TEMA 28

Programación en tiempo real. Interrupciones. Sincronización y comunicación entre tareas.Lenguajes.

Índice

1. INTRODUCCIÓN 120
2. CARACTERÍSTICAS 120
2.1. Control 121
2.2. Comunicación 121
2.3. Concurrencia 121
2.4. Dependencia del tiempo 122
2.5. Fiabilidad y seguridad 122
2.6. Complejidad de los sistemas 122
3. APLICACIONES EN TIEMPO REAL 122
4. PROGRAMACIÓN CONCURRENTE 123
4.1. Concepto 123
4.2. Tareas concurrentes 123
4.3. Planificación de tareas 124
5. INTERRUPCIONES 125
5.1. Consulta de estado 126
5.2. Interrupciones 126
5.3. Interrupciones vectorizadas 127
6. SINCRONIZACIÓN Y COMUNICACIÓN ENTRE TAREAS 127
7. LENGUAJES DE TIEMPO REAL 128
7.1. Lenguajes convencionales 128
7.2. Modula-2 y Ada 128
7.3. Eiffel y Java 129
8. CONTEXTUALIZACION 130

1. INTRODUCCIÓN

Según Young, un sistema de tiempo real es ”cualquier actividad de proceso de información o sistema que tiene que responder a estímulos generados externamente dentro de un plazo especificado y finito”. Consecuentemente, la correctitud de un sistema de tiempo real depende no sólo del resultado lógico de la computación, sino también del tiempo en el que este resultado tarda en generarse.

La computación de tiempo real no es equivalente a la computación rápida. El objetivo de una computación rápida es minimizar el **tiempo de respuesta medio** de un conjunto dado de tareas. En contraste, el objetivo de la computación de tiempo real es garantizar los requisitos temporales individuales de cada tarea. Más que ser rápido, que es un término relativo, la propiedad más importante de un sistema de tiempo real es la predecibilidad.

En los sistemas de tiempo real, el computador actúa generalmente junto con dispositivos físicos. Está dedicado al control y la monitorización de estos equipos y forma parte de un sistema de ingeniería más amplio, bien un horno microondas, bien un sistema de guía de misiles, un automóvil o una central nuclear. Por esta razón, los sistemas de tiempo real de este tipo se denominan **sistemas empotrados**.

Es interesante recalcar que no hay que confundir los sistemas de tiempo real con los sistemas ”on-line” o interactivos, cuya respuesta del sistema debe garantizarse dentro de un plazo tiempo asequible, pero no tiene un tiempo de respuesta crítico. Si se produce un retardo en la respuesta del sistema, se podría llegar a provocar un fallo general y de graves consecuencias en sistemas críticos como el caso de los controladores aéreos o de aplicaciones militares.

2. CARACTERÍSTICAS

Un sistema de tiempo real posee muchas características, bien inherentes, bien impuestas. Por lo tanto, un sistema operativo o un lenguaje que pretenda ser utilizado para construir sistemas de tiempo real debe disponer de facilidades que soporten estas características.

Para mostrar las características, consideraremos un sistema de tiempo real como el que aparece en la figura1 de la página121. En el se pretenden controlar las variables nivel y temperatura del contenido de un depósito. El sistema debe medir los valores del nivel y de la temperatura periódicamente, a traves de unos dispositivos de entrada, y actuar sobre una válvula de control y un elemento calefactor, a través de unos dispositivos de salida, controlando que ambos valores se mantengan próximos a sus valores de referencia.

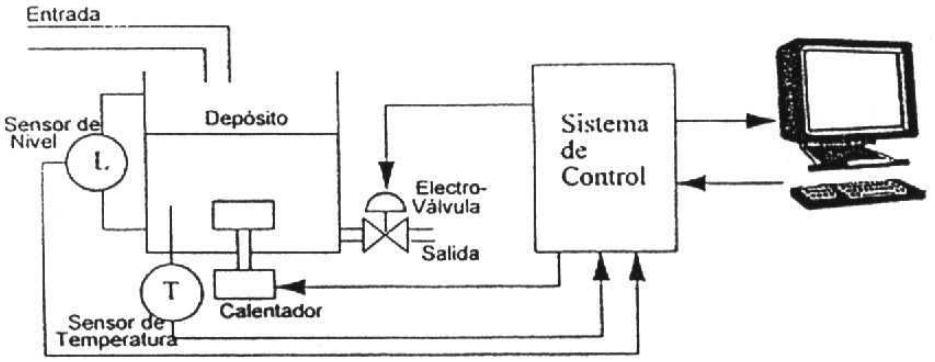


Figura 1: Sistema de control de dos variables

2.1. Control

Las tareas de control constituyen el núcleo en la mayoría de sistemas de tiempo real. Su función es supervisar y actuar sobre un sistema paralelo, que normalmente consiste en un sistema físico ajeno al sistema informático. En el ejemplo de la figura1 de la página121, aparecen las siguientes:

