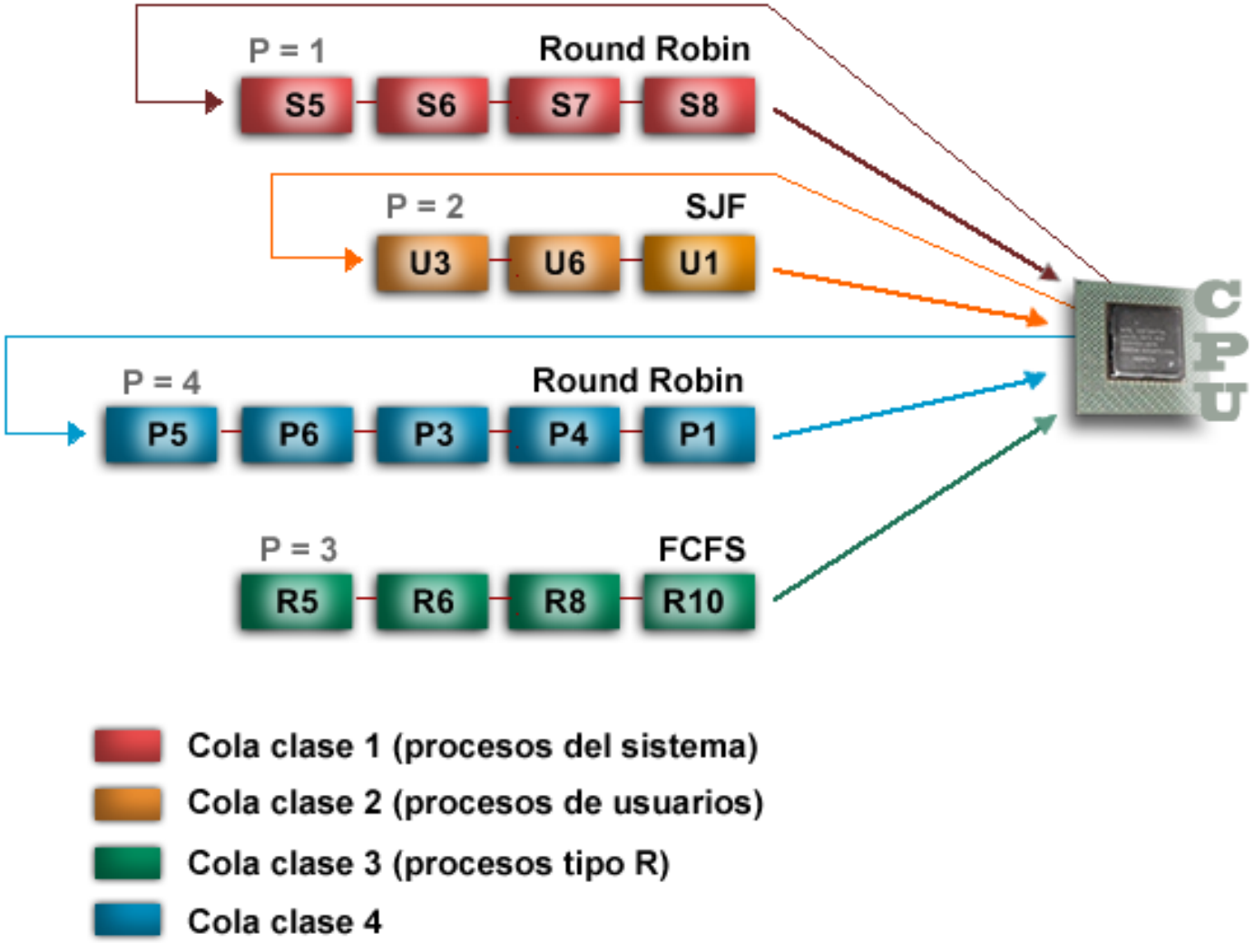
Índice

Colas múltiples

- Las colas múltiples son una solución a la prolemática que se presenta cuando coexisten procesos con diferentes necesidades.
- Esta política de planificación se basa en algún esquema predeterminado, que da un tratamiento especial a los trabajos de cada cola.
- Para la planificación en colas múltiples se requieren dos niveles de planificación:
  1. **Planificación dentro de cada cola:** cada cola puede utilizar su propia política de planificación, de acuerdo a la clase de procesos que tiene (round robin, FCFS, etc).
  2. **Planificación entre colas:**
     a) Se le asigna una prioridad P a cada cola.
     b) Se le asigna Quantum CPU a cada cola, que se parte entre los procesos de cada cola.

