TEMA 28

PROGRAMACIÓN EN TIEMPO REAL. INTERRUPCIONES. SINCRONIZACIÓN Y COMUNICACIÓN ENTRE TAREAS. LENGUAJES.

ÍNDICE

1. INTRODUCCIÓN

El software de tiempo real está muy acoplado en el mundo externo. Debe responder al ámbito del problema (el mundo real) en un tiempo dictado por el ámbito del problema. Debido a que el software de tiempo real debe operar bajo restricciones de rendimiento muy rigurosas, el diseño del software está conducido frecuentemente, tanto por la arquitectura del hardware como por la del software, por las características del sistema operativo, por los requisitos de aplicación y, tanto por los extras del lenguaje de programación como por aspectos de diseño.

Como cualquier sistema basado en computadora, un sistema de tiempo real debe integrar hardware, software, hombres y elementos de una base de datos, para conseguir adecuadamente un conjunto de requisitos funcionales y de rendimiento.

El problema para los sistemas de tiempo real es realizar la asignación adecuada de funciones y comportamiento a los elementos del sistema. El rendimiento en tiempo real es frecuentemente tan importante como la función, pero las decisiones de asignación relativas al rendimiento son, frecuentemente, difíciles de hacer con seguridad. ¿Puede un algoritmo de procesamiento cumplir varias ligaduras de tiempo o debe construirse un hardware especial para hacer el trabajo? ¿Puede un sistema operativo cumplir nuestras necesidades para un manejo eficiente de interrupciones, multitareas y comunicaciones, o debemos construir un ejecutivo a medida? ¿Puede el hardware especificado, acoplado con el software propuesto, cumplir los criterios de rendimiento? Estas y otras muchas preguntas deben ser solucionadas por el ingeniero de sistemas de tiempo real.

2. SISTEMAS EN TIEMPO REAL

Los sistemas de tiempo real generan alguna acción en respuesta a sucesos externos. Para realizar esta función, ejecutan una adquisición y control de datos a alta velocidad bajo varias ligaduras de tiempo y fiabilidad. Debido a que estas ligaduras son muy rigurosas, los sistemas de tiempo real están frecuentemente dedicados a una única aplicación.

Durante muchos años, los principales consumidores de sistemas de tiempo real eran los militares. Sin embargo, hoy la significativa reducción del coste del hardware ha hecho posible para la mayoría de las compañías, proporcionar sistemas (y productos) de tiempo real para diversas aplicaciones, que incluyen control de procesos, automatización industrial, investigación médica y científica, gráficos de ordenador, comunicaciones locales y de largo alcance, sistemas aeroespaciales, pruebas asistidas por ordenador y un vasto abanico de instrumentación industrial.

2.1. Aspectos de integración y de rendimiento

Entre los muchos aspectos relativos al diseño de tiempo real están: la coordinación entre las tareas de tiempo real, el procesamiento de interrupciones del sistema, el manejo de E/S para asegurar que no se pierdan datos, la especificación de las ligaduras de tiempo externas e internas del sistema y el asegurar la precisión de su base de datos.

Cada diseño de tiempo real relativo al software debe ser aplicado en el contexto del “rendimiento” del sistema. En la mayoría de los casos, el rendimiento de un sistema de tiempo real se mide como una o más características relativas al tiempo, pero también se utilizan otras medidas, tales como la tolerancia al fallo.

Algunos sistemas de tiempo real se han diseñado para aplicaciones en las que sólo el tiempo de respuesta o la transferencia de datos es crítica. Otras aplicaciones de tiempo real requieren la optimización de ambos parámetros bajo unas condiciones de cargas extremas. Y lo que es más, los sistemas de tiempo real deben manejar sus cargas máximas, mientras se ejecutan varias tareas simultáneamente.

Puesto que el rendimiento de un sistema de tiempo real se determina principalmente por el tiempo de respuesta del sistema y su razón de transferencia de datos, es importante comprender estos dos parámetros. El **tiempo de respuesta** del sistema es el tiempo en el que un sistema debe detectar un suceso

interno o externo y responder con una acción. Frecuentemente, la detección del suceso y la generación de la respuesta son tareas sencillas. Es el procesamiento de la información sobre el suceso para determinar la respuesta adecuada lo que puede implicar algoritmos complejos que consumen mucho tiempo.