- La tarea de control de nivel: consiste en medir el valor del nivel del tanque cada cierto tiempo (periodo de muestreo), y en consecuencia actuar sobre la valvula de control para mantener el nivel deseado.
- La tarea de control de temperatura: funciona de manera similar, pero trabajando sobre el sensor de temperatura y el calefactor.

2.2. Comunicación

Las tareas de comunicación permiten que otro sistema supervise y programe al propio sistema de tiempo real. En el ejemplo de la figura 1, la tecera tarea consistiría en leer las órdenes dadas por el operador (cambios de valores de referencia, de periodos de muestreo, etc.) y ejecutarlas.

2.3. Concurrencia

Las tareas deben ejecutarse en paralelo, sin esperar a que se termine una de ellas para llevar a cabo las actividades de las demás. En el ej. de la fig.1, es fundamental que las tareas de comunicación puedan ejecutarse paralelamente a las de control. Si no fuera así, una vez puesto el sistema en marcha, el operario no podría introducir cambios en los valores de referencia a menos que se detuviese el proceso de control.

2.4. Dependencia del tiempo

El tiempo de respuesta es primordial en los sistemas de tiempo real, pero desgraciadamente es muy difícil diseñar e implementar sistemas que garanticen todos los plazos en todas las circunstancias posibles. Una aproximación al problema es dotar al sistema de tiempo real de una potencia de cómputo bien sobrada para asegurar que la situación más desfavorable no provoque situaciones de fallo. En el ejemplo de la figura1 de la página121, la dependencia del tiempo se manifiesta de dos formas:

- Mediante la ejecución periódica de tareas de control, las cuales deben activarse con un periodo de tiempo determinado.
- La tarea de atención al operador se ejecuta en respuesta en respuesta a un estímulo externo, la introducción de una orden. El tiempo de respuesta a cada orden tendrá que estar delimitado.

2.5. Fiabilidad y seguridad

Un sistema de este tipo debe asegurar un funcionamiento correcto, al menos en parte, aún en presencia de averías o fallos en algún componente. El mismo tamaño y complejidad de los sistemas de tiempo real exacerban el problema de la fiabilidad. No sólo deben considerarse los fallos inherentes a la ejecución de la aplicación, sino también posibles errores introducidos en su diseño.

En el ejemplo expuesto, debe evitarse el desbordamiento del tanque o la superación de una temperatura determinada, incluso en presencia de fallos informáticos.

2.6. Complejidad de los sistemas

Los sistemas de tiempo real deben responder a eventos del mundo real. La variedad de estos eventos suele conducir a aplicaciones de gran tamaño. Ya que el entorno de una aplicación es contínuamente cambiante, la aplicación, grande o pequeña, debe evolucionar constantemente. El costo de un rediseño y una reescritura constante de una aplicación grande tiene un costo prohibitivo, por lo que los sistemas de tiempo real deben ser extensibles. Los lenguajes de programación deben proporcionar facilidades para dividirlos en partes más pequeñas y manejables.

3. APLICACIONES EN TIEMPO REAL

Los sistemas de tiempo real tienen aplicación en un amplio espectro de actividades:

- Los sistemas de control de procesos o sistemas empotrados: se utilizan desde principios de los años sesenta en el control de procesos industriales.
- Los sistemas de control de fabricación: su tarea consiste en coordinar y controlar un conjunto de dispositivos tales como máquinas-herramienta, robots y cintas transportadoras, reduciendo costes y aumentando la productividad.
- Los sistemas de comunicación, mando y control: Estos sistemas están constituidos por ordenadores conectados en una red de área extensa que actúan vigilando y controlando un sistema físico. Es el caso de los sistemas de monitorización de centrales nucleares o sistemas de control de riego.