Colas múltiples

Ilustración



Índice

El primer trabajo más corto

- La mayor parte de los algoritmos mencionados fueron diseñados para sistemas interactivos.

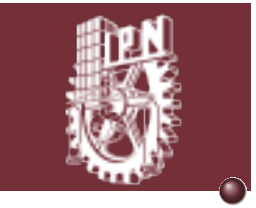
El primer trabajo más corto

Este tipo de algoritmos resultan especialmente apropiados para los trabajos por lotes cuyos tiempos de ejecución se conocen por adelantado.

Ejemplo

Se cuenta con cuatro trabajos A, B, C y D, con tiempos de ejecución de 8, 4, 4 y 4 minutos, respectivamente.

¿Cuál sería el tiempo de respuesta para cada proceso?



El primer trabajo más corto

- La mayor parte de los algoritmos mencionados fueron diseñados para sistemas interactivos.

El primer trabajo más corto

Este tipo de algoritmos resultan especialmente apropiados para los trabajos por lotes cuyos tiempos de ejecución se conocen por adelantado.

Ejemplo

Se cuenta con cuatro trabajos A, B, C y D, con tiempos de ejecución de 8, 4, 4 y 4 minutos, respectivamente.

Si se ejecutaran en ese orden el tiempo de retorno para A será de 8 minutos, para B de 12 minutos, para C de 16 minutos y para D de 20 minutos.

El primer trabajo más corto

- Se puede demostrar que la politica del primer trabajo más corto es óptima.
- Dado que produce un tiempo de respuesta medio, sería deseable poderlo usar también para procesos interactivos.

Procesos interactivos

Los procesos interactivos generalmente siguen un patrón de esperar un comando y ejecutar un comando.

Si se considera la ejecución de un comando como un trabajo individual, se puede minimizar el tiempo de respuesta global ejecutando el trabajo más corto.

Problema

¿Cómo determinar cúal de los procesos ejecutables es el más corto?

Índice

Contexto de un proceso

Introducción

El contexto de un proceso es su estado definido por:

- su código,
- los valores de sus variables de usuario globales y sus estructuras de datos,
- el valos de los registros de la CPU,
- los valores almacenados en su entrada de la tabla de procesos y en su área de usuario y
- el valor de sus pilas de usuario y supervisor

Sistema operativo

El código del sistema operativo y sus estructuras de datos globales no se consideran parte del contexto de un proceso.

Contexto de un proceso

Introducción

- Cuando se ejecuta un proceso, se dice que el sistema se está ejecutando en el contexto de un proceso.

Cambio de contexto

Cuando el núcleo decide ejecutar otro proceso, realiza un cambio de contexto y guarda la información necesaria para reanudar la ejecución del proceso interrumpido en el mismo punto donde la dejó.

- Cuando el proceso cambia del modo usuario al modo supervisor, el núcleo guarda información para que el proceso pueda volver al modo usuario.

Cambio de modo (llamada al sistema)

El cambio de modo no se considera como un cambio de contexto.

Contexto de un proceso

Introducción

- Cuando se ejecuta un proceso, se dice que el sistema se está ejecutando en el contexto de un proceso.
- Cuando el proceso cambia del modo usuario al modo supervisor, el núcleo guarda información para que el proceso pueda volver al modo usuario.

Contexto de un proceso

Formalmente el contexto de un proceso es la unión de su contexto del nivel de usuario, su contexto de registro y su contexto del nivel de sistema.

Contexto de un proceso

Contexto de usuario

El contexto del nivel de usuario se compone de los segmentos de:

- Datos,
- Código y
- Pila,

además de las zonas de memoria compartida que se encuentran en la zona de direcciones virtuales del proceso.

Direcciones virtuales

Las partes del espacio de direcciones virtuales, que periódicamente no residen en memoria principal, también son parte del contexto de usuario.

Contexto de un proceso

Contexto de registros

El contexto de registros se compone de las siguientes partes:

- El **contador de programa**, que contiene la dirección de memoria de la siguiente instrucción que debe ejecutar la CPU.
- El **registro de estado del procesador** (*PS*), especifica el estado del hardware de la máquina. Este registro contiene el estado del harware referente a un proceso.

Contexto de un proceso

Contexto de registros

El contexto de registros se compone de las siguientes partes:

- El **apuntador a la pila**, apunta a la cima de la pila de usuario o de supervisor.