Entre los parámetros clave que afectan al tiempo de respuesta está el *cambio de contexto* y la *latencia de la interrupción*. El cambio del contexto se refiere al tiempo y sobrecarga necesitado para conmutar entre tareas, y la latencia de interrupción es el tiempo que pasa antes de que el cambio sea realmente posible. Otros parámetros que afectan al tiempo de respuesta son la velocidad de cálculo y el acceso a memorias masivas.

La **razón de transferencia de datos** indica con que rapidez se introducen o salen del sistema los datos series o paralelos, tanto analógicos como digitales. Los vendedores de hardware, frecuentemente, subrayan los valores de tiempo y capacidad como características de rendimiento Sin embargo, las especificaciones hardware para el rendimiento son normalmente medidas aisladas y, frecuentemente, dicen poco respecto a la determinación del rendimiento global de un sistema de tiempo real. Por tanto, el rendimiento del dispositivo de E/S, la latencia del bus, el tamaño del buffer, el rendimiento del disco, y una serie de otros factores, aunque importantes, son sólo parte de la historia del diseño de sistemas de tiempo real.

Los sistemas de tiempo real se necesitan normalmente para procesar un flujo continuo de datos de llegada. El diseño debe asegurar que no falte ningún dato. Además, un sistema de tiempo real debe responder a los sucesos que son asíncronos. Por tanto, la secuencia de llegada y el volumen de los datos no pueden predecirse fácilmente de antemano.

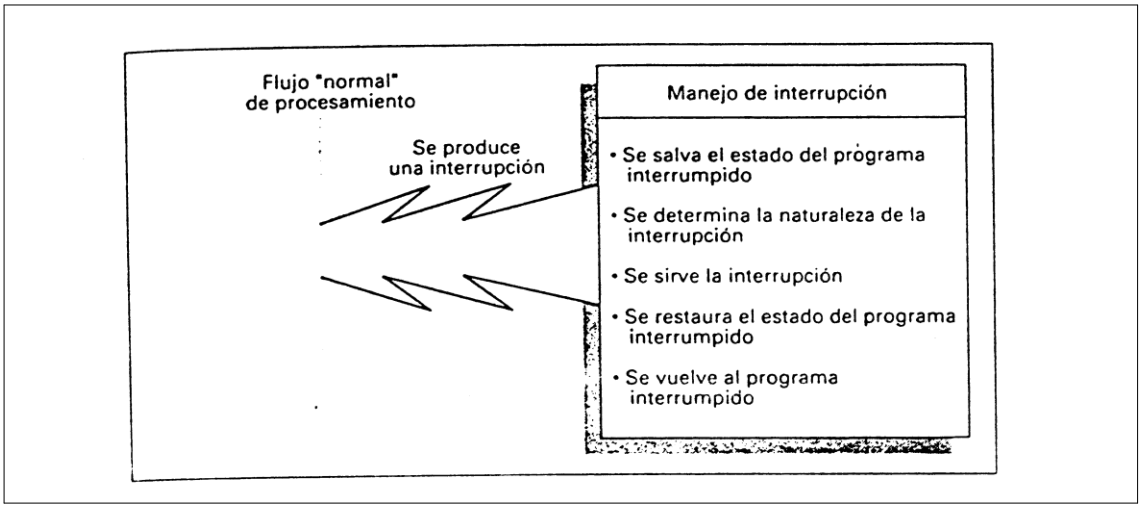
Aunque todas las aplicaciones de software deben ser fiables, los sistemas de tiempo real hacen una especial demanda de fiabilidad, reinicialización y recuperación de fallos. Debido a que el mundo real está siendo monitorizado y controlado, la pérdida de monitorización o control (o ambos) es intolerable en muchas circunstancias (p. ej.: un sistema de control de tráfico aéreo). Consecuentemente, los sistemas de tiempo real contienen mecanismos de restauración y recuperación de fallos y, frecuentemente, tienen incorporadas redundancias para asegurar la restauración.

Sin embargo, la necesidad de fiabilidad ha estimulado un continuo debate sobre si los sistemas "interactivos", tales como los sistemas de reservas de billetes y los cajeros automáticos de los bancos, también son de tiempo real. Por otra parte, tales sistemas interactivos deben responder a interrupciones externas dentro de tiempos de respuesta prescritos en el orden de un segundo. Por otra parte, no ocurre ninguna catástrofe si falla uno de esos sistemas en cumplir los requisitos de respuesta; en vez de ello, sólo se consigue una degradación del sistema.

