CAPÍTULO 1

**Introducción a los sistemas**

**informáticos**

Un sistema operativo (SO) explota los recursos de hardware de uno o más procesadores para ofrecer un conjunto de servicios a los usuarios del sistema. El sistema operativo también gestiona la memoria secundaria y los dispositivos de entrada/salida (E/S) en nombre de los usuarios. Así pues, es conveniente disponer de una cierta comprensión del hardware del sis- tema informático subyacente antes de comenzar el estudio de los sistemas operativos.

Este capítulo ofrece una visión de conjunto del hardware de los sistemas informáticos. Este resumen es extremadamente breve en la mayoría de los campos, pues se asume que el lector está familiarizado con el tema. Sin embargo, algunas áreas se abordan con un cierto detalle, por su importancia en los asuntos que se tratarán más adelante en el libro.

1.1

**ELEMENTOS BÁSICOS**

En un alto nivel, un sistema informático consta de procesador, memoria y componentes de E/S, con uno o más módulos de cada tipo. Estas componentes están interconectados de al- guna forma para llevar a cabo la función principal del computador, que es ejecutar progra- mas. Así pues, se tienen cuatro elementos principales:

•*Procesador:* Controla la operación del computador y lleva a cabo las funciones de pro- cesamiento de datos. Cuando hay un solo procesador, se suele denominar *unidad cen- tral de procesamiento (CPU, Central Processing Unit).*

•*Memoria Principal:* Almacena los datos y los programas. Esta memoria es normal- mente volátil; también se le conoce como *memoria real o memoria primaria.*

•*Interconexión de sistemas:* Ciertos mecanismos y estructuras que permiten la comuni- cación entre procesadores, memoria principal y los módulos de E/S.

**1**

*Digitalización con propósito académico*

*Sistemas Operativos*

**2 Introducción a los sistemas informáticos**

La figura 1.1 ilustra estos componentes de alto nivel. El procesador es normalmente quien lleva el control. Una de sus funciones es intercambiar los datos con la memoria. Para este propósito, hace uso de dos registros internos (al procesador): un registro de direcciones de memoria (MAR, *Memory Address Register),* el cual especifica la dirección en memoria de la próxima lectura o escritura y un registro intermedio *(buffer)* de memoria (MBR, *Memory Buffer Register),* que contiene los datos que van a ser escritos a memoria o que fueron leídos de la misma. De manera similar, un registro de direcciones de E/S (IOAR, *InputlOutput Ad-dress Register)* especifica un dispositivo particular de E/S. Un registro intermedio de E/S (IOBR, *InputtOutput Buffer Register)* se utiliza para intercambiar datos entre un módulo de E/S y el procesador.

Un módulo de memoria consta de un conjunto de ubicaciones definidas por direcciones enumeradas secuencialmente. Cada ubicación contiene un número binario que puede ser in- terpretado como una instrucción o como un dato. Un módulo de E/S transfiere datos desde los dispositivos externos hacia la memoria y el procesador y viceversa. Este contiene buffers internos para almacenar temporalmente los datos hasta que puedan ser enviados.

1.2

**REGISTROS DEL PROCESADOR**

Dentro del procesador, hay un conjunto de registros que ofrecen un nivel de memoria que es más rápido y pequeño que la memoria principal. Los registros del procesador sirven para dos funciones:

*Digitalización con propósito académico*

*Sistemas Operativos*

**Registros del procesador 3**

• *Registros visibles de usuario:* Un programador de lenguaje de máquina o ensamblador puede minimizar las referencias a memoria principal mediante un uso óptimo de estos registros. Con lenguajes de alto nivel, un compilador que optimice código intentará ha- cer una selección inteligente de qué variables asignar a registros y cuáles a ubicaciones de la memoria principal. Algunos lenguajes de alto nivel, como C, permiten que el pro-gramador indique al compilador qué variables se deben almacenar en registros.

• *Registros de control y de estado:* Son utilizados por el procesador para el control de las operaciones o por rutinas privilegiadas del sistema operativo para controlar la ejecución de los programas.

No hay una separación clara de los registros en estas dos categorías. Por ejemplo, en al-gunas máquinas el contador de programa es visible para los usuarios, pero en otras muchas no lo es. Sin embargo, para el propósito de la discusión que viene a continuación, es conve-niente emplear estas categorías.

**Registros visibles de usuario**

Un registro visible de usuario es aquél que puede ser referenciado por medio del lenguaje de máquina que ejecuta el procesador y es, por lo general, accesible para todos los programas, incluyendo tanto los programas de aplicación como los del sistema. Las clases de registro que, normalmente, están disponibles, son los registros de datos, los registros de dirección y los registros de códigos de condición.

Los **registros de datos** pueden ser asignados por el programador a diversas funciones. En algunos casos, son de propósito general y pueden ser empleados por cualquier instrucción de máquina que lleve a cabo operaciones sobre los datos. Sin embargo, suelen ponerse cier-tas restricciones a menudo. Por ejemplo, pueden haber registros dedicados a operaciones en coma flotante.

Los **registros de dirección** contienen direcciones en la memoria principal de datos e ins-trucciones o una parte de la dirección que se utiliza en el cálculo de la dirección completa. Estos registros pueden ser de propósito general o pueden estar dedicados a un modo especí-fico de direccionamiento. Entre los ejemplos se incluyen:

• *Registro índice:* El direccionamiento indexado es un modo común de direccionamiento que implica sumar un índice a un valor base para obtener la dirección efectiva.

• *Punteroí de segmento:* Con direccionamiento segmentado, la memoria se divide en segmentos, que son bloques de palabras de tamaño variable. Una referencia a memoria consta de una referencia a un segmento particular y un desplazamiento dentro del seg- mento; este modo de direccionamiento es importante en la discusión sobre la gestión de memoria de los capítulos 6 y 7. En este modo, se utiliza un registro que alberga una dirección base (ubicación inicial) de un segmento. Puede haber varios registros de este tipo: por ejemplo, uno para el sistema operativo (es decir, cuando se ejecuta código del sistema operativo en el procesador) y otro para la aplicación que está en ejecución.

• *Puntero de pila:* Si hay un direccionamiento de pila visible para los usuarios, la pila es-tará, por lo general, en la memoria principal, existiendo un registro dedicado a señalar

El término inglés *pointer* suele traducirse en Latinoamérica por el término apuntador (N. del. T.)

*Digitalización con propósito académico*

*Sistemas Operativos*

**4 Introducción a los sistemas informáticos**

la cima de la pila. Esto permite el uso de instrucciones que no contienen ningún campo de dirección, tales como *push* (poner) y *pop* (sacar). (Consúltese el Apéndice IB para una discusión sobre el tratamiento de pilas).

Una última categoría de registros que son, al menos, parcialmente visibles para los usua-rios, son aquellos que contienen **códigos de condición** (también denominados *indicadores o flags).* Los códigos de condición son bits activados por el hardware del procesador como re-sultado de determinadas operaciones. Por ejemplo, una operación aritmética puede producir un resultado positivo, negativo, cero o desbordamiento. Además de almacenar el resultado de esta operación en un registro o en memoria, también se activará un código de condición. Este código puede consultarse posteriormente como parte de una operación de salto condicional.

Los bits de código de condición se agrupan en uno o más registros. Estos forman general-mente parte de un registro de control. En general, las instrucciones de máquina permiten leer estos bits mediante referencias implícitas, pero no pueden ser alterados por el programador.

En algunas máquinas, una llamada a un procedimiento o subrutina provocará que los re-gistros visibles de usuario se salven automáticamente, para luego restaurarlos al retomar. Este proceso de salvar y restaurar lo lleva a cabo el procesador como parte de la ejecución de las instrucciones de llamada y retomo. Esto permite que cada procedimiento pueda usar los registros de forma independiente. En otras máquinas, es responsabilidad del programa- dor salvar los contenidos de los registros de usuario visibles que sean relevantes antes de ha-cer la llamada a un procedimiento, incluyendo instrucciones en el programa con tal propó-sito. Así pues, las instrucciones de salvar y restaurar pueden ser llevadas a cabo por el hardware o por el software, dependiendo de la máquina.

**Registros de control y de estado**

Varios registros se emplean para controlar las operaciones del procesador. En la mayoría de las máquinas, la mayor parte de estos registros no son visibles para los usuarios. Algunos de ellos pueden estar accesibles a las instrucciones de máquina ejecutadas en un modo de control o modo del sistema.

Por supuesto, máquinas diferentes tendrán organizaciones diferentes de registros y podrán usar terminologías distintas. Aquí se da una lista bastante completa de tipos de registros, incluyendo una breve descripción de las mismas. Además de los registros **MAR, MBR, IOAR y IOBR** mencionados anteriormente, los siguientes registros son esenciales en la ejecución de instrucciones;

• *Contador de programa (PC, Program Counter):* Contiene la dirección de la instrucción a ser leída.

• *Registro de instrucción (IR, Instruction Regíster):* Contiene la última instrucción leída.

Todos los diseños de procesadores incluyen además un registro o conjunto de registros, conocidos a menudo como palabra de estado del programa (PSW, *Program Status Word),* que contiene información de estado. Normalmente, la PSW contiene códigos de condición junto a otra información de estado. Entre los campos e indicadores más comunes se incluyen los siguientes:

• *Signo:* Contiene el bit del signo de la última operación aritmética efectuada.

• *Cero:* Se activa cuando el resultado de una operación aritmética es cero.

• *Acarreo:* Se activa cuando, como resultado de una suma o una resta, se produce un acarreo más allá del bit más significativo. Se utiliza en operaciones aritméticas de más de una palabra.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Ejecución de instrucciones 5**

*• Igualdad:* Se activa si una comparación lógica da como resultado la igualdad.

*• Desbordamiento:* Empleado para seflalar un desbordamiento aritmético.

*• Habilitar/inhahilitar interrupciónes:* Empleado para habilitar o inhabilitar interrupcio-nes. Cuando Las interrupciones están inhabilitadas, el procesador las ignora. Esto es muy deseable cuando el sistema operativo está ocupado en el tratamiento deotra interrupción.

*• Supervisor:* Indica si el procesador está ejecutando en modo supervisor o en modo usuarlo. Ciertas instrucciones privilegiadas sólo se pueden ejecutar en modo supervisor y sólo se puede tener acceso a ciertas áreas de memoria en modo supervisor.

En el diseño de un procesador específico, se pueden encontrar una serie de registros relacionados con el estado y el control. Además de la PSW, puede haber un puntero a un bloque de memoria que contenga información de estado adicional. En máquinas que utilizan varios tipos de interrupción, se puede ofrecer una serie de registros con punteros a cada rutina de tratamiento de interrupción. Si se utiliza una pila para implementar ciertas funciones (por ejemplo, las llamadas a procedimientos), entonces se necesita un puntero a la pila (ver Apéndice 1B). El hardware para la gestión de memoria que se discute en los capítulos 6 y 7 necesitará registros dedicados. Por último, los registros también pueden utilizarse para el control de las operaciones de E/S.

Una serie de factores inciden en el diseño de la organización de los registros de control y estado. Un punto clave es el soporte del sistema operativo. Cierto tipo de información de control es de utilidad específica para el sistema operativo. Si el diseñador del procesador dispone de una visión funcional del sistema operativo, la organización de los registros puede adaptarse convenientemente.

Otra decisión clave del diseño es la asignación de información de control a los registros y la memoria. Es habitual dedicar los primeros centenares o miles de palabras (las más bajas) de memoria para el control. EL diseñador debe decidir la cantidad de información de control que debe residir en los rápidos y costosos registros, junto a la cantidad que debe permanecer en memoria principal, que es más lenta y barata.

1 .3

**EJECUCIÓN DE INSTRUCCIONES**

La tarea básica que realiza un computador es la ejecución de los programas. El programa a ejecutar consta de un conjunto de instrucciones almacenadas en memoria. El procesador lleva a cabo el trabajo, ejecutando las instrucciones especificadas en el programa.

Para alcanzar una mayor comprensión de esta función y de la manera en que los componentes principales del computador interactúan para ejecutar un programa, hace falta analizar con cierto detalle los elementos de la ejecución de un programa. EL punto de vista más sencillo es considerar que el procesamiento de instrucciones consta de dos pasos. El procesador (1) trae las instrucciones desde la memoria, una cada vez y (2) ejecuta cada instrucción. La ejecución de un programa consiste en la repetición de este proceso de lectura y ejecución de la instrucción. La ejecución de la instrucción puede involucrar varias operaciones y depende de la naturaleza de la instrucción.

El procesamiento requerido para una instrucción simple se llama *ciclo de instrucción.* El ciclo de instrucción se representa en la figura 1.2, empleándose esta descripción simplifi**-**

*Digitalización con propósito académico*

*Sistemas Operativos*

**6 Introducción a los sistemas informáticos**

cada de dos pasos que se acaba de explicar. Los dos pasos se llaman *ciclo de lectura (fetch) y ciclo de ejecución.* La ejecución del programa se detiene sólo si se apaga la máquina, ocu- rre algún tipo de error irrecuperable o se encuentra una instrucción en el programa que de- tiene el computador.

