4. Sistemas Operativos de Tiempo Real

Contenido

4.1 Introducción

En muchas ocasiones, para la programación de los sistemas de tiempo real, se utiliza un sistema operativo que apoye las características necesarias en este tipo de programación. Este hecho es especialmente necesario cuando el lenguaje de programación utilizado no ofrece los servicios necesarios para el tiempo real.

No todos los sistemas operativos son válidos para la programación en tiempo real. Es importante poder distinguir cuales son válidos y cuales, por sus características, pueden presentar problemas en estos sistemas.

Existen métricas que permiten obtener una medida de las características de tiempo real de un sistema operativo. Estas permiten comparan con valores numéricos las prestaciones de distintos sistemas operativos y de distintas plataformas.

Existen en la actualidad muchos sistemas operativos de tiempo real. Unos comerciales y otros de investigación que son de libre distribución. Por su gran difusión, haremos algunos comentarios sobre la utilización del Linux para Tiempo Real y algunas adaptaciones para mejorar sus prestaciones..

Por último, estudiaremos las características más importantes del estándar para tiempo real POSIX 1003.4 que se está imponiendo como interface estándar para la utilización de estos sistemas operativos.

4.2 Requerimientos

Los Sistemas Operativos de Tiempo Real, deben de cumplir ciertas exigencias para poder utilizar en el control de sistemas de tiempo real.

Estas exigencias van encaminadas a garantizar la correcta ejecución temporal de las tareas, tanto en lo referente al momento de la activación, como en la finalización dentro de los plazos permitidos.

Las características deseables de los Sistemas Operativos de Tiempo Real son:

4.2.1 Garantizar la correcta ejecución de todas las tareas críticas.

El sistema operativo debe ser capaz de cumplir los plazos de ejecución de las tareas críticas, aún en el caso de sobrecarga del sistema. Por tanto, la cantidad de tareas críticas admisibles deben estar limitadas.

Se debe disponer de algún mecanismo o criterio que nos permita saber si nuestra aplicación está dentro de este límite o lo sobrepasamos.

El ser capaz de predecir en tiempo de ejecución que un conjunto de tareas va a ser planificable es un problema que no está resuelto. Lo normal en este sentido es que el sistema operativo permita la utilización de políticas de planificación que están bien estudiadas teóricamente y en está teoría se ofrecen métodos para asegurar la planificabilidad.

4.2.2 Administrar adecuadamente el uso de los recursos compartidos.

En una aplicación de tiempo real, es extraño que se realice con un conjunto de tareas, todas ellas independientes. Lo normal es que interactúen unas con otras o que utilicen recursos comunes y, por tanto, que se tengan que sincronizar. Normalmente esta sincronización la realiza el sistema operativo.

En la programación de tiempo real no basta una simple sincronización, sino que se debe evitar ciertas situaciones que no serían aceptables, como por ejemplo los interbloqueos y la inversión de prioridad.

La sincronización entre bloqueos afecta a la planificabilidad de un sistema, sobre todo si en ella intervienen tareas críticas. Estos efectos se deben tener en cuenta dentro de la política de planificación que permite utilizar el sistema operativo.

4.2.3 Buen tiempo de respuesta a las tareas que no tengan plazo de terminación.

Las tareas que no son urgentes y, por tanto, que no tienen plazo de finalización que cumplir, siempre se podrían ejecutar en el tiempo sobrante como tareas de fondo. Ahora bien, en algunos casos interesa ejecutarlas de forma más eficiente.

La forma de mejorar la ejecución del trabajo no urgente es retrasando la ejecución del trabajo urgente pero sin que se llegue a sobrepasar sus límites de ejecución y, por tanto, disponer de tiempo para adelantar la ejecución del trabajo no urgente. Esto supone un cambio en el mecanismo de planificación que debe tenerse en cuenta dentro del sistema operativo que es quien realiza la planificación.

4.2.4 Recuperación ante fallos software y hardware.

Es frecuente la utilización de los sistemas de tiempo real en aplicaciones de control. En estos sistemas no es aceptable una parada del sistema o un comportamiento incontrolado.

En la medida que el sistema operativo ofrezca posibilidades de controlar los fallos software y hardware facilitará la realización de aplicaciones fiables ante fallos.

Existe una dificultad adicional que consiste en el cumplimiento de las restricciones temporales aún en el caso de producirse un fallo. En este sentido, todo sistema de tiempo real estricto debería ser tolerante a fallos pues los límites temporales para las respuesta ante los eventos críticos no deberían sobrepasarse aunque se produjera algún tipo de fallo.