Las aplicaciones de tiempo real en el sector del automóvil y de la electrónica de consumo están creciendo muy rápidamente.

4. PROGRAMACIÓN CONCURRENTE

Como en los sistemas de tiempo compartido, en un sistema de tiempo real, el concepto de concurrencia es muy importante. Ya que dos estímulos pueden llegar muy cercanos en el tiempo y ambos exigen ser atendidos en un plazo mínimo, los conceptos de concurrencia y planificación adquieren incluso mayor protagonismo que en los sistemas convencionales de tiempo compartido.

4.1. Concepto

La *programación concurrente* es la denominación que se asigna a la notación y técnicas de programación usadas para expresar el paralelismo y solucionar los problemas de sincronización y comunicación entre tareas que se ejecutan de forma paralela.

4.2. Tareas concurrentes

Un sistema de tiempo real debe ejecutar simultáneamente distintas actividades que atienden a los distintos estímulos del entorno. Cada actividad se traduce en una o más tareas. Una tarea es una secuencia de instrucciones (sea un hilo o un proceso) que se ejecuta en paralelo con otras tareas. La ejecución de tareas se multiplexa en el tiempo en uno o más procesadores.

La figura2 de la página124, se muestra la ejecución concurrente de tres tareas y cómo la tarea 1, en su plazo de ejecución, debe ceder ciclos de procesador a las tareas 2 y 3.

Una tarea puede encontrarse en distintos estados a lo largo de su ejecución, inactiva, cuando ha completado su servicio al evento y un nuevo evento está por llegar, y activa, cuando se encuentra dando servicio al evento.

Una tarea activa puede estar:

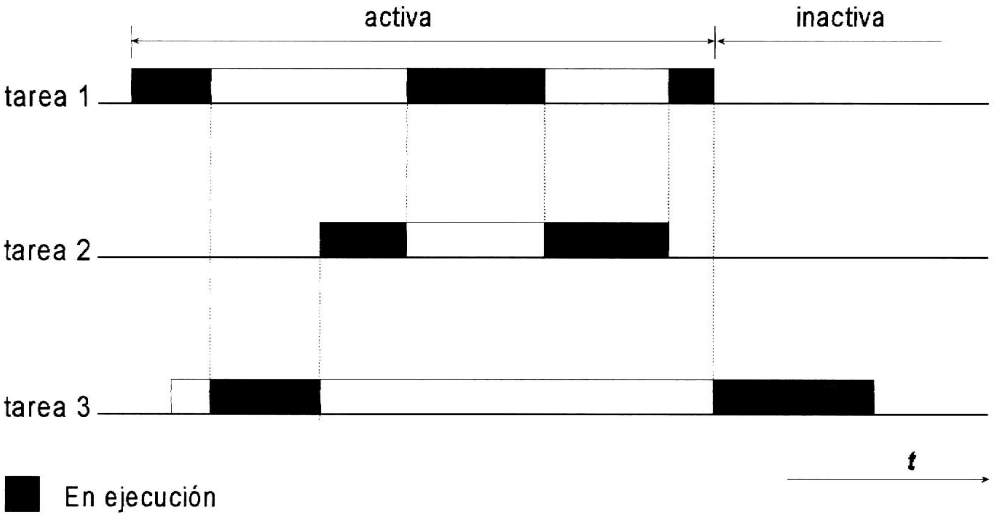


Figura 2: Multiplexación de tareas

- En ejecución
- Preparada para ejecutarse
- Bloqueada esperando alguna condición

4.3. Planificación de tareas

La planificación de la ejecución de tareas concurrentes debe asegurar el cumplimiento de algunas propiedades que no se exigen en los sistemas convencionales de tiempo compartido. Son las siguientes:

1. **Garantía de plazos.**

   Un sistema de tiempo real funciona correctamente cuando los plazos de todas las tareas están garantizados. Esto significa que todas las tareas ejecutan su actividad dentro de plazo cada vez que se activan. En contraste, en un sistema de tiempo compartido, lo importante es asegurar un flujo lo más elevado posible.

2. **Estabilidad.**

   Si a causa de una sobrecarga del sistema no se pueden ejecutar todas las tareas dentro de plazo, se debe garantizar que al menos un subconjunto de tareas críticas cumplen sus plazos. En contraste, en un sistema de tiempo compartido, el criterio es asegurar la equidad en la ejecución de las tareas, que ninguna de ellas se vea indefinidamente postergada.