**Lectura y ejecución de instrucciones**

Al comienzo de cada ciclo de instrucción, el procesador lee una instrucción de la memoria. En un procesador típico habrá un registro llamado *contador de programa* (PC), que se usa para llevar la cuenta de cuál es la próxima instrucción a leer. A menos que se diga otra cosa, el procesador siempre incrementará el PC después de leer cada instrucción, de forma que después se lea la instrucción siguiente en la secuencia (es decir, la instrucción ubicada en la dirección inmediatamente superior de la memoria). Así pues, considérese por ejemplo un computador en la que cada instrucción ocupa una palabra de memoria de 16 bits. Supóngase que el contador de programa apunta a la ubicación 300. La próxima instrucción que va a leer el procesador es la que está en la ubicación 300. En los siguientes ciclos de instrucción, le- erá las instrucciones de las ubicaciones 301, 302, 303 y así sucesivamente. Como se verá en breve, esta secuencia puede alterarse.

La instrucción leída se carga en un registro del procesador conocido como *registro de instrucción* (IR). La instrucción está en forma de código binario que especifica cuál es la acción que el procesador llevará a cabo. El procesador interpreta la instrucción y realiza la acción requerida. En general, estas acciones pueden clasificarse en las siguientes cuatro categorías:

• *Procesador-memoria:* Se transfieren datos del procesador a la memoria o viceversa.

• *Procesador-EIS:* Se transfieren datos desde o hacia un dispositivo periférico, realizán- dose la transferencia entre el procesador y un módulo de E/S.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Ejecución de instrucciones 7**

• *Tratamiento de datos:* El procesador realiza alguna operación aritmética o lógica sobre los datos,

• *Control:* La instrucción pide que se altere la secuencia de ejecución. Por ejemplo, el procesador puede leer una instrucción de la ubicación 149, la cual especifica que la pró-xima instrucción sea la de la ubicación 182. El procesador recordará este hecho ajus-tando el valor del contador de programa a 182. De este modo, en el próximo ciclo de lectura, se traerá la instrucción de la ubicación 182 y no de la 150,

Por supuesto que la ejecución de una instrucción puede incluir una combinación de estas acciones.

Como ejemplo sencillo, se considera una máquina hipotética que incluya las características enumeradas en la figura 1.3. El procesador contiene un único registro de datos, llamado *acumulador* (AC). Tanto las instrucciones como los datos son de 16 bits de longitud. Así pues, es conveniente organizar la memoria utilizando ubicaciones o palabras de 16 bits. El formato de instrucción dedica cuatro bits para el código de la operación (cod-op). Por tanto, pueden haber hasta 24 = 16 códigos de operación diferentes y hasta 212 = 4096 (4K) palabras de memoria que se pueden direccionar directamente.

La figura 1.4 ilustra la ejecución parcial de un programa, mostrando las zonas pertinentes de la memoria y los registros del procesador. El fragmento de programa que se muestra suma el contenido de la palabra de memoria de la dirección 940 al contenido de la palabra de memoria de la dirección 941 y almacena el resultado en esta última dirección. Se requieren tres instrucciones, que se pueden describir con tres ciclos de lectura y tres de ejecución:

1. El PC contiene 300, la dirección de la primera instrucción. Se carga el contenido de la ubicación 300 en el IR. Nótese que este proceso podría involucrar el uso de un MAR y un MBR. Por simplicidad, se van a ignorar estos registros intermedios.

*Digitalización con propósito académico*

*Sistemas Operativos*

**8 Introducción a los sistemas informáticos**

2. Los primeros 4 bits del IR indican que se cargará el AC. Los 12 bits restantes especifi-can la dirección, que es 940.

3. Se incrementa el PC y se lee la instrucción siguiente,

4. El contenido anterior del AC y el contenido de la ubicación 941 se suman y el resul-tado se almacena en el AC.

5. Se incrementa el PC y se lee la instrucción siguiente.

6. El contenido *del AC se almacena* en *la ubicación 941.*

En este ejemplo se necesitan tres ciclos de instrucción, donde cada uno consta de un ciclo de lectura y otro de ejecución, para sumar el contenido de la ubicación 940 al contenido de la ubicación 941. Con un conjunto de instrucciones más complejo harían falta menos ciclos. La mayoría de los procesadores actuales aportan instrucciones que incluyen más de una dirección. De esta manera, en el

*Digitalización con propósito académico*

*Sistemas Operativos*

**Interrupciones 9**

ciclo de ejecución de una instrucción particular pueden participar más de una referencia a memoria. Además, en vez de referencias a memoria, una instrucción puede especificar una operación de E/S.

**Funciones de E/S**

Hasta ahora se ha discutido que el funcionamiento del computador es controlado por el procesador y se ha analizado en primer lugar la interacción entre el procesador y la memoria. En la discusión sólo se ha aludido al papel de la componente de E/S.

Los módulos de E/S (un controlador de disco, por ejemplo) pueden intercambiar datos di-rectamente con el procesador. Al igual que el procesador puede iniciar una lectura o escritura en la memoria, indicando la dirección de una ubicación específica, el procesador también puede leer datos de un módulo de E/S o escribir datos en el módulo. En este último caso, el procesador identifica a un dispositivo específico que es controlado por un módulo de E/S determinado. De este modo se podría tener una secuencia de instrucciones similar a la de la figura 1.4, pero con instrucciones de E/S en lugar de referencias a memoria.

En algunos casos, es conveniente permitir que los intercambios de E/S se produzcan directamente con la memoria. En tal caso, el procesador dará autoridad a un módulo de E/S para leer o escribir en memoria, de modo que la transferencia de E/S ocurre sin obstruir al procesador. Durante la transferencia, el módulo de E/S emite órdenes de lectura o escritura en la memoria, librando de responsabilidades al procesador en el intercambio. Esta operación se conoce como *acceso directo a memoria* (DMA, *Direct Memory Access)* y será examinada más adelante en este mismo capítulo. Por ahora, todo lo que se necesita saber es que la estructura de interconexión del computador puede que tenga que permitir una interacción directa entre la memoria y la E/S.

1.4

**INTERRUPCIONES**

Casi todos los computadores tienen un mecanismo mediante el cual otros módulos (E/S, memoria) pueden interrumpir la ejecución normal del procesador. La tabla 1.1 enumera las clases más comunes de interrupciones.

Las interrupciones aparecen, principalmente, como una vía para mejorar la eficiencia del procesamiento. Por ejemplo, la mayoría de los dispositivos externos son mucho más lentos

**TABLA 1.1 Clases de Interrupciones**

De programa Generadas por alguna condición que se produce como resultado de la ejecución de una instrucción, como el desbordamiento aritmético, la división por cero, el intento de ejecutar una instrucción ilegal de la máquina o una referencia a una zona de memoria fuera del espacio permitido al usuario.

De reloj Generadas por un reloj interno del procesador. Esto permite al sistema operativo llevar a cabo ciertas funciones con determinada regularidad.

De E/S Generadas por un controlador de E/S, para indicar que una operación ha terminado normalmente o para indicar diversas condiciones de error.

Por fallo del hardware Generadas por fallos tales como un corte de energía o un error de paridad de la memoria.

*Digitalización con propósito académico*

*Sistemas Operativos*

**10 Introducción a los sistemas informáticos**

que el procesador. Supóngase que el procesador está transfiriendo datos hacia una impre- sora, utilizando un esquema para el ciclo de instrucción como el de la figura 1.2. Después de cada operación ESCRIBIR, el procesador hará una pausa y permanecerá desocupado hasta que la impresora se ponga al corriente. La duración de esta pausa puede ser del orden de va- rios cientos o incluso miles de ciclos de instrucción en los que la memoria no está implicada. Está claro que esto es un derroche en la utilización del procesador.

La figura 1 -5 a ilustra este estado de las cosas para la aplicación que se indica en el párrafo anterior. El programa de usuario lleva a cabo una serie de llamadas a ESCRIBIR, intercala- das con el procesamiento. Los segmentos de código 1, 2 y 3 se refieren a secuencias de ins- trucciones que no implican E/S. Las llamadas a ESCRIBIR son, en realidad, llamadas a un programa de E/S, que es una utilidad del sistema que llevará a cabo la operación concreta de E/S. El programa de E/S consta de tres secciones:

• Una secuencia de instrucciones, etiquetada con un 4 en la figura, de preparación para la operación concreta de E/S, Esto puede incluir la copia de los datos de salida hacia un buffer especial y preparar los parámetros de la orden del dispositivo.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Interrupciones 11**

• La orden concreta de E/S. Sin el uso de interrupciones, una vez que se emita esta orden, el programa debe esperar a que el dispositivo de E/S lleve a cabo la función pedida. El programa puede esperar simplemente ejecutando de forma repetida una operación que compruebe si ya se realizó la E/S.

• Una secuencia de instrucciones, etiquetada con Un *5* en La figura, para completar la operación. Esto puede incluir la activación de un código de condición que indique el éxito o el fracaso de la operación.

Debido a que la operación de E/S puede tardar un tiempo relativamente grande en terminar, el programa de E/S puede quedar colgado esperando a que se complete la operación; así pues, el programa de usuario se detendrá por un tiempo considerable en el momento de la llamada a ESCRIBIR.

**Las interrupciones y el ciclo de instrucción**

Con las interrupciones, el procesador se puede dedicar a la ejecución de otras instrucciones mientras una operación de E/S está en proceso. Considérese el flujo de control de la figura 1 .5b. Al igual que antes, el programa de usuario alcanza un punto en el que hace una llamada al sistema en forma de una llamada ESCRIBIR. El programa de E/S que se invoca consta solo del código de preparación y de la orden concreta de E/S. Después de que se ejecuten estas pocas instrucciones, se devuelve el control al programa de usuario. Mientras tanto, el dispositivo externo estará ocupado recibiendo datos desde la memoria del computador e imprimiéndolos. Esta operación de E/S se lleva a cabo concurrentemente con la ejecución de las instrucciones del programa de usuario.

Cuando el dispositivo de E/S esté disponible, es decir, cuando esté preparado para aceptar más datos desde el procesador, el módulo de E/S de dicho dispositivo enviará una señal de *solicitud de interrupción* al procesador. El procesador responde suspendiendo la operación del programa en curso y saltando a un programa que da servicio al dispositivo de E/S en par-ticular, conocido como *rutina de tratamiento de la interrupción (interrupt handler),* reanu-dando la ejecución original después de haber atendido al dispositivo. En La figura 1 *.5b,* el instante en que se produce tal interrupción viene indicado con un asterisco **(\*).**

Desde el punto de vista del programa de usuario, una interrupción es solamente eso: una interrupción de la secuencia normal de ejecución. Cuando el tratamiento de la interrupción se termina, la ejecución continúa (figura 1.6). Así pues, el programa de usuario no tiene que disponer de ningún código especial para dar cabida a las interrupciones; el procesador y el sistema operativo son los responsables de suspender el programa de usuario y reanudarlo después en el mismo punto.

Para dar cabida a las interrupciones, se añade un *ciclo de interrupción* al ciclo de instrucción, como se muestra en la figura 1.7. En el ciclo de interrupción, el procesador comprueba si ha ocurrido alguna interrupción, lo que se indicará con la presencia de una señal de interrupción. Si no hay interrupciones pendientes, el procesador sigue con el ciclo de lectura y trae la próxima instrucción del programa en curso. Si hay una interrupción pendiente, el procesador suspende la ejecución del programa en curso y ejecuta una rutina de *tratamiento de la interrupción.*

La rutina de tratamiento de la interrupción forma parte generalmente del sistema operativo. Normalmente este programa determina la naturaleza de la interrupción y realiza cuantas acciones sean necesarias. De hecho, en el ejemplo que se ha estado siguiendo, la rutina de tratamiento determina el módulo de E/S que generó la interrupción y puede saltar a un programa que escribirá más datos a dicho módulo. Cuando termina la rutina de tratamiento de la interrupción, el procesador puede reanudar la ejecución del programa de usuario en el punto en que sucedió la interrupción.

*Digitalización con propósito académico*

*Sistemas Operativos*

**12 Introducción a los sistemas informáticos**

*Digitalización con propósito académico*

*Sistemas Operativos*

**Interrupciones 13**

Está claro que hay cierta sobrecarga en este proceso. Se deben ejecutar instrucciones ex- tra (en la rutina de tratamiento de la interrupción) para determinar la naturaleza de la inte- rrupción y decidir la acción apropiada. Sin embargo, por la cantidad de tiempo que se des- perdicia esperando en una operación de E/S, puede aprovecharse el procesador de una manera mucho más eficaz con el uso de interrupciones.

Para apreciar el aumento en la eficiencia, considérese la figura 1.8, que es un diagrama de tiempos basado en el flujo de control de las figuras 1.5a y 1.5b.

En las figuras 1.5b y 1.8 se supone que el tiempo exigido para la operación de E/S es re- lativamente pequeño: menos que el tiempo para completar la ejecución de las instrucciones situadas entre dos operaciones ESCRIBIR del programa de usuario. El caso más normal, es- pecialmente para un dispositivo lento, como es el caso de una impresora, es aquél en que la operación de E/S tardará mucho más tiempo que la ejecución de una secuencia de instruc-

*Digitalización con propósito académico*

*Sistemas Operativos*

**14 Introducción a los sistemas informáticos**

clones del usuario. La figura 1.5c muestra esta situación. En este caso, el programa de usua- rio alcanza la segunda llamada a ESCRIBIR antes de que se complete la operación de E/S desencadenada por la primera llamada. El resultado es que el programa de usuario se sus- pende en este punto. Cuando termine la operación anterior de E/S, se podrá procesar esta nueva llamada a ESCRIBIR y así comenzar una nueva operación de E/S. La figura 1.9 muestra el diagrama de tiempos para esta situación, con y sin uso de interrupciones. Se

**Figura 1.9 Diagramas de tiempo de un programa; espera larga de E/S**

*Digitalización con propósito académico*

*Sistemas Operativos*

**Interrupciones 15**

puede ver que así se produce un aumento de la eficiencia, pues parte del tiempo en que la operación de E/S está en marcha se solapa con la ejecución de las instrucciones del usuario.