Crecimiento de la pila

La arquitectura de la máquina es la que dictamina si la pila crece hacia direcciones bajas o altas de la memoria.

- Los **registros de propósito general** contienen datos generados por el proceso durante su ejecución.

Contexto de un proceso

Contexto de sistema

El contexto del nivel de sistema de un proceso tiene una parte estática y una parte dinámica:

1. Parte estática:
   - La entrada en la tabla de procesos.
   - El área de usuario.
   - Tabla de regiones y tabla de páginas, que definen el mapa de transformación entre las direcciones del espacio virtual y las direcciones físicas.
2. Parte dinámica:
   - La pila de supervisor, que contiene los marcos de pila de las funciones ejecutadas en modo supervisor.
   - Una serie de capas que se almacenan a modo de pila. Cada capa contiene información necesaria para recuperar la capa anterior.

Contexto de un proceso

Contexto de sistema

- El núcleo introduce una capa de contexto cuando se produce una interrupción, una llamada al sistema o un cambio de contexto.
- La capa introducida es del último proceso que se estaba ejecutando.
- La capa extraída es la del proceso que pasará a ejecución.
- Cada una de las entradas de la tabla de procesos contiene información suficiente para recuperar las capas de contexto.

Ejecución de un proceso

Un proceso se ejecuta en su contexto, más exactamente, en su capa de contexto actual.

Distribución y planificación de procesos

Orígenes de UNIX

Las primeras versiones de UNIX ocupaban 42 KB y se ejecutaban, a principios de los años 70', en computadoras PDP-11 con un procesador de 16 bits y 144 KB de memoria[a].

[a] Richie & Thompson, 1974.

- UNIX fue concebido como un sistema de tiempo compartido por procesadores con un solo procesador.
- En sistemas monoprocesador, la distribución se realiza generalmente con desalojo por prioridad circular (*round robin*), utilizando colas múltiples realimentadas.

Distribución y planificación de procesos

- Un parámetro de diseño importante es el valor del *cuantum de tiempo*.
  1. Cuantums pequeños mejoran la disponibilidad del sistema.
  2. Cuantums grades mejoran el rendimiento, ya que reducen las sobrecargas por cambios de contexto.
- El planificafor es la parte del núcleo encargada de asignar las prioridades.

Cambio de prioridad

El planificador toma en cuenta para ello el uso reciente de la CPU por parte de cada proceso con objeto de incrementar la prioridad de aquellos que llevan más tiempo esperando la CPU.

Cada vez que un proceso cambia de prioridad, también es cambiado de cola para que el planificador le pueda asignar la CPU de acuerdo con su nueva prioridad.

Distribución y planificación de procesos

- La prioridad de un proceso se codifica a través de un número entero que varia entre 0 (prioridad más alta) y 127 (prioridad)
  1. El rango de valores reservado para la ejecución de procesos en modo supervisor es $[0, 49]$.
  2. Para el modo usuario el rango es $[50, 127]$.
- Los rangos mencionados garantizan una rápida respuesta de las llamadas al sistema.

Expropiación

Durante la ejecución en modo supervisor los procesos son desalojados a petición propia o por bloqueo, ya que los núcleos tradicionales de UNIX no son interrumpibles.

Distribución y planificación de procesos

- La prioridad en modo usuario se calcula con la expresión:

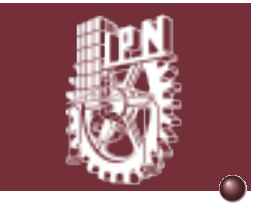
$$P_u = min_{P_u} + \frac{t_{CPU}}{4} + 2 \cdot P_{nice} \quad (1)$$

Donde:

min_{P_u}: valor más pequeño que puede tomar P_u,

t_{CPU}: es una medida del uso reciente de CPU del proceso,

P_{nice}: es un parámetro que elije el usuario entre $[0, 49]$



Distribución y planificación de procesos

- El valor de t_{CPU} es actualizado por el núcleo, incrementándose en 1, por cada pulso de reloj para el proceso que actualmente ocupa el CPU.
- El valor de t_{CPU} se decrementa D unidades cada segundo para el resto de los procesos.

$$D = \frac{2 \cdot \overline{N_P}}{2 \cdot \overline{N_P} + 1} \tag{2}$$

Donde:

$\overline{N_P}$: es el número de procesos que están esperando la CPU durante el último segundo,

Asignación de la CPU

Considerando estos criterios, la CPU se cede a los procesos que más tiempo llevan esperando.