Alguna de las facilidades que ofrecen los sistemas operativos son:

- Funcionamiento distribuido. La ejecución de la aplicación no se realiza sobre una única máquina sino que intervienen más de una. En el caso de algún fallo localizado el sistema podría seguir funcionando con el resto de máquinas.

- Procesos replicados. Consiste en facilitar la ejecución y sincronización de procesos idénticos en varias máquinas.

- Grupos de puertos. Son puertos especiales de los que puede leer más de una tarea y todas reciben lo mismo.

- Soportar puntos de recuperación y vuelta atrás. Son más fáciles de manejar a nivel de sistema operativo aprovechando los mecanismos de cambios de contexto y de protección de memoria en los procesos.

4.2.5 Soportar los cambios de modo.

A menudo un sistema debe soportar varios modos de funcionamiento. Por ejemplo, el sistema de control de un avión realizará funciones distintas en las fases de despegue, aterrizaje, vuelo normal y vuelo automático. En cada fase o modo de funcionamiento el control lo realizará un conjunto de tareas distinto.

El cambio de modo no se puede producir arbitrariamente, deteniendo todas las tareas que intervienen en un modo y arrancando las de otro, pues esto seguramente llevará a que se dejen de cumplir algún límite temporal.

El sistema deberá facilitar el cambio ordenado del conjunto de tareas, permitiendo que durante un tiempo coexistan tareas de uno y otro modo, asegurando que no se viola ninguna restricción temporal.

4.2.6 Buen tiempo de respuesta a las interrupciones.

Las interrupciones juegan un papel muy importante en los sistemas de tiempo real.

Cuando ocurre un suceso externo un dispositivo hardware lo detecta y mueve una señal que le llega al procesador provocando una solicitud de interrupción justo en el instante de producirse el suceso. Si el sistema debe producir una respuesta acotada en el tiempo para esta interrupción, es importante el tiempo que tarda el sistema en reaccionar a la interrupción, es decir, el tiempo que transcurre desde que se solicita la interrupción hasta que se activa su manejador.

Existen tres factores importantes que influyen en el tiempo de respuesta a las interrupciones:

1) **Niveles de interrupción**. La disponibilidad de distintos niveles de interrupción, es decir, la capacidad de que una interrupción de nivel superior interrumpa a un manejador de una interrupción de nivel inferior, permite organizar las interrupciones de forma que una interrupción importante no pueda ser bloqueada por otra de menos importancia, aunque esta segunda precise de un manejador de larga duración.

2) **Duración del tratamiento de las interrupciones**, es decir, el tiempo máximo que el sistema está tratando una interrupción. Durante el tratamiento de una interrupción, se inhiben el tratamiento de las interrupciones para evitar inconsistencias en el manejo del hardware asociado a la interrupción, al menos las de un determinado tipo o nivel. La duración máxima determinará la máxima frecuencia a la que pueden producirse las interrupciones.

3) **Tiempo de latencia de interrupción**. Es el tiempo que el sistema inhibe las interrupciones en zonas críticas que no puede ser interrumpido. Los tiempos de latencia de interrupción provocarán retrasos en la activación de los manejadores produciendo, no solo el retraso de la respuesta a la interrupción sino que puede provocar que se pierdan interrupciones.

El primer factor viene dado por el hardware utilizado, aunque depende de la utilización que se de a las posibilidades del hardware en el diseño del sistemas. Los otros dos factores son exclusivamente software y dependen del diseño del sistema operativo.

4.2.7 Eficiencia en los cambios de contexto (procesos ligeros).

Los cambios de contexto se producen cada vez que se modifica la tarea en ejecución. Cuando corresponde ejecutarse una nueva tarea esta no comenzará inmediatamente, sino que su ejecución se verá retrasada por el tiempo que necesita el procesador para guardar el estado de la tarea que estaba en ejecución en ese momento y cargar el estado con el que debe continuar la nueva tarea.

Cuanto más tiempo le lleve al procesador el cambio de contexto, más se retrasará la ejecución de la tarea. El tiempo de cambio de contexto se puede considerar como tiempo de ejecución que habría que añadir a la tarea que se activa pero, en cualquier caso, en los sistemas de tiempo real este tiempo debe ser lo más reducido posible, sobre todo en sistemas que por sus características temporales se realizan cambios de tarea muy frecuentemente.