**Tratamiento de interrupciones**

El acontecimiento de una interrupción desencadena una serie de sucesos, tanto en el hard- ware del procesador como en el software. La figura 1.10 muestra una secuencia típica. Cuando un dispositivo de E/S completa una operación de E/S, se produce en el hardware la siguiente secuencia de sucesos:

1. El dispositivo emite una señal de interrupción al procesador.

2. El procesador finaliza la ejecución de la instrucción en curso antes de responder a la interrupción, tal como se indica en la figura 1.7.

3. El procesador pregunta por la interrupción, comprueba que hay una y envía una señal de reconocimiento al dispositivo que generó la interrupción. Este reconocimiento le permite al dispositivo suprimir la señal de interrupción.

4. El procesador necesita ahora prepararse para transferir el control a la rutina de inte- rrupción. Para empezar, hace falta salvar la información necesaria para reanudar la eje- cución del programa en curso en el punto de la interrupción. La mínima información

**Figura 1.10 Tratamiento de una interrupción simple**

*Digitalización con propósito académico*

*Sistemas Operativos*

**16 Introducción a los sistemas informáticos**

requerida es la palabra de estado del programa (PSW) y la ubicación de la próxima ins-trucción a ejecutar, que se almacena en el contador de programa. Estos pueden meterse en la pila de control del sistema (ver el Apéndice IB).

5. El procesador carga ahora el contador de programa con la ubicación de entrada del programa de tratamiento de la interrupción. Dependiendo de la arquitectura del com- putador y del diseño del sistema operativo, puede haber un solo programa por cada tipo de interrupción, o uno por cada dispositivo y por cada tipo de interrupción. Si hay más de una rutina de tratamiento de interrupción, el procesador debe determinar a cuál invocar. Esta información pudiera estar incluida en la señal original de la interrupción, o el procesador debe preguntarle al dispositivo que creó la interrupción para obtener respuesta sobre la información que necesita.

Una vez que se ha cargado el contador de programa, el procesador procede con el próximo ciclo de instrucción, que comienza trayendo la próxima instrucción. Debido a que esta instrucción se determina por el contenido del contador de programa, el resultado es que el control se le transfiere al programa que trata la interrupción. La ejecución de este programa se traduce en las operaciones siguientes:

6. En este punto, el contador de programa y la PSW relativa al programa interrumpido han sido salvadas en la pila del sistema. Sin embargo, hay más información que es considerada parte del "estado" de ejecución del programa. En particular se necesita salvar el contenido de los registros del procesador ya que estos registros pudieran ser utilizados por la rutina de tratamiento de la interrupción. Así pues es necesario salvar todos estos valores más cualquier otra información sobre el estado. Normalmente la rutina de tratamiento de la interrupción comienza salvando en la pila el contenido de todos los registros. En el capítulo 3 se discute sobre otra información del estado que también puede que tenga que salvarse. La figura l.lla muestra un ejemplo simple. En este caso, se muestra un programa de usuario que es interrumpido después de la instrucción de la ubicación N. El contenido de todos los registros más la dirección de la próxima instrucción (N+l) son guardados en la pila. El puntero a la pila se actualiza para que apunte a la nueva cima y el contador de programa se actualiza para que apunte al comienzo de la rutina de servicio de la interrupción.

7. La rutina de tratamiento de la interrupción puede ahora proceder a procesar la inte-rrupción. Esto incluirá un examen del estado de la información relativa a la operación de E/S o a cualquier otro evento que haya causado la interrupción. Esto puede también involucrar el envío adicional de órdenes o reconocimientos al dispositivo de E/S.

8. Cuando se completa el tratamiento de la interrupción, se recuperan de la pila los valores de los registros que se salvaron y se restauran los registros (ver por ejemplo la figura 1.1 Ib).

9. El acto final es restaurar los valores de la PSW y del contador de programa a partir de la pila. Como resultado, la próxima instrucción a ser ejecutada será del programa inte-rrumpido previamente.

Es importante salvar toda la información sobre el estado del programa interrumpido para su reanudación posterior, porque la rutina de tratamiento de la interrupción no es una rutina llamada desde el programa. Por el contrario, la interrupción puede producirse en cualquier momento y por tanto en cualquier punto de la ejecución de un programa de usuario. Su ocurrencia es impredecible. Más aún, como se verá más adelante en esta sección, los dos programas pueden no tener nada en común y pueden pertenecer a dos usuarios diferentes.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Interrupciones 17**

**FIGURA 1.11 cambios en la memoria y en los registros durante una interrupción**

*Digitalización con propósito académico*

*Sistemas Operativos*

**18 Introducción a los sistemas informáticos**

**Interrupciones múltiples**

En la discusión anterior se ha analizado el acontecimiento de una sola interrupción. Sin em-bargo, supóngase que pueden producirse múltiples interrupciones. Por ejemplo, un programa puede estar recibiendo datos de una línea de comunicaciones e imprimiendo resultados. La impresora generará una interrupción cada vez que se completa una operación de impresión. El controlador de la línea de comunicaciones generará una interrupción cada vez que llega una unidad de datos. Esta unidad puede ser un solo carácter o un bloque, dependiendo de la naturaleza de las normas de comunicación. En cualquier caso, es posible que se produzca una interrupción del controlador de comunicaciones cuando se está procesando una interrupción de la impresora.

Hay dos enfoques para tratar las interrupciones múltiples. El primero es inhabilitar las in-terrupciones mientras se esté procesando una. Una *interrupción inhabilitada* quiere decir que el procesador ignorará la señal de interrupción. Si durante este tiempo se produce una interrupción, ésta generalmente quedará pendiente y será comprobada por el procesador después que éste habilite las interrupciones. De este modo, cuando un programa de usuario está ejecutándose y se produce una interrupción, se inhabilitan inmediatamente las interrup-ciones. Después de terminar la rutina que trata la interrupción, se habilitan las interrupciones antes de reanudar el programa de usuario y el procesador comprueba si se ha producido al-guna interrupción adicional. Este enfoque es simple y elegante porque las interrupciones se tratan en un orden estrictamente secuencial (figura 1.12a).

La limitación de este enfoque es que no tiene en cuenta las prioridades relativas o necesi-dades críticas en tiempo. Por ejemplo, cuando llega una interrupción desde la línea de comu-nicaciones, se necesita atender ésta rápidamente para hacer sitio a nuevas entradas. Si el pri-mer lote de entrada no ha sido aún procesado cuando llega el segundo, pueden perderse datos.

Un segundo enfoque es definir prioridades para las interrupciones y permitir que una interrupción de una prioridad más alta pueda interrumpir a la rutina de tratamiento de una interrupción de prioridad más baja (figura 1.12b).

Como ejemplo de este segundo enfoque, considérese un sistema con tres dispositivos de E/S: una impresora, un disco y una línea de comunicaciones, con prioridad creciente de 2,4 y 5, respectivamente. La figura 1.13 ilustra una posible secuencia. Un programa de usuario comienza en *t =* 0. En *t*= 10, se produce una interrupción de la impresora; la información del usuario se coloca en la pila del sistema y la ejecución continúa en la *rutina de servicio de la interrupción* (ISR, *Interrupt Service Routine)* de la impresora. Mientras esta rutina sigue ejecutándose, en *t =* 15, se produce una interrupción de comunicaciones. Como la línea de comunicaciones tiene una prioridad más alta que la impresora, la interrupción es atendida. La ISR es interrumpida, su estado se pone en la pila y la ejecución continúa en la ISR de comunicaciones. Mientras esta rutina está ejecutándose, se produce una interrupción del disco (*t =* 20). Debido a que esta interrupción es de más baja prioridad, queda retenida y la ISR de comunicaciones se ejecuta hasta el final.

Cuando se completa la ISR de comunicaciones *(t* = 25), se restaura el estado previo del procesador, lo que significa continuar ejecutando la ISR de la impresora. Sin embargo, antes de ejecutar una sola instrucción de esta rutina, el procesador atiende a la mayor prioridad de la interrupción del disco y transfiere el control a la ISR del disco. Sólo cuando se completa esta rutina (*t* = 35) se reanuda la ISR de la impresora. Cuando se termina esta última *(t =* 40), el control vuelve finalmente al programa de usuario.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Interrupciones 19**

**Multiprogramación**

Aún con el uso de interrupciones, puede que un procesador no esté aprovechado de una manera muy eficiente. Por ejemplo, considérese de nuevo la figura 1.9b. Si el tiempo nece- sario para completar una operación de E/S es mucho mayor que el código del usuario entre llamadas de E/S (una situación habitual), entonces el procesador va a estar desocupado du- rante gran parte del tiempo. Una solución a este problema es permitir que varios programas de usuario estén activos a un mismo tiempo.

Supóngase, por ejemplo, que el procesador tiene que ejecutar dos programas. Uno de ellos es simplemente un programa que lee datos de la memoria y los saca por un dispositivo

*Digitalización con propósito académico*

*Sistemas Operativos*

**20 Introducción a los sistemas informáticos**

externo; el otro es una aplicación que supone una gran cantidad de cálculos. El procesador puede dar comienzo al programa de salida, enviar una orden ESCRIBIR al dispositivo ex- terrno y luego comenzar la ejecución de la otra aplicación.

Cuando el procesador tiene que tratar con una serie de programas, la secuencia en que es- tos se ejecutan dependerá de su prioridad relativa y de si están esperando una E/S. Cuando un programa es interrumpido y se transfiere el control a la rutina de tratamiento de la inte- rrupción, una vez que ésta haya terminado, puede que no se devuelva el control inmediata- mente al programa de usuario que estaba ejecutándose en el momento de la interrupción. En su lugar, el control puede transferirse a algún otro programa pendiente que tenga mayor prioridad. Finalmente, cuando tenga la prioridad más alta, se reanudará el programa de usuario que fue interrumpido. Esta situación de varios programas que se ejecutan por turnos se conoce como *multiprogramación* y será discutida más adelante en el capítulo 2.

1.5

**JERARQUÍA DE MEMORIA**

Las limitaciones de diseño de la memoria de un computador se pueden resumir en tres preguntas: ¿qué cantidad?, ¿qué velocidad? y ¿qué coste?

La cuestión de qué cantidad está relativamente abierta. Según sea la capacidad, probable-mente se construirán aplicaciones que la utilicen. La cuestión de la velocidad es, en cierto sentido, fácil de responder. Para lograr un mayor rendimiento, la memoria debe ser capaz de ir al ritmo del procesador. Es decir, mientras el procesador está ejecutando instrucciones, se-

*Digitalización con propósito académico*

*Sistemas Operativos*

**Jerarquía de memoria 21**

ría conveniente no tener que hacer pausas esperando a instrucciones u operandos. La última pregunta también hay que tenerla en cuenta. Para un sistema práctico, el coste de la memo- ria debe ser razonable en relación a los otros componentes.

Como se puede suponer, estas tres características compiten entre sí: coste, capacidad y tiempo de acceso. Desde siempre se ha utilizado una gran variedad de tecnologías para im-plementar los sistemas de memoria. A lo largo de este abanico de tecnologías, se cumplen las siguientes relaciones:

• A menor tiempo de acceso, mayor coste por bit

• A mayor capacidad, menor coste por bit

• A mayor capacidad, mayor tiempo de acceso

El dilema que se plantea el diseñador es evidente. El diseñador desearía usar tecnologías de memoria que le ofrezcan una gran capacidad, porque se necesita y para que el coste por bit sea bajo. Sin embargo, para cumplir con los requisitos de rendimiento, puede necesitar usar memoria cara, de capacidad relativamente menor y con tiempos de acceso rápidos.

La salida a este dilema no es depender de un único componente de memoria o una tecno-logía, sino emplear *una jerarquía de memoria.* En la figura 1.14a se ilustra una jerarquía tra-dicional. A medida que se desciende por la jerarquía se tienen las siguientes condiciones:

1. Disminución del coste por bit

2. Aumento de la capacidad

3. Aumento del tiempo de acceso

4. Disminución de la frecuencia de acceso a la memoria por parte del procesador

Así pues, las memorias más pequeñas, caras y rápidas son reemplazadas por memorias de más capacidad, más baratas y lentas. La clave del éxito de esta organización es el último punto: disminuir la frecuencia de acceso. Este concepto se examinará en mayor detalle cuando se discuta la *caché,* dentro de este capítulo y la memoria virtual, más adelante en el texto. Ahora se ofrecerá una breve explicación.

Supóngase que el procesador tiene acceso a dos niveles de memoria. El nivel 1 contiene 1000 palabras y tiene un tiempo de acceso de 0,1μs*;* el nivel 2 contiene 100.000 palabras y tiene un tiempo de acceso de l μs. Supóngase que, si la palabra a acceder está en el nivel 1, entonces el procesador accede a ella directamente. Si está en el nivel 2, entonces la palabra se transfiere primero al nivel 1 y después accede el procesador. Para simplificar, se ignorará el tiempo exigido por el procesador para saber si la palabra está en el nivel 1 o en el nivel 2. La figura 1.15 muestra la forma general de la curva que expresa esta situación. Como se puede observar, para altos porcentajes de accesos al nivel 1, el tiempo medio de acceso total es mucho más parecido al del nivel 1 que al del nivel 2.