3. **Tiempo de respuesta máximo.**

En un sistema de tiempo real se trata de acotar el tiempo de respuesta en el peor caso de todas las tareas. En un sistema de tiempo compartido, se trata de conseguir que el tiempo de respuesta medio sea lo más corto posible. La figura2 de la página124, se resume los requisitos anteriores.

Factor	Sistemas de tiempo real	Sistemas de tiempo compartido
Capacidad	Garantía de plazos	Flujo
Reactividad	Tiempo de respuesta máximo	Tiempo de respuesta medio
Sobrecarga	Estabilidad	Equidad

Figura 3: Resumen de requisitos

Para asegurar estas propiedades en los sistemas de tiempo real, hay que utilizar un método de **planificación de tareas** adecuado. La teoría de planificación de tiempo real proporciona algoritmos y métodos de análisis que permiten determinar si en todo momento se garantizan los plazos de respuesta de las tareas activas. Para ayudar en el proceso de planificación vamos a introducir el concepto de prioridad. La prioridad de una tarea indica al planificador el orden en el que las tareas deberían ser ejecutadas.

La asignación de prioridades a las tareas puede ser estática o dinámica. En el primer caso, cada tarea tiene una prioridad fija y en cada momento se ejecuta la tarea activa con mayor prioridad. ¿A qué tareas se asigna la prioridad más alta? Por una parte, podemos asignar la prioridad más alta a las tareas más frecuentes. Este tipo de planificación se denomina *monotónica en frecuencia*. Por otra parte, podemos asignar una prioridad mayor a las tareas más urgentes. Este segundo tipo de planificación se denomina *monotónica en plazo*. Ambos tipos de planificación disponen de herramientas matemáticas que permiten analizar el sistema y comprobar si los plazos están garantizados.

5. INTERRUPCIONES

En los sistemas de tiempo real, la CPU del ordenador debe responder rápidamente a eventos externos sin impedir con ello que dicha unidad se pueda dedicar a otras tareas. Se produce, pues, un problema de sincronización entre la CPU y el mundo exterior.

Por tanto, es necesario un método de sincronización que permita que el ordenador pueda atender al control de proceso durante el tiempo mínimo necesario para que éste se produzca correctamente.

5.1. Consulta de estado

En el método de consulta de estado, también denominado sondeo o muestreo, es la CPU la encargada de la sincronización realizando periódicamente una encuesta a los distintos dispositivos consultando su situación.

Este método de sincronización es de gran simplicidad y flexibilidad, aunque presenta inconvenientes que se agravan con los sistemas de tiempo real:

- El muestreo debe ser sistemático. El programador de be incluirlo en sus programas.
- El muestreo debe ser frecuente.
- La mayoría de las veces se pierde tiempo de la CPU en consultar los dispositivos.
- El problema se agrava cuando hay muchos dispositivos.

5.2. Interrupciones

El uso de interrupciones permite que sean los propios dispositivos los que interrumpan la ejecución del programa en la CPU cuando se produce el evento cuyo control se desea efectuar. Esta interrupción se hace a través de una línea especial del bus de control llamada INTR o IRQ. En muchos casos, tal vez no se pueda aceptar la interrupción, debido a cuestiones de prioridad u otros motivos. En este caso, o bien se deshabilitará el sistema de interrupciones o se recordará la petición de interrupción, a fin de servirla cuando se pueda.

Hay dos tipos de interrupciones:

- Enmascarables: Se pueden atender con posterioridad a su petición.
- No Enmascarables (NMI): Necesitan atención inmediata. Ejemplos podrían ser un típico error de memoria o situaciones de error del sistema. Si se sigue la ejecución del sistema se podrían producir resultados inesperados y efectos no deseados.

Cuando le llega una petición de interrupción a la CPU, se siguen una serie de pasos:

1. Se termina de ejecutar la instrucción en curso.
2. Si la petición es aceptada, se inhibe total o parcialmente el sistema de interrupciones.
3. Se guarda el estado de la tarea en curso y el contador de programa. Se salta a la dirección donde se encuentra la rutina de servicio de la interrupción.