Distribución y planificación de procesos

- Este esquema de asignación de prioridades dinámicas es sencillo y efectivo en sistemas de tiempo compartido.
- Sin embargo presenta los siguientes inconvenientes:
  1. Es ineficiente e introduce sobrecargas cuando se tiene que recalcular la prioridad de un número grande de procesos.
  2. No se puede reservar la CPU para un proceso determinado que así lo requiera.
  3. No se puede garantizar los tiempos de respuesta de los procesos y por lo tanto es aplicable a sistemas con requisitos de tiempo real.
  4. Produce inversión de prioridades entre los procesos en modo supervisor por que el núcleo no es interrumpible.
  5. Los procesos tienen poco control sobre su prioridad.

Índice

Sistemas con requisitos de tiempo real

UNIX

Para afrontar las necesidades de los sistemas de tiempo real ha sido necesario rediseñar el núcleo de los sistemas UNIX tradicionales y cambiar sus esquemas de distribución y planificación.

- La teoría sobre planificación de sistemas de tiempo real se ha venido desarrollando durante las últimas 4 décadas y los resultados más importantes, en sistemas monoprocesador, se han dado en: [Liu & Layland, 1973], [Leung & Whitehead, 1982], [Sha & Lehoczky, 1989] y [Lehoczky et al.,1989].

Planificación con prioridades fijas

- Un sistema de tiempo real está compuesto por un número fijo de tareas peródicas independientes. En la literatura de tiempo real se habla de tareas sin distinguir entre procesos y tareas.
- Dichas tareas se ejecutan en un sistema monoprocesador y son distribuidos con desalojo por prioridad y planificados con prioridad fija.
- Cada tarea se define por un conjunto de atributos temporales.

Atributo	Descripción
T_i	Periodo de la tarea
C_i	Tiempo de cómputo (WCET, worst case execution time)
D_i	Plazo de respuesta
P_i	Prioridad, a mayor P_i mayor prioridad
R_i	Tiempo de respuesta en el peor de los casos
U_i	Factor de utilización $U_i = \frac{C_i}{T_i}$

Planificación con prioridades fijas

Atributo	Descripción
T_i	Periodo de la tarea
C_i	Tiempo de cómputo (WCET, worst case execution time)
D_i	Plazo de respuesta
P_i	Prioridad, a mayor P_i mayor prioridad
R_i	Tiempo de respuesta en el peor de los casos
U_i	Factor de utilización $U_i = \frac{C_i}{T_i}$

Objetivo

El objetivo de la planificación es asignar una prioridad distinta a cada tarea para que al distribuirlas con desalojo pro prioridad se cumplan los requisitos temporales para cada tarea:

$$R_i \leq D_i \tag{3}$$

Planificación con prioridades fijas

- En este escenario, el criterio de prioridades monótonas en frecuencia, tambien llamado prioridad al más frecuente o *RMS* (*Rate Monotonic Scheduling*).
- Es óptimo para tareas periódicas independientes con plazos de respuesta iguales a su periodo $D_i = T_i$.
- Este criterio dice que las prioridades deben asignarse en orden inverso a los periodos, $T_i < T_j \rightarrow P_i > P_j$

Sha & Lehoczky, 1986

Si asignando prioridades con otro esquema garantizamos los plazos, con RMS también se garantizan. Si con RMS no se pueden garantizar los plazos, con cualquier otro esquema tampoco.

Planificación con prioridades fijas

- La prueba del factor de utilización total del sistema presentada en [Liu & Layland, 1973] es condición suficiente para que todas kas tareas cumplan sus requisitos temporales:

$$\sum_{i=1}^{N} \frac{C_i}{T_i} < N\left(2^{\frac{1}{N}} - 1\right) \tag{4}$$

- Esta condición es suficiente, pero no necesaria y, por lo tanto, si no se cumple no dice si el sustema es planificable o no.

Planificación con prioridades fijas

- Se puede recurrir al cálculo del tiempo de respuesta en el peor de los casos (R_i) de cada tarea, [Lehoczky et al., 1986]:

$$R_i = C_i + \sum_{i \in pM(i)} \left\lceil \frac{R_i}{T_i} \right\rceil \cdot C_j \tag{4}$$

Donde:

$pM(i)$: es el conjunto de índices de todas las tareas cuya prioridad es mayor que la prioridad τ_i, $pM(i) = \{j | P_j > P_i\}$.

