**10.2**

La lectura habla sobre la planificación en tiempo real en sistemas informáticos. Describe cómo estos sistemas son esenciales en aplicaciones como control de procesos industriales, robótica, telecomunicaciones y militares. También explica que en los sistemas en tiempo real, la importancia radica en no solo el resultado del cálculo, sino también en cuándo se produce el resultado.

Habla sobre las tareas en tiempo real, que pueden ser clasificadas como duras o blandas, dependiendo de si cumplir con sus fechas límite es crítico o no. Además, menciona si estas tareas son periódicas o aperiódicas.

Luego, se discuten las características de los sistemas operativos en tiempo real, como su determinismo, capacidad de respuesta, control de usuario, fiabilidad y operación suave contra fallas. También se explica cómo estos sistemas priorizan y responden a las solicitudes de manera rápida y confiable, con una mínima latencia de interrupción.

El texto también menciona cómo los sistemas en tiempo real manejan las fallas, priorizando la estabilidad y la preservación de datos y capacidad en caso de problemas. Finalmente, describe diferentes enfoques de planificación de tareas en tiempo real, desde simples colas hasta preferencia inmediata, dependiendo de la aplicación y su exigencia de tiempo de respuesta.

Se habla sobre la programación en tiempo real, una área activa de investigación en informática. Se describen diferentes enfoques de programación y se identifican clases de algoritmos basados en si el sistema realiza un análisis de programabilidad, si es estático o dinámico, y si produce un cronograma para las tareas en tiempo de ejecución.

Se detallan cuatro clases de algoritmos: estáticos basados en tablas, preventivos estáticos basados en prioridades, dinámicos basados en planificación y dinámicos de mejor esfuerzo. Cada uno se diferencia en cómo maneja la planificación de tareas en tiempo real y en qué momento toma decisiones sobre la ejecución de las mismas.

La programación estática basada en tablas es predecible pero inflexible, ya que cualquier cambio requiere rehacer el cronograma. La preventiva estática basada en prioridades asigna prioridades a las tareas según la duración de sus períodos. La dinámica basada en planificación intenta ajustar el cronograma para acomodar nuevas llegadas, y la de mejor esfuerzo asigna prioridades en función de las características de las tareas, sin realizar un análisis previo de viabilidad.

Se discute cómo la programación en tiempo real considera la hora de preparación, fechas límite de inicio y finalización, tiempo de procesamiento, requisitos de recursos, prioridad y estructura de subtareas de cada tarea. Además, se explora la importancia de la preferencia en la planificación de tareas y se presentan ejemplos para ilustrar diferentes técnicas de programación.

Se ilustra cómo diferentes esquemas de programación afectan el cumplimiento de los plazos de las tareas en un sistema informático. Se comienza describiendo un escenario donde se intenta utilizar un esquema de programación basado en prioridades, donde la computadora toma una decisión de programación cada 10 ms. Se muestra que este enfoque no garantiza el cumplimiento de los plazos para ambas tareas, ya que al otorgar prioridad a una sobre la otra, una tarea puede incumplir su fecha límite.

Luego, se propone un enfoque de programación con la fecha límite más temprana, que demuestra ser más efectivo para cumplir con los plazos de las tareas periódicas y predecibles. Se explica cómo este enfoque funciona programando siempre la tarea con la fecha límite más cercana, lo que garantiza que se cumplan todos los requisitos del sistema.

Después, se analiza un esquema para manejar tareas aperiódicas con fechas límite iniciales. Se muestra que un enfoque simple de programar la tarea con la fecha límite más temprana puede llevar a denegar el servicio a ciertas tareas, lo que representa un riesgo, especialmente cuando se trata de tareas aperiódicas. Se propone una mejora de esta política llamada "plazo más temprano con tiempos de inactividad no forzados", que permite que el procesador permanezca inactivo si no hay tareas listas, lo que mejora el rendimiento en ciertos casos.

Finalmente, se compara esta política con la política FCFS (primero en llegar, primero en ser servido), que muestra que en algunos casos no se cumplen los plazos de las tareas. Esto destaca la importancia de elegir un enfoque de programación adecuado según las características de las tareas y los requisitos del sistema.

Se describe la programación monótona (RMS), que asigna prioridades a las tareas periódicas según sus periodos. Se explica que en RMS, la tarea de mayor prioridad es aquella con el período más corto, y así sucesivamente. Cuando hay varias tareas disponibles para ejecución, se selecciona primero la que tiene el período más corto.

Se introduce una medida de la efectividad de un algoritmo de programación periódica: la capacidad de cumplir con todos los plazos estrictos. Se menciona que para que esto sea posible, la suma de las utilizaciones del procesador de las tareas individuales no puede exceder 1. Se proporciona una desigualdad que proporciona un límite al número de tareas que un algoritmo de programación perfecto puede manejar con éxito.

Se muestra que para RMS, esta desigualdad se cumple, lo que significa que si se utiliza RMS, todas las tareas se programarán correctamente. Se discute brevemente por qué RMS ha sido ampliamente adoptado a pesar de la disponibilidad de otros enfoques. Se señala que aunque hay otros enfoques que pueden permitir una mayor utilización general del procesador, RMS es preferido en la práctica debido a su estabilidad y facilidad de implementación en sistemas duros o complejos en tiempo real.

La inversión de prioridad es un fenómeno que puede ocurrir en cualquier sistema preventivo basado en prioridades y es particularmente relevante en el contexto de la programación en tiempo real. Un ejemplo famoso de esto fue en la misión Mars Pathfinder. Después de que el robot explorador aterrizó en Marte en 1997, comenzó a experimentar reinicios del sistema debido a la inversión de prioridad.

La inversión de prioridad ocurre cuando una tarea de menor prioridad bloquea un recurso que una tarea de mayor prioridad necesita, lo que obliga a la tarea de mayor prioridad a esperar. Este problema puede ser grave, especialmente cuando la duración de la inversión de prioridad depende de acciones impredecibles de otras tareas no relacionadas.

Para evitar la inversión de prioridad, se utilizan dos enfoques alternativos: el protocolo de herencia de prioridad y el protocolo de techo de prioridad. La herencia de prioridad implica que una tarea de menor prioridad hereda temporalmente la prioridad de cualquier tarea de mayor prioridad pendiente en un recurso compartido. Esto asegura que la tarea de menor prioridad no bloquee a la tarea de mayor prioridad. Este enfoque se utilizó para resolver el problema en la misión Mars Pathfinder.

En el enfoque del límite de prioridad, se asocia una prioridad a cada recurso, que es superior a la prioridad de las tareas que lo utilizan. Esto garantiza que una tarea que accede al recurso temporalmente adquiera una prioridad más alta, evitando así la inversión de prioridad. Una vez que la tarea completa su uso del recurso, su prioridad vuelve a su nivel normal.