Así pues, la estrategia funciona en principio, pero solamente si se cumplen las condiciones de la 1 a la 4. La figura 1.16 muestra las características típicas de otros sistemas de memoria. Esta figura demuestra que, empleando varias tecnologías, existe una gama de sistemas de memoria que satisfacen las condiciones de la 1 a la 3. Por fortuna, la condición 4 también se cumple en general.

La base del cumplimiento de la condición 4 es un principio conocido como *cercanía de referencias* [DENN68]. Durante el curso de ejecución de un programa, las referencias a me-moria por parte del procesador, tanto para instrucciones como para datos, tienden a estar agrupadas. Los programas contienen normalmente un cierto número de bucles y de subruti-

*Digitalización con propósito académico*

*Sistemas Operativos*

**22 Introducción a los sistemas informáticos**

nas iterativas. Una vez que se entra en un bucle o en una subrutina, se producirán referencias repetidas a un pequeño conjunto de instrucciones. De forma similar, las operaciones sobre tablas y vectores suponen el acceso a un conjunto de palabras de datos que están agrupadas. Durante un largo período, las agrupaciones en uso cambian, pero, en un período corto, el procesador trabaja principalmente con grupos fijos de referencias a memoria.

Por consiguiente, es posible organizar los datos en la jerarquía de modo que el porcentaje de accesos a los niveles inmediatamente inferiores sea considerablemente menor que el del nivel superior. Considérese el ejemplo de dos niveles ya presentado. Sea el nivel 2 de me- moria el que contiene todos los datos e instrucciones del programa. Las agrupaciones en curso se pueden situar temporalmente en el nivel 1. De cuando en cuando, una de las agru- paciones del nivel 1 tendrá que ser descargada de nuevo al nivel 2, para hacer sitio a alguna nueva agrupación que entre en el nivel 1. En promedio, sin embargo, la mayoría de las refe-

rencias serán a instrucciones y datos contenidos en el nivel 1.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Jerarquía de Memoria 23**

Este principio puede aplicarse con más de dos niveles de memoria. Considérese la jerar- quía que se muestra en la figura 1. 14a. El tipo más rápido y caro de memoria consta de re-gistros internos del procesador. Normalmente, un procesador tendrá unas pocas decenas de registros, aunque algunas máquinas poseen cientos de estos registros. Bajando dos niveles se tiene la memoria principal, también conocida como memoria real, que es la memoria interna principal del computador. Cada ubicación de la memoria principal tiene una única dirección y la mayoría de las instrucciones de máquina se refieren a una o más direcciones de la memoria principal. La memoria principal se suele ampliar con una pequeña memoria caché de alta velocidad. La caché no es generalmente visible para el programador o incluso el procesador. Es un dispositivo para encauzar el movimiento de los datos entre la memoria principal y los registros del procesador y así mejorar el rendimiento.

Las tres formas de memoria descritas son, generalmente, volátiles y emplean tecnologías de semiconductores. El uso de los tres niveles aprovecha la variedad de tipos de las memorias de semiconductores, que difieren en velocidad y coste. Los datos se almacenan de manera permanente en dispositivos externos de almacenamiento masivo, de los cuales los más comunes son los discos y las cintas magnéticas. La memoria externa, no volátil, también se denomina memoria secundaria o auxiliar. Ésta es usada para almacenar programas y archivos de datos y suelen ser visibles para el programador sólo en forma de archivos y registros y no mediante bytes o palabras individuales. Los discos también se usan para ofrecer una ampliación de la memoria principal, conocida como *almacenamiento virtual* o *memoria vir-tual,* que será discutida en los capítulos 6 y 7.

*Digitalización con propósito académico*

*Sistemas Operativos*

**24 Introducción a los sistemas informáticos**

Se pueden incluir otras formas de memoria en la jerarquía. Por ejemplo, los grandes com- putadores centrales *(mainframes)* de IBM incluyen una forma de memoria interna conocida como *memoria expandida,* que utiliza una tecnología de semiconductores más lenta y me- nos cara que la de la memoria principal. Estrictamente hablando, esta tecnología no encaja en la jerarquía sino que constituye una rama paralela: Los datos se pueden mover entre la memoria principal y la memoria expandida, pero no entre la memoria expandida y la me- moria externa. Otras formas de memoria secundaria son los discos ópticos y la memoria de burbuja. Por último, se pueden añadir, por software, niveles adicionales a la jerarquía. Una parte de la memoria principal puede ser utilizada como un buffer para guardar temporal- mente los datos transferidos con el disco. Dicha técnica, más conocida como *caché de disco* (y examinada en detalle en el capítulo 10), mejora el rendimiento de dos formas:

• Las escrituras a disco se agrupan. En lugar de muchas transferencias pequeñas de datos,

se tienen unas pocas transferencias grandes de datos. Esto mejora el rendimiento del disco y reduce la utilización del procesador.

• Algunos datos destinados a la salida pueden ser referenciados por un programa antes del próximo volcado a disco. En tal caso, los datos son recuperados rápidamente desde la caché de software en lugar de hacerse lentamente desde el disco.

La figura 1.14b muestra una jerarquía moderna de memoria que incluye una caché de disco y un disco óptico como otros tipos de memoria secundaria.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Memoria cache 25**

El Apéndice 1A analiza las implicaciones que tienen en el rendimiento las estructuras de memoria multinivel.

1.6

**MEMORIA CACHE2**

Aunque la memoria caché es invisible para el sistema operativo, interactúa con otras partes del hardware de gestión de memoria. Es más, muchos de los principios utilizados en la me- moria virtual son también aplicables a la memoria caché.

**Motivación**

En todos los ciclos de instrucción, el procesador accede a la memoria, al menos una vez, para leer la instrucción y, a menudo, algunas veces más para leer los operandos y/o almacenar los resultados. La frecuencia con que el procesador puede ejecutar las instrucciones está claramente limitada por el tiempo de ciclo de memoria. Esta limitación ha sido de hecho un problema significativo, debido al persistente desacuerdo entre las velocidades del procesador y de la memoria principal. La figura 1.17, que puede considerarse representativa, ilustra esta situación.

La figura muestra que la velocidad de la memoria no se corresponde con la velocidad del procesador. Se debe adoptar un compromiso entre la velocidad, el coste y el tamaño de la memoria. En el mejor de los casos, la memoria principal se construiría con la misma tecno-

2 El término inglés *cache* suele traducirse en la jerga técnica en español por el término *caché* (N. del T.)

*Digitalización con propósito académico*

*Sistemas Operativos*

**26 Introducción a los sistemas informáticos**

logía que los registros del procesador, obteniéndose unos tiempos de ciclo de memoria com-parables con los de los ciclos del procesador. Esta estrategia siempre ha resultado demasiado costosa. La solución es aprovechar el principio de cercanía, disponiendo una memoria pequeña y rápida entre el procesador y la memoria principal, es decir, la caché.

**Principios de la cache**

La memoria caché intenta obtener una velocidad cercana a la de las memorias más rápidas y, al mismo tiempo, aportar una memoria grande al precio de las memorias de semiconductores, que son menos costosas. Este concepto se ilustra en la figura 1.18. Hay una memoria principal más lenta y relativamente grande, junto a una memoria caché más pequeña y rápida. La caché contiene una copia de parte de la memoria principal. Cuando el procesador intenta leer una palabra de la memoria, se comprueba si la palabra está en la memoria caché. Si es así, la palabra se envía al procesador. Si no, se rellena la caché con un bloque de memoria principal, formado por un número fijo de palabras y, después, la palabra es enviada al procesador. Debido al fenómeno de la cercanía de referencias, cuando se carga en la caché un bloque de datos para satisfacer una sola referencia a memoria, es probable que ya se hayan hecho antes otras referencias a palabras del mismo bloque.

La figura 1.19 representa la estructura de un sistema de caché y memoria principal. La memo-ria principal consta de hasta 2n palabras direccionables, teniendo cada palabra una única direc-ción de *n* bits. Se considera que esta memoria consta de un número de bloques de longitud fija de *K* palabras cada uno. Es decir, hay *M* = 2 n/*K* bloques. La caché consta de *C* secciones de *K* palabras cada una y el número de secciones es considerablemente menor que el número de bloques de memoria principal *(C«M).* En todo momento, algún subconjunto de los bloques de memoria reside en las secciones de la caché. Si se va a leer una palabra de un bloque de memoria que no está en caché, entonces el bloque se transfiere para alguna de las secciones de la caché. Debido a que hay más bloques que secciones, una sección individual no puede ser dedicada única y permanentemente a un bloque en particular. De este modo, cada sección incluye un indicador que identifica cuál es el bloque particular que está siendo actualmente almacenado en ella. Este indicador es usualmente un cierto número de los bits más significativos de la dirección.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Memoria cache 27**

La figura 1.20 ilustra la operación de LEER. El procesador genera la dirección *DL,* de la palabra a leer. Si la palabra está en la cache es entregada al procesador. En caso contrario, el bloque que contiene a la palabra se carga en cache y la palabra se envía al procesador.

**Diseño de la cache**

Una discusión detallada sobre la estructura de la cache se escapa del alcance de este libro. Aquí se resumirán de forma breve los elementos clave. Se comprobará que hay que abordar cuestiones similares cuando se haga frente al diseño de la memoria virtual y de la cache del disco. Estos aspectos se encuadran en las siguientes categorías:

•Tamaño de cache.

•Tamaño del bloque.

•Función de correspondencia (*mapping*)

•Algoritmo de reemplazo.

•Política de escritura.

Ya se ha hablado sobre el **tamaño de cache**. De aquí se desprende que caches razona- clemente pequeñas pueden tener un impacto significativo sobre el rendimiento. Otra cues-

*Digitalización con propósito académico*

*Sistemas Operativos*

**28 Introducción a los sistemas informáticos**

tión de tamaño es el **tamaño del bloque,** que es la unidad de intercambio de datos entre la cache y la memoria principal. A medida que el tamaño del bloque aumenta, desde los blo- ques más pequeños a los más grandes, la tasa de aciertos (proporción de veces que una re- ferencia se encuentre en la cache) aumentará al comienzo, debido al principio de cercanía: la alta probabilidad de que se haga referencia en un futuro próximo a datos cercanos a la palabra referenciada. A medida en que el tamaño del bloque crece, pasan a la cache más da- tos útiles. Sin embargo, la tasa de aciertos comenzará a disminuir, dado que el bloque se hace aún mayor y la probabilidad de uso del dato leído más recientemente se hace menor que la probabilidad de reutilizar el dato que hay que sacar de la cache para hacer sitio al bloque nuevo.

Cuando se trae a cache un nuevo bloque de datos, la **función de correspondencia** *(map-ping)* determina la posición de la cache que ocupará el bloque. Dos limitaciones influyen en el diseño de la función de traducción. En primer lugar, cuando un bloque se trae a la cache, puede que otro tenga que ser reemplazado. Convendría hacer esto de forma que se redujera la probabilidad de reemplazar un bloque que se vaya a necesitar en un futuro próximo.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Técnicas de comunicación de E/S 29**

Mientras más flexible sea la función de traducción, mayor será la envergadura del diseño del algoritmo de reemplazo que maximice la tasa de aciertos. En segundo lugar, cuanto más flexible sea la función de traducción, más compleja será la circuitería necesaria para determinar si un bloque dado está en la cache.

El **algoritmo de reemplazo** escoge, bajo las restricciones de la función de traducción, el bloque que hay que reemplazar: Sena conveniente reemplazar aquel bloque que tenga me- nos probabilidad de necesitarse en un futuro cercano. Aunque es imposible identificar un bloque tal, una estrategia bastante efectiva es reemplazar el bloque que lleva más tiempo en la cache sin que se hayan hecho referencias a él. Esta política se denomina algoritmo del *usado hace más tiempo* (LRU, *Least Recently Used).* Se necesitan mecanismos de hardware para identificar el bloque usado hace más tiempo.

Si se modifica el contenido de un bloque de la cache, entonces hace falta escribirlo de nuevo a la memoria principal, antes de reemplazarlo. La **política de escritura** dicta cuándo tiene lugar la operación de escribir en memoria. Por un lado, la escritura puede producirse cada vez que el bloque se actualice. Por otro lado, la escritura se produce sólo cuando se reemplace el bloque. Esta última política reduce las operaciones de escritura en memoria pero deja la memoria principal en un estado obsoleto. Esto puede dificultar la operación de los multiprocesadores y el acceso directo a memoria por parte de los módulos de E/S.

1.7

**TÉCNICAS DE COMUNICACIÓN DE E/S**

Para las operaciones de E/S son posibles las tres técnicas siguientes:

• E/S programada

• E/S dirigida por interrupciones

• Acceso Directo a Memoria (DMA: Direct Memory Access)

**E/S programada**

Cuando el procesador está ejecutando un programa y encuentra una instrucción de E/S, ejecuta dicha instrucción, enviando una orden al módulo apropiado de E/S. Con E/S programada, el módulo de E/S llevará a cabo la acción requerida y luego activará los bits apropiados en el registro de estado de E/S. El módulo de E/S no lleva a cabo ninguna otra acción para avisar al procesador. En particular, no interrumpe al procesador. Así pues, es responsabilidad del procesador comprobar periódicamente el estado del módulo de E/S hasta saber que se ha completado la operación.