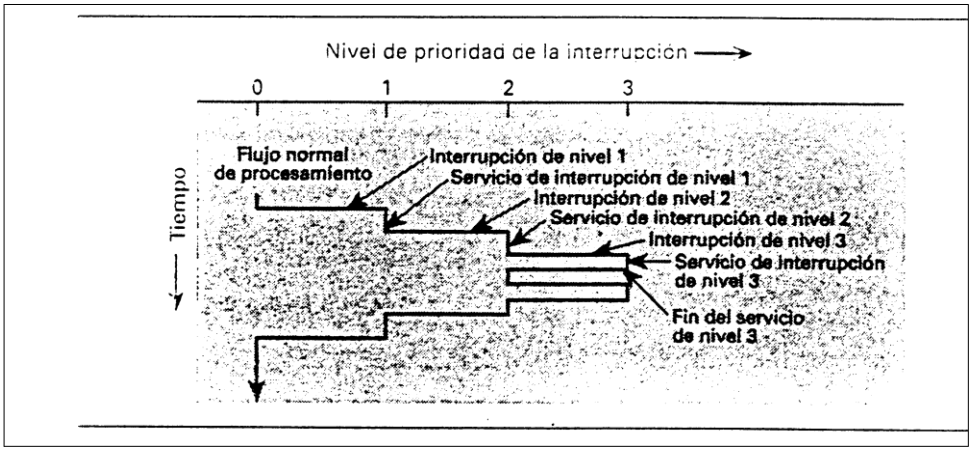
2.2. Manejo de interrupciones

Una característica que sirve para distinguir a los sistemas de tiempo real de cualquier otro tipo es el *manejo de interrupciones*. Un sistema de tiempo real debe responder a un estímulo externo -interrupción- en un tiempo dictado por el mundo externo. Debido a que, frecuentemente, se presentan múltiples estímulos (interrupciones), deben establecerse prioridades e interrupciones prioritarias. En otras palabras, la tarea más importante debe siempre ser servida dentro de las ligaduras de tiempo predefinidas, independientemente de otros sucesos.

El manejo de interrupciones supone, no sólo almacenar información, de forma que el ordenador pueda restablecer correctamente la tarea interrumpida, sino también evitar interbloqueos y bucles sin fin. EL enfoque global para el manejo de interrupciones se ilustra en la Figura. El flujo normal de procesamiento es "interrumpido" por un suceso que es detectado por el hardware del procesador. Un *suceso* es cualquier ocurrencia que necesita un servicio inmediato y puede ser generado por hardware o por software. Se salva el estado del programa interrumpido (es decir, se guardan todos los contenidos de los registros, los bloques de control, etc.) y se pasa el control a una rutina de servicio de interrupción, que bifurca al software apropiado para el manejo de la interrupción. Al terminar el servicio de la interrupción, se restaura el estado de la máquina y continúa el flujo normal de procesamiento.



En muchas situaciones, el servicio de interrupción de un suceso puede a su vez ser interrumpido por otro suceso de mayor prioridad. Pueden establecerse niveles de prioridad de interrupciones. Si de permite accidentalmente a un proceso de prioridad más baja interrumpir a uno de prioridad mayor, puede ser difícil restaurar los procesos en el orden correcto y puede producirse un bucle sin fin.



Para manejar las interrupciones y cumplir también las ligaduras de tiempo del sistema, muchos sistemas operativos de tiempo real hacen cálculos dinámicos para determinar si pueden cumplirse los objetivos del sistema. Estos cálculos dinámicos se basan en la frecuencia media de ocurrencia de sucesos, la cantidad de tiempo que le lleva el servirlos (si pueden ser servidos) y las rutinas que pueden interrumpirlos y temporalmente evitar su servicio.

Si los cálculos dinámicos muestran que es posible manejar los sucesos que pueden ocurrir en el sistema y también cumplir las ligaduras de tiempo, el ingeniero de sistemas debe decidir sobre un esquema de acción. Un posible esquema consiste en almacenar en un buffer los datos, de forma que puedan ser procesados rápidamente cuando el sistema esté preparado.