$\lceil \cdot \rceil$: representa la función *redondeo a la alza.*

- Dado que no se puede despejar R_i:

$$w_i^{n+1} = C_i + \sum_{j \in pM(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil \cdot C_j \tag{5}$$

$$w_i^0 = C_i + \sum_{j \in pM(i)} C_j \tag{6}$$

Planificación con prioridades fijas

- El criterio de asignación de prioridades RMS deja de ser óptimo cuando alguna de las tareas cumple que $D_i < T_i$.
- En esas condiciones se emplea el criterio de prioridad al más urgente (DMS, *DeadLine Monotonic Scheduling*).
- DMS asigna la mayor prioridad a la tarea con menor plazo de respuesta, [Leung & Whitehead, 1982]:

$$D_i < D_j \rightarrow P_i > P_j \tag{7}$$

Factor de utilización

La prueba del factor de utilización total ya no es aplicable para comprobar si el sistema es planificable y tenemos que recurrir nuevamente al cálculo del pero tiempo de respues de cada tarea

Planificación con prioridades dinámicas

- Todo lo expuesto anteriormente es aplicable a sistemas planificados con prioridades fijas.

Planificación dinámica

En el caso de la planificación dinámica se demuestra que la condición necesaria y suficiente para que el sistema sea planificable es que el factor de utilización total no supere la unidad, [Lui & Layland, 1973]:

$$\sum_{i=1}^{N} \frac{C_i}{T_i} \leq 1 \tag{8}$$

Planificación con prioridades dinámicas

- Para conseguir un factor de utilización del 100 % hay que asignar en cada instante de planificación la máxima prioridad a la tarea más urgente (EDF, Earliets Deadline First).

EDF

Exige recalcular constantemente el tiempo de cómputo consumido por la tarea para determinar cuánto falta para que se concluya su plazo de respuesta.

Planificación de tareas dependientes

Escenario real

El escenario planteado hasta el momento donde las tareas sólo compiten por la CPU como único recurso compartido es el menos frecuente en la realidad.

- Lo común es que las tareas de un sistema con requisitos de tiempo real se comuniquen entre sí y accedan a recursos compartidos.
- La *inversión de prioridades* es una posible interferencia y afecta al tiempo de respuesta de las tareas.

Planificación de tareas dependientes

- Lo común es que las tareas de un sistema con requisitos de tiempo real se comuniquen entre sí y accedan a recursos compartidos.
- La *inversión de prioridades* es una posible interferencia y afecta al tiempo de respuesta de las tareas.

Inversión de prioridad

Se dice que un sistema presenta inversión de prioridades cuando una tarea es suspendida a la espera de que otra tarea de menor prioridad realice algún calculo necesario y la tarea de prioridad mayor ve alargado su tiempo de respuesta.

- Se han diseñado mecanismos de planificación dinámica que limitan la duración de la *inversión de prioridades*.

Planificación de tareas dependientes

Herencia de prioridad

Consiste en hacer que una tarea herede la prioridad de otra tarea más prioritaria a la que bloquea.

- A estos mecanismos se les suele llamar *protocolos de herencia de prioridad* y algunos de ellos son:

Planificación de tareas dependientes

- A estos mecanismos se les suele llamar *protocolos de herencia de prioridad* y algunos de ellos son:

Herencia simple: Cuando una tarea está bloqueando a otra más prioritaria, hereda la prioridad de ésta, y así, la prioridad dinámica de una tarea es el máximo de su prioridad básica y las prioridades de todas las tareas bloqueadas por ella.

Planificación de tareas dependientes

Herencia del techo de prioridad: el techo de prioridad Tp_k de un recurso ρ_k es la máxima prioridad de las tareas que lo usan:

$$Tp_k = max\ usa(i,k) \times Pe_i, i \in \{1, ...N\} \quad (9)$$

$$usa(i,k) = \begin{cases} 1 & si\ \tau_i\ usa\ alguna\ vez\ el\ recurso\ \rho_k \\ 0 & si\ no\ lo\ usa\ nunca \end{cases} \quad (10)$$

Con este protocolo la prioridad dinámica Pd_i de una tarea τ_i es el máximo de su prioridad estática y las prioridades de las tareas que bloquea.

Planificación de tareas dependientes

- En estas condiciones una tarea sólo puede bloquear un recurso si su prioridad dinámica es mayor que el techo de todos los recursos en uso por otras tareas.
  - Cada tarea se bloque una vez como máximo en cada ciclo.
  - Se impiden los interbloqueos.
  - Se impiden los bloqueas encadenados.