Puede ocurrir que la activación de una tarea se de por la llegada de una interrupción. En tal caso, el cambio de contexto lo realiza el manejador de la interrupción, por que le urge que sea lo más rápido posible para afectar al tratamiento de las interrupciones lo menos posible.

Para facilitar el cambio de contexto se utilizan los procesos ligeros (también llamados hilos o threads). Estos son tareas que forman parten de un mismo proceso, compartiendo todas sus propiedades desde el punto de vista del sistema operativo (espacio de direcciones, ficheros abiertos, control de privilegios, terminal de entrada / salida, …) pero que poseen caminos de ejecución independientes. De esta forma, la conmutación entre procesos ligeros pertenecientes a un mismo procesos puede hacerse de forma mucho más rápida que el cambio de procesos independientes.

Muchos sistemas operativos para desarrollo de aplicaciones empotradas soportan en realidad un único proceso con múltiples hilos. Su funcionamiento es mucho más eficaz que los sistemas operativos que manejan procesos (pesados) y poseen las misma prestaciones de concurrencia y sincronización.

4.3 Métricas

Los programas estándar de referencia (benchmark) como son el Wheststone, Drystone y Linpack miden típicamente la velocidad de una CPU en un entorno de una sola tarea. Sin embargo una CPU puede tener una eficiencia muy alta y escasas prestaciones para tiempo real. Por tanto se necesitan unas medidas específicas para los ordenadores dedicados a sistemas en tiempo real.

Veamos cuatro metodologías y sus métricas relacionadas para medir objetivamente el rendimiento de tiempo real de una CPU:

1) Métrica Rhealstone
2) Tiempo de latencia en la activación de procesos
3) Medida tridimensional
4) Real/Stone Benchmark

4.3.1 Métrica Rhealstone

Consiste en obtener valores cuantitativos de seis medidas que influyen en el comportamiento de los sistemas de tiempo real.

Los factores a medir en un sistema operativo, según esta métrica son:

- **Tiempo de cambio de tarea**: tiempo medio que emplea el sistema en conmutar entre dos tareas independientes de la misma prioridad.
- **Tiempo de prioridad**: tiempo medio que emplea el sistema en conmutar de una tarea de menos prioridad a otra de más.
- **Tiempo de latencia de interrupción**: tiempo que transcurre desde que la CPU recibe una interrupción y activa la rutina que la sirve.
- **Tiempo de transición en un semáforo**: retraso introducido entre que una tarea libera un semáforo y otra tarea que estaba a la espera se activa.
- **Tiempo de ruptura**: Tiempo que necesita el sistema para suspender una tarea que utiliza un recurso usado por una tarea de menor prioridad y activar esta última.
- **Velocidad de flujo de datos en entrada/salida**: Es el flujo de información en Kbytes por segundo que una tarea puede enviar a otra utilizando primitivas del sistema operativo y sin utilizar mensajes situados en buffers de memoria compartida

4.3.2 Tiempo de latencia en la activación de procesos

Esta métrica mide el intervalo de tiempo entre el instante en que el sistema recibe una interrupción hasta que se activa la tarea que va a responder a este estímulo. Las componentes más importantes de este tiempo son el tiempo de latencia de interrupciones y el tiempo de cambio de contexto, pero hay otras componentes de esta medida:

- Tiempo de respuesta a la interrupción
  - Tiempo de retardo hardware en atender la interrupción
  - Finalización de la instrucción actual
  - Tiempo de latencia de la interrupción
- Rutina de tratamiento de la interrupción
  - Pre-procesado
  - Rutina de servicio de interrupción
  - Post-procesado
- Decisión de cambio de contexto
- Cambio de contexto

4.3.3 Medida tridimensional

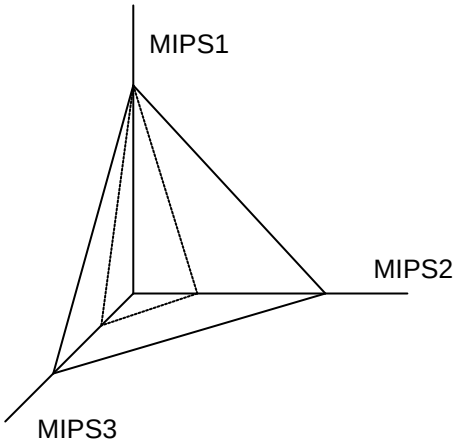
Incluye las tres medidas más importantes de los computadores en tiempo real:

(a) Velocidad de cálculo de la CPU, medido en MIPS1 (Millones de Instrucciones por Segundo).
(b) Capacidad de tratar interrupciones, medido en MIPS2 (Millones de Interrupciones Por Segundo)
(c) Rendimiento de Entradda/Salida, medido en MIPS3. (Millones de operaciones de E/S (Mbytes/seg) Por Segundo)

Estas tres medidas no son independientes, normalmente en las condiciones en que se obtiene el máximo para una de estas medidas se reducen las prestaciones referentes a las otras dos medidas. La relación con la que se degradan estas medidas como consecuencia de una carga del sistema indica la efectividad del sistema en aplicaciones de tiempo real.