2.3. Sistemas operativos de tiempo real (SOTR)

Hoy, dos amplias clases de sistemas operativos se utilizan para los trabajos de tiempo real: **(1)** un SOTR diseñado exclusivamente para aplicaciones de tiempo real y **(2)** sistemas operativos de propósito general que se han reforzado para suministrar capacidades de tiempo real. El uso de un **ejecutivo de tiempo real** hace factible el rendimiento de tiempo real para un sistema operativo de propósito general. Comportándose como software de aplicación, el ejecutivo ejecuta varias funciones del sistema operativo -particularmente las que afectan al rendimiento de tiempo real- de una forma más rápida y eficiente que el sistema operativo de propósito general.

Todos los sistemas operativos deben tener un mecanismo de planificación de prioridades, pero un SOTR debe dar un **mecanismo de prioridades** que permita que las interrupciones de prioridad alta tengan preferencia sobre la menos importante. Además, debido a que las interrupciones ocurren en respuesta a sucesos asíncronos no recurrentes, deben ser servidas sin consumir primero un tiempo de carga de un programa de disco. Consecuentemente, para garantizar el tiempo de respuesta requerido, un sistema operativo de tiempo real debe tener un **mecanismo de bloqueo de memoria**, esto es, mantener unos mínimos programas en memoria principal, de forma que se evite la sobrecarga de almacenamiento en la misma.

Para determinar qué tipo de sistema operativo de tiempo real es más adecuado a una aplicación, pueden definirse y evaluarse medidas de la calidad de SOTR. El tiempo de cambio de contexto y el de latencia de interrupción (discutido anteriormente) determinan la capacidad de manejo de interrupciones, el aspecto más importante de un sistema de tiempo real. El **tiempo de cambio de contexto** es el tiempo que el sistema operativo se toma para almacenar el estado de la computadora y los contenidos de los registros, de forma que pueda volver a la tarea de procesamiento después de servir a la interrupción.

La **latencia de interrupción**, el tiempo máximo que pasa antes de que el sistema pueda conmutar una tarea, ocurre porque en un sistema operativo existen frecuentemente caminos de procesamiento críticos no rentables, que deben ser terminados antes de que pueda procesarse una interrupción. La longitud de estos caminos (el número de instrucciones requeridas antes de que el sistema pueda servir a una interrupción) indica el tiempo perdido en el caso peor. El peor caso se presenta si una interrupción de alta prioridad se genera inmediatamente después de que el sistema entra en un camino crítico entre una interrupción y un servicio de interrupción. Si el tiempo es demasiado largo, el sistema puede perder un trozo irrecuperable de datos. Es importante para el diseñador conocer el tiempo de retraso, de forma que el sistema pueda compensarlo.

Muchos sistemas operativos ejecutan procesamiento multitarea o concurrente, otro requisito importante para los sistemas de tiempo real. Pero para ser viable en la operación en tiempo real, el solapamiento del sistema debe ser bajo, en términos de tiempo de conmutación y espacio de memoria usado.

2.4. Sincronización y comunicación de tareas

Un sistema de multitarea debe suministrar un mecanismo por el que las tareas se pasen información unas a otras, así como para asegurar su sincronización. Para estas funciones, los sistemas operativos y los lenguajes con soporte de tiempo real, utilizan frecuentemente semáforos de colas, buzones o sistemas de mensajes. Los **semáforos** suministran sincronización y señalización pero no contienen información. Los **mensajes** son similares a los semáforos, excepto en que ellos llevan una información asociada. Por otra parte, los **buzones** no señalizan la información sino que la contienen.

Los **semáforos de colas** son primitivas de software que ayudan a gestionar el tráfico. Suministran un método para dirigir varias colas -por ejemplo, colas de tareas en espera de recursos, acceso a base de datos o dispositivos, así como colas de recursos y dispositivos. Los semáforos coordinan (sincronizan) las tareas en espera con lo que estén esperando, sin dejar que las tareas o recursos interfieran entre sí.

En un sistema de tiempo real, los semáforos se utilizan frecuentemente para implementar y gestionar buzones. Los buzones se almacenan temporalmente en lugares (también llamados **buffers** o *almacenes de mensajes*) para enviar mensajes de un proceso a otro. Un proceso produce una información, la sitúa en el buzón y luego señala a un proceso consumidor que hay una información en el buzón para que la utilice.