4. Tratamiento de la interrupción.
5. Restitución del estado de la tarea interrumpida.
6. Rehabilitación, en caso necesario, del sistema de interrupciones.
7. Retorno al programa interrumpido.

5.3. Interrupciones vectorizadas

En las interrupciones vectorizadas, los dispositivos que provocan la interrupción han de suministrar la dirección de salto en la que se ubica la rutina de servicio a la interrupción. Las direcciones suelen agruparse en una tabla que recibe el nombre de tabla de vectores de interrupción y lo que los dispositivos facilitan es el número de entrada de la tabla donde se encuentra el vector de interrupción con la dirección de la rutina de servicio. Existen entradas vacías, donde el programador se puede crear sus propias interrupciones.

6. SINCRONIZACIÓN Y COMUNICACIÓN ENTRE TAREAS

En multitud de ocasiones es preciso que las tareas se comuniquen entre sí. Para ello, deberá establecerse un mecanismo de sincornización entre las tareas que pemita una correcta comunicación, evitando en lo posible el uso de interrupciones.

Cuando hay varias tareas que se ejecutan concurrentemente, puede suceder que varias de ellas intenten acceder a los mismos recursos compartidos, produciéndose condiciones de competencia. La solución a estos problemas de acceso a recursos compartidos es sencilla: prohibir que más de un proceso acceda simultáneamente a dichos recursos, es decir, establecer una exclusión mutua. Denominaremos sección crítica a aquella parte del programa en la cual se accede a los recursos compartidos. De este modo, si evitamos que dos procesos entren simultáneamente en sus secciones críticas evitaremos las condiciones de competencia.

Para que dos procesos cooperen paralelamente y usen eficazmente los recursos compartidos se deben dar las siguientes condiciones:

1. Dos o más procesos no deben solaparse en sus secciones críticas.
2. Cada proceso debe permanecer en su sección crítica el mínimo tiempo posible.
3. Ningún proceso en ejecución fuera de su sección crítica puede bloquear a otros procesos.
4. Ningún proceso debe esperar eternamente para entrar en su sección crítica. En lo referente a los métodos de exclusión mutua, se hallan todos explicados

detenidamente en el [TEMA 16 de ”Gestión de Procesos”, y no es el centro de atención de este tema].

7. LENGUAJES DE TIEMPO REAL

Teniendo en cuenta los requisitos típicos de un sistema de tiempo real, es posible determinar qué características debe poseer un lenguaje de programación para facilitar su desarrollo:

- Concurrencia: debe permitir la definición de procesos o tareas concurrentes.
- Control del tiempo: debe existir un reloj de tiempo real accesible desde el lenguaje.
- Planificación de la ejecución de tareas: debe ser posible definir el algoritmo de planificación de tareas.
- Comunicación con dispositivos de hardware: permitir el manejo de los dispositivos de hardware y las interrupciones.
- Control de errores y excepciones: deben existir instrucciones para la detección y recuperación de fallos, tanto del lenguaje como del propio sistema.

7.1. Lenguajes convencionales

En un principio, la programación de aplicaciones de tiempo real consistía en utilizar un lenguaje de programación secuencial (Fortran, Pascal, ...), relegando las construcciones propias de la programación de tiempo real a un sistema operativo de tiempo real. Éste proporcionaba una serie de operaciones para el arranque de tareas concurrentes, sincronización entre estas, medidas de tiempo, etc.

7.2. Modula-2 y Ada

Estos dos lenguajes son los representantes más significativos de la generación actual de lenguajes de programación de sistemas de tiempo real.

Modula-2 es una extensión de Pascal. Dispone de mecanismos de descomposición modular, concurrencia, control de interrupciones y acceso a dispositivos hardware. El tratamiento de la concurrencia es muy flexible, permitiendo construir distintos esquemas de ejecución y de sincronización de procesos concurrentes.

Ada dispone de un mecanismo de descomposición modular similar al de Modula-2, con la posibilidad adicional de definir módulos genéricos. La concurrencia está integrada en el lenguaje a través del concepto de tareas o procesos concurrentes. La sincronización de tareas sigue el esquema denominado ”rendezvous”. Dispone de un mecanismo flexible para el tratamiento de errores y excepciones.

7.3. Eiffel y Java

Ambos lenguajes forman parte de un novedoso enfoque de la concurrencia denominado *programación concurrente orientada a objetos simplificada* (SCOOP).