Con esta técnica, el procesador es el responsable de extraer los datos de la memoria principal cuando va a hacer una salida o poner los datos en la memoria principal cuando se hace una entrada. El software de E/S se escribe de manera tal que el procesador ejecute unas instrucciones que le otorguen el control directo sobre la operación de E/S, incluyendo la comprobación del estado de los dispositivos, el envío de órdenes de lectura o escritura y la transferencia de los datos. Por lo tanto, en el conjunto de instrucciones se incluyen instrucciones de E/S de las siguientes categorías:

*Digitalización con propósito académico*

*Sistemas Operativos*

**30 Introducción a los sistemas informáticos**

• *Control:* Empleadas para activar un dispositivo externo y decirle qué debe hacer. Por ejemplo, una unidad de cinta magnética puede ser instruida para rebobinar o avanzar un registro.

• *Comprobación:* Empleadas para comprobar varias condiciones de estado asociadas con un módulo de E/S y sus periféricos.

• *Lectura, escritura:* Empleadas para transferir los datos entre los registros del procesa- dor y los dispositivos externos.

La figura 1.21a muestra un ejemplo de utilización de la E/S programada para leer en un bloque de datos desde un dispositivo externo (por ejemplo, un registro de una cinta) hacia la memoria. Los datos se leen por palabra (por ejemplo, 16 bits) cada vez. Para cada palabra que se lea, el procesador debe permanecer en un bucle de comprobación del estado hasta que determine que la palabra está disponible en el registro de datos del módulo de E/S. El diagrama que se muestra en la figura 1.21 destaca la desventaja principal de esta técnica: Es un proceso que consume tiempo y que mantiene al procesador ocupado de forma innecesaria.

**E/S dirigida por interrupciones**

El problema de la E/S programada es que el procesador tiene que esperar un largo rato a que el módulo de E/S en cuestión esté listo para recibir o transmitir más datos. El procesador, mientras está esperando, debe interrogar repetidamente el estado del módulo de E/S. Como resultado, el nivel de rendimiento del sistema en conjunto se degrada fuertemente.

Una alternativa es que el procesador envíe una orden de E/S al módulo y se dedique a ha- cer alguna otra tarea útil. El módulo de E/S interrumpirá entonces al procesador para reque- rir sus servicios cuando esté listo para intercambiar los datos. El procesador ejecuta enton- ces la transferencia de los datos y reanuda el procesamiento anterior.

Seguidamente veremos cómo funciona esto, en primer lugar desde el punto de vista del módulo de E/S. Para la entrada, el módulo de E/S recibe una orden LEER desde el procesador. El módulo de E/S procede entonces con la lectura de los datos desde el periférico asociado. Una vez que los datos están en el registro de datos del módulo, éste envía una señal de interrupción al procesador a través de una línea de control. El módulo espera entonces a que los datos sean solicitados por el procesador. Cuando se haga esta solicitud, el módulo pondrá los datos en el bus de datos y estará listo para otra operación de E/S.

Desde el punto de vista del procesador, la acción para la entrada es como sigue. El procesador envía una orden LEER. Salva entonces el contexto (el contador de programa y los registros del procesador, por ejemplo) del programa en curso, se sale del mismo y se dedica a hacer otra cosa (por ejemplo, el procesador puede estar trabajando con diferentes programas al mismo tiempo). Al finalizar cada ciclo de instrucción, el procesador comprueba si hubo alguna interrupción (figura 1.7). Cuando se produce una interrupción desde el módulo de E/S, el procesador salva el contexto del programa que está ejecutando en ese momento y comienza la ejecución de la rutina de tratamiento de la interrupción. En este caso, el procesador lee la palabra de datos del módulo de E/S y la almacena en la memoria. Luego restaura el contexto del programa que emitió la orden de E/S (o de algún otro programa) y reanuda la ejecución.

La figura 1.2 Ib muestra el uso de E/S por interrupciones para leer en un bloque de datos, Compárese con la figura 1.2ª, La E/S por interrupciones es más eficiente que la E/S programada porque elimina las esperas innecesarias. Sin embargo, la E/S por interrupciones sigue

*Digitalización con propósito académico*

*Sistemas Operativos*

**Técnicas de comunicación de E/S 31**

*Digitalización con propósito académico*

*Sistemas Operativos*

**32 Introducción a los sistemas informáticos**

consumiendo una gran cantidad de tiempo del procesador, debido a que cada palabra de datos que va de la memoria al módulo de E/S o del módulo de E/S a la memoria debe pasar a través del procesador.

Casi siempre habrá varios módulos de E/S en un sistema informático, así que hacen falta mecanismos que capaciten al procesador para determinar qué dispositivo causó la interrupción y decidir, en caso de varias líneas de interrupción, cuál debe tratar primero. En algunos sistemas, hay varias líneas de interrupción, de forma que cada módulo de E/S envía una señal por una línea diferente. Cada línea tiene una prioridad diferente. Otra solución sería habilitar una única línea de interrupción, pero utilizando líneas adicionales para indicar la dirección del dispositivo. De nuevo, a diferentes dispositivos se les asignarán prioridades diferentes.

**Acceso directo a memoria**

La E/S dirigida por interrupciones, aunque es más eficiente que la simple E/S programada, todavía requiere de la intervención activa del procesador para transferir los datos entre la memoria y un módulo de E/S y, además, cualquier transferencia de datos debe recorrer un camino que pasa por el procesador. Así pues, ambas formas de E/S adolecen de dos desventajas inherentes:

1. La velocidad de transferencia de E/S está limitada por la velocidad con la que el procesador puede comprobar y dar servicio a un dispositivo.

2. El procesador participa en la gestión de la transferencia de E/S; debe ejecutarse una se- rie de instrucciones en cada transferencia de E/S.

Cuando se tienen que mover grandes volúmenes de datos, se necesita una técnica más efi-ciente: el acceso directo a memoria (DMA, *Direct Memory Access).* La función de DMA se puede llevar a cabo por medio de un módulo separado sobre el bus del sistema o puede estar incorporada dentro de un módulo de E/S. En cualquier caso, la técnica funciona como sigue. Cuando el procesador desea leer o escribir un bloque de datos, emite una orden hacia el módulo de DMA, enviándole la información siguiente:

• Si lo que se solicita es una lectura o una escritura

• La dirección del dispositivo de E/S involucrado

• La dirección inicial de memoria desde la que se va a leer o a la que se va a escribir

• El número de palabras a leer o escribir

El procesador continúa entonces con otro trabajo. Habrá delegado la operación de E/S en el módulo de DMA y dicho módulo es el que tendrá que encargarse de ésta. El módulo de DMA transfiere el bloque entero, una palabra cada vez, directamente hacia o desde la memoria, sin pasar por el procesador. Cuando se completa la transferencia, el módulo de DMA envía una señal de interrupción al procesador. De esta manera, el procesador se ve involucrado sólo al inicio y al final de la transferencia (figura 1.21c).

El módulo de DMA debe tomar el control del bus para transferir los datos con la me- moria. Debido a la competencia por la utilización del bus, puede ocurrir que el procesador necesite el bus pero deba esperar. Nótese que esto no es una interrupción; el procesador no salva el contexto y se dedica a hacer otra cosa. En su lugar, el procesador hace una pausa durante un ciclo del bus. El efecto general es hacer que el procesador ejecute con más len-

*Digitalización con propósito académico*

*Sistemas Operativos*

**Lecturas recomendadas 33**

titud durante una transferencia de DMA. No obstante, para una transferencia de E/S de varias palabras, el DMA es bastante más eficiente que la E/S programada o la dirigida por interrupciones.

Cuando el módulo de E/S en cuestión es un sofisticado canal de E/S, el concepto de DMA debe tenerse en cuenta con más razón. Un canal de E/S es un procesador propiamente dicho, con un conjunto especializado de instrucciones, diseñadas para la E/S. En un sistema informático con tales dispositivos el procesador no ejecuta instrucciones de E/S. Dichas instrucciones se almacenan en la memoria principal para ser ejecutadas por el propio canal de E/S. Así pues, el procesador inicia una transferencia de E/S instruyendo al canal de E/S para ejecutar un programa en memoria. El programa especifica el o los dispositivos, la zona o zonas de memoria para almacenamiento, la prioridad y la acción que llevar a cabo bajo ciertas condiciones de error. El canal de E/S sigue estas instrucciones y controla la transferencia de datos, informando de nuevo al procesador al terminar.

2 parte del capitulo tres(yo tengo la primera parte)

**Control de procesos 125**

**FIGURA 3.12 Estructuras de colas de procesos**

• Un error en una sola rutina, como la de tratamiento de interrupciones, puede dañar los bloques de control de procesos, lo que destruirla la capacidad del sistema para administrar los procesos afectados.

• Un cambio de diseño en la estructura o en la semántica del bloque de control de procesos podría afectar a varios módulos del sistema operativo.

Estos problemas se pueden abordar exigiendo a todas las rutinas del sistema operativo que pasen a través de una rutina de manejo, cuya única tarea serla la de proteger los bloques de control de proceso y que se constituiría en el único árbitro para leer y escribir en estos bloques. La concesión en el empleo de una rutina tal está en el rendimiento y en el grado con el que pueda confiarse en que el resto del software del sistema sea correcto.

**3.3**

**CONTROL DE PROCESOS**

**Modos de ejecución**

Antes de continuar la discusión sobre la forma en que el sistema operativo gestiona los procesos, hace falta distinguir entre el modo de ejecución del procesador que normalmente se asocia con el sistema operativo y el modo que normalmente se asocia con los programas

*Digitalización con propósito académico*

*Sistemas Operativos*

**126 Descripción y control de procesos**

de usuario. La mayoría de los procesadores dan soporte para dos modos de ejecución por lo menos. Ciertas instrucciones pueden ejecutarse solo en modo privilegiado. Entre éstas están la lectura o modificación de registros de control (como la palabra de estado del programa), instrucciones primitivas de E/S e instrucciones relativas a la gestión de memoria. Además, se puede acceder a ciertas regiones de memoria solo en el modo más privilegiado.

El modo menos privilegiado a menudo se conoce como *modo de usuario,* ya que los pro-gramas de usuario ejecutan normalmente en ese modo. Al modo más privilegiado normalmente se le conoce como *modo del sistema, modo de control* o*, modo del núcleo.* Este último término se refiere al núcleo del sistema operativo, que es la parte del sistema operativo que lleva a cabo las funciones importantes del sistema. La tabla 3.10 enumera las funciones que normalmente se hallan en el núcleo de un sistema operativo.

La razón por La que se usan dos modos debe quedar clara. Es necesario proteger al sistema operativo y a las tablas importantes del mismo, tales como los bloques de control de procesos, de las injerencias de los programas de usuario. En el modo del núcleo, el software tiene control completo del procesador y de todas sus instrucciones, registros y memoria. Este nivel de control no es necesario y, por seguridad, tampoco conveniente para los programas de usuario.

Surgen dos preguntas: ¿Cómo conoce el procesador en qué modo va a ejecutar? ¿Cómo se cambia de modo? Para la primera pregunta, normalmente hay un bit en la PSW que indica el modo de ejecución. El bit es cambiado como respuesta a ciertos sucesos. Por ejemplo, cuando un usuario hace una llamada a un servicio del sistema operativo, el modo se cambia al de núcleo. Esto se suele llevar a cabo ejecutando una instrucción que cambia el modo. Un ejemplo de cómo se hace esto es la instrucción de Cambio de Modo (CHM, *Change Mode)* del VAX. Cuando el usuario hace una llamada a un servicio del

**TABLA 3.10 Funciones Básicas del Núcleo de un Sistema Operativo**

**Gestión de Procesos**

• Creación y terminación de los procesos

• Planificación y expedición de los procesos

• Cambio de procesos

• Sincronización de procesos y soporte para la comunicación entre procesos

• Gestión de los bloques de control de procesos

**Gestión de memoria**

• Asignación de espacios de direcciones a los procesos

• Intercambio

• Gestión de páginas y segmentos

**Gestión de E/S**

• Gestión de buffers

• Asignación de canales de E/S y dispositivos a los procesos

**Funciones de Soporte**

• Tratamiento de interrupciones

• Contabilidad

• Supervisión

*Digitalización con propósito académico*

*Sistemas Operativos*

**Control de procesos 127**

sistema o cuando una interrupción transfiere el control a una rutina del sistema, la rutina ejecuta CHM para entrar en un modo más privilegiado y la ejecuta de nuevo para pasar a un modo menos privilegiado, antes de devolver el control al proceso del usuario. Si un programa de usuario intenta ejecutar un CHM, se originará simplemente una llamada al sistema operativo, que devolverá un error a menos que esté permitido el cambio de modo.

**Creación de procesos**

Anteriormente, en la sección 3.1, se discutieron los sucesos que conducían a la creación de un nuevo proceso. Una vez tratadas las estructuras de datos asociadas a los procesos, se está en condiciones de describir brevemente los pasos que entran en juego en la creación real de los procesos.

Una vez que el sistema operativo decide, por alguna razón (tabla 3.1), crear un nuevo pro-ceso, éste puede proceder como sigue:

1. Asignar un único identificador al nuevo proceso. En ese momento se añade una nueva entrada a la tabla principal de procesos, que contiene una entrada por proceso.