Algunos métodos para los sistemas operativos de tiempo real o sistemas de soporte en tiempo de ejecución ven los buzones como la forma más eficiente de implementar comunicaciones entre procesos. Algunos sistemas de tiempo real suministran un lugar para enviar y recibir referencias a los datos del buzón. Esto elimina la necesidad de transferir todos los datos -ahorrando así tiempo y sobrecarga.

Un tercer método para la comunicación y sincronización entre procesos es un **sistema de mensajes**. Con un sistema de mensajes, un proceso envía un mensaje a otro. El último se activa entonces automáticamente por el sistema de soporte de tiempo de ejecución o sistema operativo para que procese el mensaje. Tal sistema incurre en sobrecarga debido a la transferencia real de la información, pero suministra una mayor flexibilidad y facilidad de uso.

2.5. Bases de datos de tiempo real

Como muchos sistemas de procesamiento de datos, los sistemas de tiempo real, frecuentemente, van junto con una función de gestión de base de datos. Sin embargo, puede parecer que las *bases de datos distribuidas* constituyen el método preferido en los sistemas de tiempo real, debido a que la multitarea es muy común y que los datos se procesan frecuentemente en paralelo. Si la base de datos es distribuida y no centralizada, las tareas individuales pueden acceder a sus datos de forma más rápida, fiable y con menos cuellos de botella. El uso de una base de datos distribuida para aplicaciones de tiempo real divide el *tráfico* de entrada/salida y acorta las colas de las tareas en espera, para acceder a una base de datos. Además, un fallo de una base de datos raramente causará el fallo del sistema entero si se construyen con redundancia.

3. LENGUAJES DE TIEMPO REAL

Debido a los requisitos especiales de rendimiento y de fiabilidad demandados por los sistemas de tiempo real, la elección del lenguaje de programación. Pueden usarse con efectividad muchos lenguajes de programación de propósito general (p. ej.: C, FORTRAN, Modula-2) para aplicaciones de tiempo real. Sin embargo, una clase llamada *lenguajes de tiempo real* (p. ej.: Ada, Jovial, HAL/S, Chill y otros) se utilizan frecuentemente en aplicaciones especializadas de comunicaciones y militares.

Varias características hacen un lenguaje de tiempo real diferente de un lenguaje de propósito general. Estas incluyen la capacidad de multitarea, construcciones para implementación directa de funciones de tiempo real y características modernas de programación que ayuden a asegurar la corrección del programa.

Es importante que el lenguaje de programación soporte directamente la multitarea debido a que los sistemas de tiempo real deben responder a sucesos asíncronos que ocurren simultáneamente. Aunque muchos SOTR dan capacidad de multitarea, frecuentemente existe software de tiempo real empotrado sin un sistema operativo. En vez de ello, las aplicaciones empotradas se escriben en un lenguaje que da un soporte de tiempo real suficiente para la ejecución del programa en tiempo real. El soporte de tiempo de ejecución requiere menos memoria que un sistema operativo y puede ser adaptado a una aplicación, incrementando así el rendimiento.

Un sistema de tiempo real que haya sido diseñado par acomodar múltiples tareas debe también acomodar la sincronización entre tareas. Un lenguaje de programación que soporte directamente primitivas de sincronización, tales como SCHEDULE, SIGNAL y WAIT, simplifica mucho la traducción del diseño al código. La orden SCHEDULE planifica un proceso basándose en el tiempo o en un suceso; las órdenes SIGNAL y WAIT manipulan un *semáforo*, que facilita la sincronizaci6n de tareas concurrentes.

Finalmente, son necesarias facilidades que permitan una programación fiable, debido a que los programas de tiempo real son frecuentemente grandes y complejos. Estas características incluyen la programación modular, un fuerte reforzamiento de la tipificación de los datos y una multitud de otras construcciones de control y definición de datos.

4. BIBLIOGRAFÍA

Pressman, R.S.
Ingeniería del software
Mc Graw-Hill

Alberto Prieto
Introducción a la Informática
Mc Graw-Hill, 2ª edición, 1997

Alfonso Ureña López
Fundamentos de Informática
Ra-ma, 1997