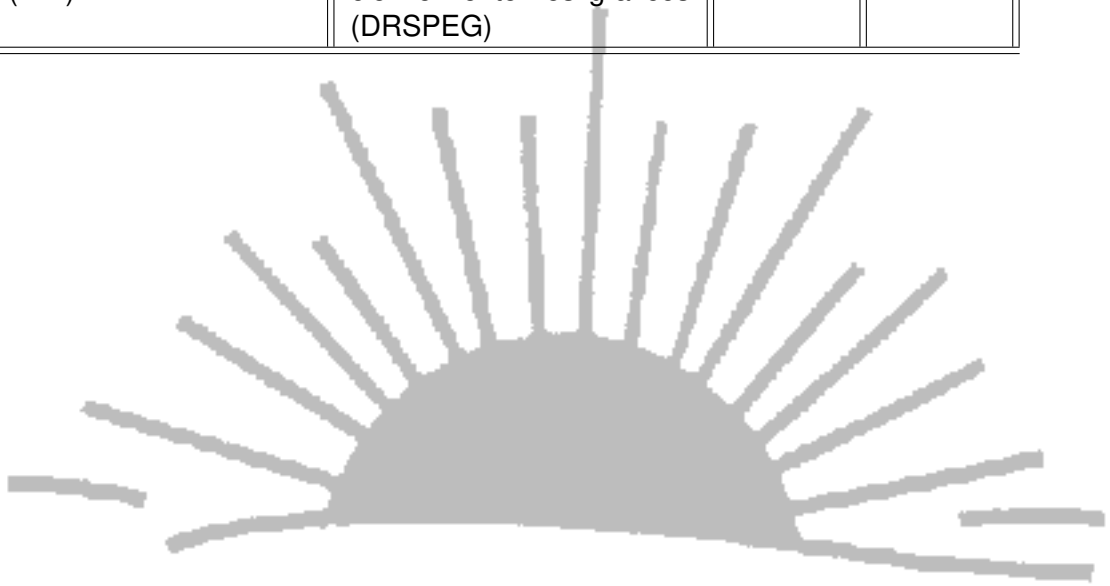
El lenguaje Eiffel soluciona todos los problemas típicos de sincronización y comunicación con una única palabra reservada, separate, que se añade a la notación secuencial orientada a objetos.

Por su parte, Java implementa el esquema clásico de concurrencia. Este lenguaje utiliza la definición explícita de tareas paralelizables a través de la clase Thread y el interfaz Runnable. Al igual que los procesos clásicos, los threads pueden estar en varios estados. Java incorpora mecanismos de sincronización entre tareas concurrentes. Para ello, utiliza la palabra reservada syncronized que no es nada más que un semáforo binario.

Cabe tener en cuenta, que el comportamiento que puede esperar el programador del soporte de tareas concurrentes en Java no permite la realización de aplicaciones con fuertes requisitos temporales. No es posible tan siquiera cambiar la política de planificación de la máquina virtual, y lo que es peor, ésta depende de la plataforma donde resida el intérprete. Por estos motivos, la utilización de multithreading en Java debe limitarse simplemente a situaciones donde sea conveniente tener varias tareas en paralelo, sin que importe demasiado la precisión con que se reparten el procesador o la frecuencia con que se realizan cambios de contexto.

8. CONTEXTUALIZACION

CICLO FORMATIVO	MODULO	CURSO	CUERPO
Administración de sistemas informáticos (ASI)	Fundamentos de Programación (FPR)	1	PS
Desarrollo de Aplicaciones informáticas (DAI)	Programación en Lenguajes Estructurados (PLE)	1	PS
Desarrollo de Aplicaciones informáticas (DAI)	Diseño y realización de servicios de presentación en entornos gráficos (DRSPEG)	2	PT



BIBLIOGRAFIA

[PRE02] ROGUER. S. PRESSMAN. *Ingeniería del Software. Un enfoque práctico.* McGrawHill, 5 edition, 2002. ISBN: 8448132149.

[PRI97] ALBERTO PRIETO. *Introducción a la Informática*. Mcgraw-hill, 2 edition, 1997.

[UL97] ALFONSO UREÑA LOPEZ. *Fundamentos de Informática*. Ra-ma, 1997.