La variable MIPS1 disminuirá cuanto mayor sea la carga de MIPS2 y MIPS3.

Si representamos en un espacio de tres dimensiones MIPS1 en función de MIPS2 y MIPS3 se obtiene aproximadamente un plano de la forma:



La línea de puntos representa una CPU convencional. Una CPU de tiempo real, para un valor equivalente de MIPS1 presenta mejor rendimiento al aumentar la carga de MIPS2 y MIPS3.

El volumen de la figura obtenida da una idea de las prestaciones de tiempo real de la CPU.

Una medida más práctica es el volumen del sólido rectangular definido por los puntos de corte con los ejes:

$$\text{Volumen} = M1 \bullet M2 \bullet M3$$

Se puede definir un valor equivalente para el rendimiento de las CPU de tiempo real, teniendo en cuenta que MIPS2 suele ser en dos ordenes de magnitud inferior a MIPS1 y MIPS3, como:

$$MIPS_e = \sqrt[3]{M1 \bullet (M2 \bullet 100) \bullet M3}$$

4.3.4 Real/Stone Benchmark

Se trata de un test artificial para simular un entorno del mundo real. Se trata de un benchmark puramente software que no necesita ningún test del hardware y que puede ser portado fácilmente a diferentes plataformas hardware.

Consiste en unos tests que miden las siguientes características de los sistemas en tiempo real:

(a) Sensibilidad del sistema
(b) Priorizabilidad del sistema
(c) Rendimiento del sistema para las E/S

Para los dos primeras medidas el test se divide en un conjunto de procesos en tiempo real denominados RTn (n = 1, 2, 3, ...) activados por timers internos.

Otro denominado TRIANG de mayor prioridad genera ondas triangulares y guarda los datos en memoria global. Se utiliza para simular una periférico de entrada que muestrea datos desde un proceso externo con una cierta cadencia de activación (240 Hz en el caso del Real/Stone).

Los procesos RTn se activan por timers internos con diferentes cadencias (240 Hz, 120 Hz, 80 Hz, etc.), submúltiplos de la frecuencia del proceso TRIANG. Todos tienen el mismo código y se ejecutan con la misma prioridad. Se puede seleccionar cualquier número de procesos para simular un entorno del mundo real. Estos procesos utilizan el muestreo generado por el proceso TRIANG para regenerar la onda triangular y evaluar una función de error. El error indica la eficacia en regenerar la onda triangular original generada por TRIANG..

Si el sistema que se evalúa tiene una alta sensibilidad (a) y administra bien las prioridades (b) es de esperar que se puedan activar gran número de procesos antes de que el error sea elevado. Si no es así el error aumentará rápidamente al aumentar la carga de procesos.

Para evaluar la última característica (c) otro proceso llamada WHETR que se ejecuta a baja prioridad calcula el valor residual del coeficiente Whetstone en MIPS y que equivaldrá a la capacidad de cálculo residual para los programas de aplicación. Otro proceso llamado WHET evalua los MIPS que ofrece la máquina cuando no hay carga. El factor de carga para la aplicación será:

$$R = \frac{\text{MIPS aplicación}}{\text{MIPS sin carga}}$$

El resultado del test Real/Stone se expresa en tres gráficas:

- Capacidad de proceso (MIPS de aplicación) en función del número de procesos de tiempo real
- Factor de carga para la aplicación (R) en función del número de procesos
- Error en función del número de procesos

El segundo test que mide también la priorizabilidad del sistema tiene una estructura similar, sólo que se sustituye el proceso WHETR por el proceso SYSCALL que se ejecuta también con la prioridad más baja. Este proceso se encarga de ejecutar llamada al sistema de larga duración, por ejemplo *fork* y *exec*.

Si el sistema que se evalúa posee un kernel interrumpible el proceso SYSCALL será interrumpido inmediatamente y el control se transferirá rápidamente a los procesos en tiempo real. Los procesos RTn leerán la onda generada por TRIANG y la regenerarán correctamente.