2. Asignar espacio para el proceso. Esto incluye todos los elementos de la imagen del pro-ceso. Así pues, el sistema operativo debe saber cuánto espacio se necesitará para el espa-cio privado de direcciones del usuario (programas y datos) y para la pila del usuario. Estos valores se pueden asignar por omisión en función del tipo de proceso o bien se puede asignar a partir de la solicitud del usuario cuando se crea el trabajo. Si un proceso es ge-nerado por otro, el proceso padre puede pasarle al sistema operativo los valores necesarios como parte de la solicitud de creación del proceso. Si algún espacio de direcciones exis-tente se va a compartir con este nuevo proceso, entonces se deben establecer los enlaces adecuados. Por último, se debe asignar espacio para el bloque de control del proceso.

3. Debe inicializarse el bloque de control del proceso. La parte de identificación del proceso contiene el ID de este proceso junto a otros ID apropiados, tales como el del proceso padre. La parte de información del estado del procesador normalmente se inicializa con la mayor parte de las entradas a cero, excepto para el contador de programa (que se prepara con el punto de entrada del programa) y los punteros a las pilas del sistema (que establecen los limites de la pila del proceso). La parte de información de control del procesador se inicializa a partir de los valores estándares por omisión y los atributos que se han solicitado para el proceso. Por ejemplo, el estado del proceso suele inicializarse a Listo o a Listo y suspendido. La prioridad puede asignarse por omisión al valor más bajo, a menos que se haya hecho una solicitud explicita de un valor mayor. Inicialmente, puede que el proceso no posea ningún recurso (dispositivos de E/S, archivos), a menos que se haya hecho una solicitud explicita de los mismos o a menos que se hayan heredado del proceso padre.

4. Se deben establecer los enlaces apropiados. Por ejemplo, si el sistema operativo mantiene cada cola de planificación como una lista enlazada, entonces el proceso nuevo se debe poner en la cola de Listos o de Listos y suspendidos.

*5.* Puede haber otras estructuras de datos que crear o ampliar. Por ejemplo, el sistema operativo puede mantener un archivo de contabilidad para cada proceso que sea utilizado más tarde con propósitos de facturación y/o evaluación del rendimiento.

*Digitalización con propósito académico*

*Sistemas Operativos*

**128 Descripción y control de procesos**

**Cambio de proceso**

A primera vista, la función de cambio de proceso parece sencilla. En cierto momento, un proceso que está ejecutándose se interrumpe, el sistema operativo pone a otro proceso en el estado de Ejecución y pasa el control a dicho proceso. Sin embargo, surgen diversas cuestiones de diseño. En primer lugar, ¿qué sucesos provocan un cambio de proceso? Otra cuestión es que se debe hacer una distinción entre cambio de contexto y cambio de proceso. Por último, ¿qué debe hacer el sistema operativo con las diferentes estructuras de datos bajo su control para llevar a cabo un cambio de proceso?

*Cuándo cambiar de proceso*

Un cambio de proceso puede producirse en cualquier momento en que el sistema operativo haya tornado el control a partir del proceso que está actualmente ejecutándose. La tabla 3.11 propone los sucesos posibles que pueden dar el control al sistema operativo.

En primer lugar, se van a tener en cuenta las interrupciones del sistema. Se pueden distinguir, como hacen muchos sistemas, dos clases de interrupciones del sistema, una conocida simplemente como interrupción y otra conocida como cepo. La primera es originada por algún tipo de suceso que es externo e independiente del proceso que está ejecutándose, como la culminación de una operación de E/S. La segunda tiene que ver con una condición de error o de excepción generada dentro del proceso que está ejecutándose, como un intento ilegal de acceso a un archivo. En una **interrupción ordinaria, el control** se transfiere primero a un gestor de interrupciones, quien lleva a cabo algunas tareas básicas y, después, se salta a una rutina del sistema operativo que se ocupa del tipo de interrupción que se ha producido. Algunos ejemplos son los siguientes:

*• Interrupción de reloj:* El sistema operativo determina si el proceso que está en ejecución ha estado ejecutando durante la fracción máxima de tiempo permitida. Si esto ocurre, el proceso debe pasar al estado Listo y se debe expedir otro proceso.

*• Interrupción de E/S:* El sistema operativo determina exactamente que se ha producido una acción de E/S. Si la acción constituye un suceso que están esperando uno o más procesos, entonces el sistema operativo traslada todos los procesos bloqueados correspondientes al estado Listo (y los procesos Bloqueados y suspendidos pasan al estado de Listos y suspendidos). El sistema operativo debe entonces decidir si se reanuda La ejecución del proceso que está actualmente en estado de Ejecución o se expulsa a dicho proceso en favor de un proceso Listo de mayor prioridad.

**TABLA 3.11 Mecanismos para la interrupción de la Ejecución de un Proceso [KRAK88]**

Mecanismo Causa Uso

Interrupción Externa a la ejecución de Reacción a un suceso

la instrucción en curso asincrónico externo

Cepo Asociada con a ejecución de Tratamiento de un error o de

la instrucción en curso una condición excepcional

Llamada del supervisor Solicitud explicita Llamada a una función

del sistema operativo

*Digitalización con propósito académico*

*Sistemas Operativos*

**Control de procesos 129**

*• Fallo de memoria:* El procesador encuentra una referencia a una dirección de memoria virtual de una palabra que no está en memoria principal. El sistema operativo debe traer el bloque (página o segmento) que contiene la referencia, de la memoria secundaria a la memoria principal. Después de hacer la solicitud de E/S para traer el bloque de memoria, el sistema operativo puede llevar a cabo un cambio de contexto para reanudar la ejecución de otro proceso; el proceso que cometió el fallo de memoria se pasa a estado Bloqueado. Después de que el bloque en cuestión se cargue en memoria, dicho proceso se pondrá en estado Listo.

En los cepos, el sistema operativo determina si el error es fatal. Si lo es, el proceso que se estaba ejecutando pasa al estado de Terminado y se produce un cambio de proceso. Si no es fatal, la acción del sistema operativo dependerá de La naturaleza del error y del diseño del sistema operativo. Se puede intentar algún procedimiento de recuperación o, simplemente, notificarlo al usuario. Se puede hacer un cambio de proceso o, simplemente, reanudar el mismo proceso que se estaba ejecutando.

Finalmente, el sistema operativo puede activarse mediante una llamada de supervisor desde e programa que se está ejecutando. Por ejemplo, está ejecutándose un proceso de usuario y se alega a una instrucción que solicita una operación de E/S, tal como abrir un archivo. Esta llamada provoca la transferencia a una rutina que forma parte del código del sistema operativo. Por lo general, el uso de una llamada al sistema hace que el proceso de usuario pase al estado Bloqueado.

**Cambio de contexto**

En el capitulo 1 se discutió sobre la inclusión de un ciclo de interrupción como parte del ciclo de instrucción. Recuérdese que, en el ciclo de interrupción, el procesador comprueba si se ha producido alguna interrupción, lo que se indicaría por la presencia de una señal de interrupción. Si no hay pendiente ninguna interrupción, el procesador continúa con el ciclo de lectura de la instrucción siguiente del programa en curso del proceso actual. Si hay alguna interrupción pendiente, el procesador hace lo siguiente:

1. Salva el contexto del programa que está ejecutándose.

2. Asigna al contador de programa el valor de la dirección de comienzo de un programa de *tratamiento de la interrupción.*

El procesador continúa entonces con el ciclo de lectura de instrucción y trae la primera instrucción del programa de tratamiento de interrupciones, que atenderá a la interrupción.

Una pregunta que puede plantearse es: ¿Que es lo que constituye el contexto que se salva? La respuesta es que se debe incluir cualquier información que pueda alterarse por la ejecución de la rutina de tratamiento de la interrupción y que pueda sen necesaria para reanudar el programa que fue interrumpido. Así pues, debe salvarse la parte del bloque de control del proceso que se denomina información de estado del procesador. Esto incluye el contador de programa, otros registros del procesador y la información de la pila.

¿Hace falta hacer algo más? Ello dependerá de lo que ocurra a continuación. La rutina de tratamiento de la interrupción es normalmente un programa corto que lleva a cabo unas pocas tareas básicas relacionadas con una interrupción. Por ejemplo, se marca el indicador que señala la presencia de una interrupción, puede enviar un acuse de recibo a la entidad que produjo la interrupción (como un módulo de E/S) y puede hacer algunas labores básicas relacio

*Digitalización con propósito académico*

*Sistemas Operativos*

**130 Descripción y control de procesos**

nadas con los efectos del suceso que causó la interrupción. Por ejemplo, si la interrupción está relacionada con un suceso de E/S, el gestor de interrupciones comprobará condiciones de error. Si se ha producido un error, la rutina de tratamiento puede enviar una señal al proceso que solicitó originalmente la operación de E/S*.* Si La interrupción es de reloj, el gestor deberá darle el control al distribuidor, quien querrá pasarle el control a otro proceso, puesto que la fracción de tiempo asignada al proceso que estaba ejecutándose habrá expirado.

¿Qué sucede con el resto de información del bloque de control del proceso? Si la interrupción va a venir seguida de un cambio a otro proceso, entonces hace falta hacer cierto trabajo. No obstante, la palabra clave en la oración anterior es la condiciona *si.* En la mayoría de los sistemas operativos, el acontecimiento de una interrupción no provoca necesariamente un cambio de proceso. Es posible que después de que el gestor de interrupciones haya ejecutado, el proceso que estaba ejecutándose reanude su ejecución. En tal caso, todo lo que hay que hacer es salvar la información de estado del procesador cuando se produzca la interrupción y restaurar dicha información cuando el control vuelva al programa que estaba en marcha. Las funciones de salvar y restaurar suelen llevarse a cabo en el hardware.

***Cambio de estado de los procesos***

Está claro entonces que el cambio de contexto es un concepto distinto del cambio de proceso.6 Puede producirse un cambio de contexto sin cambiar el estado del proceso que está actualmente en estado de Ejecución. En tal caso, salvar el contexto y restaurarlo posteriormente involucra un pequeño coste extra. Sin embargo, si el proceso que estaba ejecutándose tiene que pasar a otro estado (Listo, Bloqueado, etc.), el sistema operativo tiene que llevar a cabo cambios substanciales en su entorno. Los pasos involucrados en un cambio completo de proceso son los siguientes:

1. Salvar el contexto del procesador, incluyendo el contador de programa y otros registros.

2. Actualizar el bloque de control del proceso que estaba en estado de Ejecución. Esto implica cambiar el estado del proceso a alguno de los otros estados (Listo, Bloqueado, Listo y suspendido, Terminado). También se tienen que actualizar otros campos significativos, incluyendo la razón por la que se abandona el estado de Ejecución y la información de contabilidad.

3. Mover el bloque de control del proceso a la cola apropiada (Listos, Bloqueados por el Suceso *i,* Listos y suspendidos).

4. Seleccionar otro proceso para ejecución; este tema se explora en los capítulos 6 y 7.

5. Actualizar el bloque de control del proceso seleccionado. Esto incluye cambiar el estado del proceso a Ejecución.

6. Actualizar las estructuras de datos de gestión de memoria. Esto puede hacer falta de-pendiendo de cómo se gestione la traducción de direcciones; este tema se analiza en los capítulos 6 y 7.

7. Restaurar el contexto del procesador a aquel que existía en el momento en el que el proceso seleccionado dejó por última vez el estado de Ejecución, cargando los valores previos del contador de programa y de otros registros.

\_\_\_\_\_\_\_\_\_\_\_

*6* Desafortunadamente, algunos libros de texto sobre el tema usan el término *cambio de contexto* queriendo decir *cambio de proceso* y no tienen un término particular para la sencilla acción que aquí se ha definido como cambio de contexto.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Control de procesos 131**

Así pues, el cambio de proceso, que supone un cambio de estado, requiere un esfuerzo considerablemente mayor que un cambio de contexto.

**Ejecución del sistema operativo**

En el capitulo 2 se señalaron algunos hechos curiosos sobre los sistemas operativos:

• El sistema operativo funciona de la misma forma que un software corriente, es decir, es un programa ejecutado por el procesador.

• El sistema operativo abandona frecuentemente el control y debe depender de que el procesador le permita recuperarlo.

Si el sistema operativo es solamente una colección de programas y si es ejecutado por el procesador, como cualquier otro programa, ¿es el sistema operativo un proceso? Si lo fuese, ¿cómo se controla?

Estas interesantes preguntas han merecido variadas respuestas de los diseñadores de sistemas operativos. La figura 3.13 ilustra un rango de enfoques que pueden encontrarse en varios de los sistemas operativos contemporáneos.

***Núcleo fuera de todo proceso***

Un enfoque bastante tradicional y habitual en muchos de los sistemas operativos más antiguos es ejecutar el núcleo del sistema operativo fuera de cualquier proceso (figura 3.1 3a). Con este enfoque, cuando el proceso en ejecución es interrumpido o hace una llamada de supervisor, se salva el contexto del procesador para este proceso y se pasa el control al núcleo. El sistema operativo tiene su propia región de memoria y su propia pila del sistema para controlar las llamadas y retornos de procedimientos. El sistema operativo puede llevar a cabo cualquier función deseada y luego restaurar el contexto del proceso interrumpido para reanudarlo. Otra solución seria que el sistema operativo pudiese completar la función de salvar el entorno del proceso y continuar con la planificación y expedición de otro proceso. Que sea esto lo que pase depende de la causa de la interrupción y de las circunstancias del momento.