Si las llamadas al sistema no son interrumpibles los procesos RTn tendrán que esperar la finalización de las llamadas al sistema y no regenerarán correctamente la onda.

El resultado de este test se expresa con una gráfica representando el error obtenido en función del número de procesos.

4.4 Linux y tiempo real

4.4.1 Características más importantes

El sistema operativo Linux, al igual que el Unix, no ha sido diseñado como un sistema operativo para tiempo real, por lo que presenta algunos inconvenientes para ser usado en aplicaciones con restricciones de tiempo duras. Las más importantes son:

- Linux desactiva las interrupciones del procesador para proteger secciones críticas. Así como la mayoría de drivers las desactivan durante unos pocos microsegundos, el subsistema de disco puede desactivarlas durante varios cientos de microsegundos de una vez. Esto daña seriamente la predicibilidad del sistema cuando la interrupción de reloj es bloqueada durante un tiempo tan largo.

- La resolución del timer del Linux es actualmente 10 milisegundos, lo que puede ser pequeña para algunas aplicaciones. Por otro lado, las funciones activadas por el timer son encoladas para ejecutarse a la salida de un servicio de Linux. Por tanto, los servicios largos pueden demorar en exceso la ejecución de funciones activadas por el timer.

- La librería de pthreads de Linux no soporta las extensiones de tiempo real.

4.4.2 Extensiones de Tiempo Real en Linux

Para resolver o minimizar estos inconvenientes se han creado varias extensiones de Linux, cada una con sus ventajas e inconvenientes.

4.4.2.1 UTIME. Servicios de tiempo de alta resolución

Ofrece un reloj con granularidad de microsegundos. Las llamadas al sistema con parámetros de tiempo pueden usar esta precisión.

4.4.2.2 KURT. Kansas University Real-Time System

Utiliza UTIME y extiende el Linux para aplicaciones de tiempo real.

Está diseñado para aplicaciones que puedan tolerar alguna demora en los plazos de ejecución (firm realtime).

Distingue los procesos de tiempo real de los procesos estándar, y ofrece tres modos de funcionamiento.

1. En el **Modo Dedicado** solo se pueden ejecutar procesos de tiempo real.

2. En el **Modo Normal** todos los procesos se ejecutan como procesos estándar de Linux. En este modo es como arranca el sistema.

3. En el **Modo Mixto** los procesos que no son de tiempo real se ejecutan en background.

Todos los procesos de Tiempo Real son periódicos o no periódicos, en los que se conocen sus tiempos de activación o el tiempo de activación de sus eventos respectivamente.

Los procesos se registran en un servicio de KURT, indicando sus parámetros y la política de planificación.

El cambio a un modo de Tiempo Real (Dedicado o Mixto) lo realiza un proceso ejecutivo una vez conocidos todos los tiempos de activación y haber calculado los instantes de activación de cada proceso y evento.

En KURT persisten varias fuentes que introducen impredecibilidad. Por ejemplo, puede manejar tablas de tiempos de activación grandes que residen parcialmente en memoria. Y también persisten los tiempos largos de desactivación de interrupciones.

Pese a sus ventajas, no es adecuado para las aplicaciones de tiempo real estricto (hard realtime).

4.4.2.3 RT Linux

Está diseñado para ofrecer prestaciones de tiempo real estricto.

Asume que una aplicación se puede dividir en dos partes:

1. La parte de tiempo real que corre en el núcleo de tiempo real.

2. La parte no de tiempo real que corre en Linux.

Estas partes se comunican por unos canales llamados FIFO, que en la parte de tiempo real se fijan en memoria. Se llaman RT-FIFO.

En los procesos Linux los FIFO se ven como periféricos.

Las lecturas / escrituras en RT-FIFO son no bloqueantes y atómicas.

El problema de la inhibición de interrupciones en el Linux se resuelve emulando las interrupciones hardware por interrupciones software.

Si llega una interrupción que es para el Linux, el núcleo de tiempo real se la pasa sólo cuando el Linux tiene las interrupciones habilitadas.

La parte de tiempo real de la aplicación se escribe como módulos cargables en el kernel. Todas las tareas de tiempo real se ejecutan en el espacio del kernel.

Las tareas en cada módulo pueden tener su propio planificador. La versión actual ofrece:

- Planificador Rate Monotonic (RM)

- Planificador Earliest Deadlien First (EDF).