En cualquier caso, el punto clave es que se considera que el concepto de proceso se aplica solo a los programas del usuario. El código del sistema operativo se ejecuta como una entidad separada que opera en modo privilegiado.

***Ejecución dentro de los procesos de usuario***

Una alternativa que es común en los sistemas operativos de máquinas pequeñas (minicom-putadores y microcomputadores) es ejecutar casi todo el software del sistema operativo en el contexto de un proceso de usuario. El enfoque es que el sistema operativo es principalmente una colección de rutinas que el usuario llama para llevar a cabo varias funciones y que son ejecutadas dentro del entorno del proceso de usuario, como se ilustra en la figura 3. 13b. En un punto dado cualquiera, el sistema operativo estará gestionando *N* imágenes de procesos. Cada imagen incluye no solo las regiones que se ilustran en la figura 3.11, sino también zonas de programa, de datos y de pila para los programas del núcleo.

La figura 3.14 propone una estructura típica para la imagen de un proceso de usuario según esta estrategia. Una pila del núcleo separada se utiliza para gestionar las llamadas y los retornos mientras que el proceso esté en el modo del núcleo. El código y los datos del sistema operativo están en el espacio de direcciones compartidas y son compartidos por todos los procesos de usuario. *Digitalización con propósito académico*

*Sistemas Operativos*

**132 Descripción y control de procesos**

**FIGURA 3.13 Relación entre el sistema operativo y los procesos de usuario**

Cuando se produce una interrupción, un cepo o una llamada del supervisor, el procesador se pone en modo del núcleo y el control pasa al sistema operativo. Con tal fin, se salva el contexto del procesador y tiene lugar un cambio de contexto hacia una rutina del sistema operativo. Sin embargo, la ejecución continúa dentro del proceso de usuario en curso. De esta manera, no se ha llevado a cabo un cambio de proceso, sino un cambio de contexto dentro del mismo proceso.

Si el sistema operativo, al completar su trabajo, determina que el proceso en curso debe continuar ejecutando, entonces se lleva a cabo un cambio de contexto para reanudar el programa interrumpido del proceso en curso. Esta es una de las ventajas clave de este enfoque: Un programa de usuario se interrumpe para emplear alguna rutina del sistema operativo, luego se reanuda y todo se produce sin la penalización de dos cambios de proceso. No obstante, si se determina que va a producirse un cambio de proceso en vez de retomar a! programa que estaba ejecutándose previamente, entonces se pasa el control a una rutina que hace el cambio entre procesos. Esta rutina puede o no ejecutar en el proceso en curso, dependiendo del diseño del sistema. Sin embargo, en algún punto, el proceso en curso debe ponerse en estado de No Ejecución y otro proceso debe designarse como proceso en Ejecución. Durante esta fase, es lógicamente más conveniente considerar que la ejecución tiene lugar fuera de todos los procesos.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Control de procesos 133**

**FIGURA 3.14 Imagen del proceso: el sistema operativo ejecuta dentro del proceso de usuario**

En cierto sentido, este enfoque del sistema operativo es bastante singular Expuesto de forma simple, cada cierto tiempo, se salvará la información de estado de un proceso, se escogerá otro proceso para ejecutar de entre todos los que estén listos y se cederá el control a dicho proceso. La razón por la que ésta no es una situación arbitraria ni mucho menos caótica es que, durante el tiempo critico, el código que se ejecuta en el proceso de usuario es código compartido por el sistema operativo y no código del usuario. Debido al concepto de modo de usuario y modo del núcleo, el usuario no puede entrometerse ni estorbar a las rutinas del sistema operativo, aún cuando éstas estén ejecutando en el entorno del proceso de usuario. Esto sirve para recordar una vez más que existe una distinción entre los conceptos de proceso y programa y que la relación entre los dos no es de uno a uno. Dentro de un proceso pueden ejecutarse tanto un programa de usuario como uno del sistema operativo y los programas del sistema operativo que ejecutan en los diferentes procesos de usuario son idénticos.

***Sistema operativo basado en procesos***

Una última alternativa, ilustrada en la figura 3. 13c, es la de implementar el sistema operativo como una colección de procesos del sistema. Al igual que en las otras opciones, el software que forma parte del núcleo ejecutará en modo de núcleo. En este caso, sin embargo, las funciones más importantes del núcleo se organizan en procesos separados. Una vez más, puede haber una pequeña cantidad de código de cambio de procesos que se debe ejecutar fuera de todo proceso.

*Digitalización con propósito académico*

*Sistemas Operativos*

**134 Descripción y control de procesos**

Este enfoque tiene varias ventajas. Impone unas normas de diseño de programas que pro-mueven el uso de un sistema operativo modular con unas interfaces mínimas y claras entre los módulos. Además, algunas funciones no criticas del sistema operativo se pueden implementas como procesos separados. Por ejemplo, se mencionó anteriormente un programa supervisor que registrara el nivel de utilización de Los recursos (procesador, memoria, canales) y la velocidad de progreso de los procesos de usuario en el sistema. Como este programa no provee un servicio particular a ningún proceso activo, puede ser invocado solamente por el sistema operativo. Como un proceso, la función podrá ejecutar con un nivel de prioridad asignado y ser intercalada con otros procesos bajo el control del distribuidor. Por último, implementar el sistema operativo como un conjunto de procesos es útil en un entorno de multiprocesador o de varios computadores, en el cual algunos de los servicios del sistema operativo pueden enviarse a procesadores dedicados, mejorando así el rendimiento.

**Micronúcleos**

Un concepto que ha recibido mucha atención últimamente es el de micronúcleo *(microkernel).* Un micronúcleo es un pequeño núcleo de sistema operativo que proporciona las bases para ampliaciones modulares. Sin embargo, el término es algo confuso, pues hay una serie de cuestiones sobre los micronúcleos cuyas respuestas son diferentes según los distintos equipos de diseño de sistemas operativos. Estas cuestiones son sobre lo pequeño que debe ser el núcleo para ser calificado de micronúcleo, sobre cómo diseñar los controladores *(drivers)* de dispositivos para alcanzar el mejor rendimiento a la vez que sus funciones se independizan del hardware, si las operaciones que no sean del núcleo deben ejecutar en el espacio del núcleo o en el del usuario, y si se debe mantener el código de los subsistemas existentes (por ejemplo, una versión de UNIX) o empezar desde cero.

El enfoque de los micronúcleos fue popularizado por el sistema operativo Mach y sus im-plementaciones en la línea de computadores de Next. En teoría, el enfoque del núcleo se supone que brinda una gran flexibilidad y modularidad. En la práctica, este beneficio fue hasta cierto punto denegado por el sistema operativo servidor monolítico BSD 4.3 que Next construyó en torno a Mach. Otra aplicación muy conocida del enfoque de micronúcleos es Windows NT, que proclama no solo la modularidad, sino la portabilidad, como los beneficios clave. El núcleo está rodeado por una serie de subsistemas compactos de forma que facilita la tarea de implementar NT en una gran variedad de plataformas. Actualmente, otros productos presumen de tener implementaciones de micronúcleo y este enfoque general de diseño parece que va asentarse en casi todos los computadores personales, estaciones de trabajo y sistemas operativos servidores que se desarrollen en el futuro próximo [VARH94].

La filosofía en que se basa el micronúcleos es que solo las funciones absolutamente esen- ciales del sistema operativo deben permanecer en el núcleo. Las aplicaciones y los servicios menos esenciales se construyen sobre el micronúcleo. Aunque la línea divisoria de lo que esta adentro y lo que está afuera del micronúcleo varia de un diseño a otro, la característica común es que muchos servicios que tradicionalmente han formado parte del sistema operativo ahora son subsistemas externos que interactúan con el núcleo y con otros **subsistemas; aquí se incluyen** los sistemas de archivo, los servicios de ventana y los servicios de seguridad.

Una vez más aunque los diseños específicos de cada micronúcleo pueden diferir en general la arquitectura de micronúcleo tiende a reemplazar la estratificación tradicional, en ca-

*Digitalización con propósito académico*

*Sistemas Operativos*

**Procesos e hilos 135**

pas verticales, de un sistema operativo, por una horizontal. Las componentes del sistema operativo externas al micronúcleo interactúan una con otra sobre una base común, normalmente a través de mensajes distribuidos a través del micronúcleo. De este modo, el micronúcleo funciona como un distribuidor de mensajes: Valida los mensajes, los pasa entre las componentes y otorga el acceso al hardware. Esta estructura es ideal para entornos de proceso distribuido, ya que el micronúcleo puede pasar mensajes tanto en local como en remoto, sin necesidad de cambios en otras componentes del sistema operativo.

**3.4**

**PROCESOS E HILOS**

En la discusión llevada a cabo, se ha presentado el concepto de proceso incluyendo las dos características siguientes:

*• Unidad de propiedad de los recursos:* A cada proceso se le asigna un espacio de direcciones virtuales para albergar a la imagen del proceso y, de cuando en cuando, a! proceso se le puede asignar memoria virtual y otros recursos, tales como canales de E/S, dispositivos de E/S y archivos.

*• Unidad de expedición:* Un proceso es un camino de ejecución (traza) a través de uno o más programas. Esta ejecución puede ser intercalada con la de otros procesos. De este modo, un proceso tiene un estado de ejecución (Ejecución, Listo, etc.) y una prioridad de expedición. La unidad planificada y expedida por el sistema operativo es el proceso.

En la mayoría de los sistemas operativos, estas dos características son, de hecho, la esencia de un proceso. Sin embargo, algunos argumentos pueden convencer de que estas dos características son independientes y que deben ser tratadas de manera independiente por el sistema operativo. Esto se hace así en una serie de sistemas operativos, en particular en algunos sistemas operativos de desarrollo reciente. Para distinguir estas dos características, la unidad de expedición se conoce corno **hilo** *(thread)* o proceso ligero *(lightweight process),* mientras que a la unidad de propiedad de los recursos se le suele llamar **proceso** o **tarea.7**

**Varios hilos en un solo proceso**

La utilización que más salta a la vista del concepto de hilo es la de una disposición en la que pueden existir varios hilos dentro de un mismo proceso. Algo aproximado a este enfoque es lo que se hace en MVS. De una forma más explicita, este enfoque es el asumido por Windows NT, OS/2, la versión que tiene Sun del UNIX y un importante sistema operativo conocido como Mach LAC92, RASH89, TEVA89]. Mach es una evolución ampliada de UNIX que se utiliza en las estaciones Next y que forma la base de la versión de UNIX de la Fundación para el Software Abierto (OSF, *Open Software Foundation).* Este apartado describe el enfoque tornado en Mach; las técnicas utilizadas en Windows NT y MVS se discuten en la sección *3.5.*

***\_\_\_\_\_\_\_\_\_\_\_\_***

7Desafortunadamente, incluso este grado de consistencia no se puede mantener. En el MVS, los conceptos de espacio de direcciones y tarea, respectivamente, se corresponden a grandes rasgos con los conceptos de proceso e hilo que se describen en esta sección.

*Digitalización con propósito académico*

*Sistemas Operativos*

**136 Descripción y control de procesos**

Mach está diseñado específicamente para trabajar en un entorno multiprocesador, aunque también se adapta bien a los sistemas monoprocesadores. En Mach, una tarea se define como la unidad de protección o unidad de asignación de recursos. A las tareas se les asocian los siguientes elementos:

• Un espacio de direcciones virtuales, que contiene la imagen de la tarea

• Acceso protegido a los procesadores, otros procesos (para la comunicación entre proce-sos), archivos y recursos de E/S (dispositivos y canales)

En una tarea pueden haber uno o más hilos, cada uno con lo siguiente:

• El estado de ejecución del hilo (Ejecución, Listo, etc.)

• El contexto del procesador, que se salva cuando no está ejecutando; una forma de contem-plar al hilo es con un contador de programa independiente operando dentro de una tarea

• Una pila de ejecución

• Almacenamiento estático para las variables locales

• Acceso a la memoria y a los recursos de la tarea, que se comparten con todos los otros hilos de la tarea

Los beneficios clave de los hilos se derivan de las implicaciones del rendimiento: Se tarda mucho menos tiempo en crear un nuevo hilo en un proceso existente que en crear una nueva tarea, menos tiempo para terminar un hilo y menos tiempo para cambiar entre dos hilos de un mismo proceso. Por tanto, si hay una aplicación o una función que pueda implementarse como un conjunto de unidades de ejecución relacionadas, es más eficiente hacerlo con una colección de hilos que con una colección de tareas separadas.8 Algunos estudios llevados a cabo por los desarrolladores de Mach demuestran que la aceleración en la creación de procesos, comparada con la de Las implementaciones de UNIX que no utilizan hilos, está en un factor de 10 [ITEVA87].

Un ejemplo de aplicación que podría hacer uso de hilos es un servidor, como puede ser un servidor de archivos de una red de área local. Cada vez que llegue una solicitud de un nuevo archivo, se puede generar un nuevo hilo para el programa de gestión de archivos. Puesto que el servidor debe manejar muchas solicitudes, se crearán y destruirán muchos hilos en un corto periodo de tiempo. Si el servidor es un multiprocesador, se pueden ejecutar varios hilos de una misma tarea simultáneamente y en diferentes procesadores. Los hilos son también útiles en los monoprocesadores para simplificar la estructura de los programas que lleven a cabo diversas funciones. Otros ejemplos de uso efectivo de los hilos es en las aplicaciones de proceso de comunicaciones [COOP90] y en los supervisores de transacciones.

Otra forma en la que los hilos aportan eficiencia es en la comunicación entre diferentes pro-gramas en ejecución. En la mayoría de los sistemas operativos, Ia comunicación entre procesos independientes requiere la intervención del núcleo para ofrecer protección y para proporcionar

**\_\_\_\_\_\_\_\_\_**

8 Es interesante comentar que han aparecido conceptos similares al mismo tiempo en campos tales como los protocolos de co-municación, la arquitectura de computadores y los sistemas operativos. En la arquitectura de computadores, el enfoque de las computadores con juego reducido de instrucciones (RISC, *Reduced Instruction Set Computer)* ha mejorado la velocidad del procesador, perfeccionando la arquitectura del procesador. En los protocolos de comunicaciOn, la necesidad de altas velocidades de transferencia de datos entre las redes de computadores ha provocado el desarrollo de protocolos ligeros de transporte. Estos dos conceptos se discuten con amplitud en [TAL93a] y [TAL94a], respectivamente.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Procesos e hilos 137**

los mecanismos necesarios para la comunicación. Sin embargo, puesto que los hilos de una misma tarea comparten memoria y archivos, pueden comunicarse entre sI sin invocar al núcleo.

[ETW88] da cuatro ejemplos de uso de los hilos en un sistema de multitarea:

*• Trabajo interactivo* y *de fondo:* Esto se produce en el sentido de la interacción directa con el usuario, no en el de sesiones interactivas y de fondo. Por ejemplo, en un programa de hoja de cálculo, un hilo puede estar visualizando los menús y leyendo la entrada del usuario mientras que otro hilo ejecuta las órdenes y actualiza la hoja de cálculo. Esta medida suele aumentar la velocidad que se percibe de la aplicación, permitiendo que el programa pida la orden siguiente antes de terminar la anterior.

*• Proceso asíncrono:* Los elementos asíncronos del programa se pueden implementar como hilos. Por ejemplo, para protegerse de un corte de alimentación, se puede diseñar un procesador de textos que escriba su buffer de la RAM al disco una vez por minuto. Se puede crear un hilo cuya única tarea sea hacer estas copias de respaldo periódicas y que se planifique directamente con el sistema operativo; no hay necesidad de ningún código superfluo en el programa principal que haga la comprobación de tiempo o que coordine la entrada y la salida.

*• Aceleración de la ejecución:* Un proceso con hilos múltiples puede computar un lote de datos mientras lee el lote siguiente de un dispositivo. En un sistema con multiproceso, varios hilos de un mismo proceso podrán ejecutar realmente a la vez.

*• Organización de los programas:* Los programas que suponen una variedad de actividades o varios orígenes y destinos de entrada y salida pueden hacerse más fáciles de diseñar e implementar mediante hilos.

La planificación y la expedición se llevan a cabo con los hilos; por tanto, la mayor parte de la información de estado relacionada con la ejecución se mantiene en estructuras de datos al nivel de los hilos. Sin embargo, hay varias acciones que afectan a todos lo hilos de una tarea y que el sistema operativo debe gestionar al nivel de las tareas. La suspensión implica la descarga del espacio de direcciones fuera de la memoria principal. Puesto que todos los hilos de una tarea comparten el mismo espacio de direcciones, todos deben entrar en el estado Suspendido al mismo tiempo. De manera similar, la terminación de una tarea supone terminar con todos los hilos dentro de dicha tarea.

***Ejemplo* — *Aldus PageMaker***

Como ejemplo del uso de los hilos, considérese la aplicación *Aldus PageMaker,* que se ejecuta para OS/2. PageMaker es una herramienta de composición, diseño y producción de pu-blicaciones. La estructura de hilos, que se muestra en la figura 3.15 RON9O], se escogió para optimizar el grado de respuesta de la aplicación. Siempre hay tres hilos activos: un hilo de gestión de sucesos, un hilo para el dibujo de la pantalla y un hilo de servicio.

En general, OS/2 responde peor en la gestión de las ventanas si hay algún mensaje de entrada que necesita mucho procesamiento. Los principios generales de OS/2 establecen que ningún mensaje puede necesitar más de 1/10 de segundo. Por ejemplo, llamar a una subrutina para imprimir una página mientras se está procesando una orden de impresión podría impedir que el sistema atendiera a algún mensaje posterior para alguna aplicación, lo que ralentiza el rendimiento. Para satisfacer este criterio, las operaciones largas del usuario con PageMaker, como la impresión, la importación de datos y el flujo de textos, se llevan a cabo mediante un hilo de servicio. La inicialización del programa también se lleva a cabo con un

*Digitalización con propósito académico*

*Sistemas Operativos*

**138 Descripción y control de procesos**

**FIGURA 3.15 Estructura de hilos de Aldus PageMaker**

hilo de servicio, que aprovecha el tiempo libre mientras que el usuario invoca un dialogo para la creación de un nuevo documento o para abrir un documento existente. *Un* hilo aparte espera nuevos mensajes de sucesos.

La sincronización entre el hilo de servicio y el hilo de gestión de sucesos es complicada porque el usuario puede seguir tecleando o moviendo el ratón, lo que activa el hilo de ge- stión de sucesos, mientras que el hilo de servicios se mantiene ocupado. Si este conflicto se produce, PageMaker filtra estos mensajes y acepta solo algunos básicos, como cambiar el tamaño de las ventanas.

El hilo de servicio envía un mensaje para indicar que ha terminado su tarea. Hasta que esto ocurra, la actividad del usuario en PageMaker está limitada. El programa indica esta restricción inhabilitando los elementos de los menús y visualizando un cursor de “ocupado”. El usuario es libre de cambiarse a otra aplicación y, cuando el cursor “ocupado” se traslada a otra ventana, se cambiará por el cursor adecuado para la aplicación.

Se utiliza un hilo independiente para dibujar la pantalla por dos razones:

1. PageMaker no limita el número de objetos que pueden aparecer en una página; por tanto, procesar una petición de dibujo puede exceder fácilmente de la pauta de 1/10 sg.

2. Utilizar un hilo separado permite que el usuario abandone el dibujo. En este caso, cuando el usuario cambia la escala de una página, ésta puede dibujarse de inmediato. El programa es menos sensible si termina visualizando la página en la escala anterior y después la vuelve a dibujar por completo con la nueva escala.

El desplazamiento dinámico *(dynamic scrolling)* —volver a dibujar la pantalla a medida que el usuario mueve el indicador de desplazamiento— también es posible. El hilo de gestión de sucesos supervisa la barra de desplazamiento y vuelve a dibujar las regias de los márgenes (que se vuelven a dibujar rápidamente, ofreciéndole al usuario una sensación de respuesta inmediata). Mientras tanto, el hilo de dibujo de la pantalla trata constantemente de volver a dibujar la pantalla y así ponerse al corriente.

Implementar el dibujo dinámico sin el uso de varios hilos impone una gran sobrecarga a la aplicación, al obligarla a sondear los mensajes en varios puntos. Los múltiples hilos permiten que se puedan separar las actividades concurrentes en el código de una manera más natural.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Procesos e hilos 139**

**Otras estructuras**

Como se ha dicho, los conceptos de unidad de asignación de recursos y de unidad de expedición se han englobado tradicionalmente dentro del concepto único de proceso, es decir, como una relación de uno a uno entre hilos y procesos. Recientemente, ha habido un gran interés en disponer de varios hilos dentro de un mismo proceso, es decir, una relación de muchos a uno. Sin embargo, como se muestra en la tabla 3.12, también se han investigado las otras dos combinaciones, las llamadas relaciones de muchos a muchos y las relaciones de uno a muchos.

*Relación de muchos a muchos*

La idea de tener una relación muchos a muchos entre los kilos y los procesos se ha explorado en el sistema operativo experimental TRIX [SIEB83, WARD8O]. En TRIX, se tienen los conceptos de dominio y de hilo. Un dominio es una entidad estática que consta de un espacio de direcciones y de unos “puertos” a través de los cuales se envían y reciben los mensajes. Un hilo es un camino sencillo de ejecución, con una pila de ejecución, el estado del procesador y la información de planificación.

Al igual que Mach, se pueden ejecutar varios hilos en un solo dominio, lo que aporta el aumento de eficiencia discutido anteriormente. Sin embargo, también es posible que una ac-tividad de un solo usuario o aplicación se pueda llevar a cabo en varios dominios. En tal caso, existirá un hilo que se puede mover de un dominio a otro.

El uso de un mismo hilo en varios dominios parece motivado, más que nada, por el deseo de brindar al programador una herramienta de estructuración. Por ejemplo, considérese un programa que utiliza un subprograma de E/S. En un entorno de multiprogramación que permita la generación de procesos de usuario, el programa principal puede generar un nuevo proceso que se encargue de la E/S y luego continúe ejecutándose. Sin embargo, si el progreso futuro del programa principal depende del resultado de la operación de E/S, el programa principal tendrá que esperar a que termine el programa de E/S. Hay varias formas de implementar esta aplicación:

1. El programa entero se puede implementar como un único proceso. Esta es una solución razonable y sencilla. Hay algunos inconvenientes relativos a la gestión de memoria. El proceso global puede exigir un volumen considerable de memoria principal para ejecutar eficientemente, mientras que el subprograma de E/S requiere un espacio

**TABLA 3.12 Relación Entre Hilos y Procesos**

Hilos:Procesos Descripción Sistemas de Ejemplo

1:1 Cada hilo de ejecución es un 6nico proceso UNIX System V

con sus propios recursos y espacio de direcciones.

M:1 Un proceso define un espacio de direcciones OS/2, MVS, MACH

y recursos dinámicos propios. Pueden crearse

varios hilos que ejecuten en dicho proceso.

1:M Un hilo puede emigrar del entorno de un proceso Ra

a otro. Esto permite que un hilo se pueda mover

fácilmente entre sistemas distintos.

M:M Combina los atributos de los casos M:1 y 1 :M TRIX

*Digitalización con propósito académico*

*Sistemas Operativos*

**140 Descripción y control de procesos**

de direcciones relativamente pequeño para el buffer de E/S y para manejar la pequeña cantidad de código del programa. Puesto que el programa de *E/S* se ejecuta en el espacio de direcciones del programa mayor, o bien el proceso entero debe permanecer en memoria principal durante la operación de E/S o bien la operación de E/S está sujeta al intercambio. Este efecto en la gestión de memoria también se produce si el programa principal y el subprograma de E/S se implementaran como hilos del mismo espacio de direcciones.

2. El programa principal y el subprograma de E/S pueden implementarse como dos procesos separados. Esto incurre en la sobrecarga de crear el proceso subordinado. Si las actividades de *E/S* son frecuentes, se puede dejar vivo al proceso subordinado, lo que consume recursos de gestión o bien crear y destruir frecuentemente el subprograma, lo que es ineficiente.

3. Tratar al programa principal y al subprograma de E/S como una actividad única que se debe implementar como un único hilo. Sin embargo, puede crearse un espacio de di-recciones (dominio) para el programa principal y otro para el subprograma de E/S. Así pues, el hilo puede moverse entre los dos espacios de direcciones a medida que avanza la ejecución. El sistema operativo puede administrar los dos espacios de direcciones independientemente, sin incurrir en sobrecarga alguna de creación de procesos. Más aún, el espacio de direcciones usado por el subprograma de E/S puede estar compartido también por otros subprogramas sencillos de E/S.

La experiencia de los desarrolladores de TRIX observa el mérito de la tercera opción y demuestra que ésta puede ser la solución más eficaz para algunas aplicaciones.

***Relación de uno a muchos***

En el campo de los sistemas operativos distribuidos (diseñados para controlar sistemas in-formáticos distribuidos), ha habido un gran interés en el concepto de hilo, principalmente como una entidad que se puede mover entre los espacios de direcciones.9 Un ejemplo notable de esta investigación es el sistema operativo Clouds [DASG88] y espacialmente en su núcleo, conocido como Ra [ERN89]. Otro ejemplo es el sistema Emerald UL88].

Un hilo en Clouds es una unidad de actividad desde la perspectiva del usuario. Un proceso es un espacio de direcciones virtuales con un bloque de control de proceso asociado. Al crearse, un hilo comienza a ejecutar en un proceso invocando un punto de entrada a un programa de dicho proceso. Los hilos se pueden mover de un espacio de direcciones a otro, atravesando de hecho las fronteras de una máquina (es decir, se mueven de un computador a otra). Al trasladarse, un hilo debe llevar consigo cierta información, tal como el terminal de control, unos parámetros globales y las guías de planificación (por ejemplo, la prioridad).

El enfoque de Clouds ofrece una forma efectiva de aislar al usuario y al programador de los detalles del entorno distribuido. Una actividad del usuario puede estar representada por un solo hilo y el movimiento de dicho hilo entre las máquinas puede estar dictado por el sistema operativo, debido a razones propias del sistema, tales como la necesidad de acceder a un recurso remoto o equilibrar la carga.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

9 El traslado de los procesos o hilos entre espacios de direcciones de máquinas diferentes se ha convertido en un tema candente en los últimos años (véase, por ejemplo, [RTS89a,b]). Este tema se explorará en el capitulo 13.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Ejemplos de descripción y control de procesos 141**

**3.5**

**EJEMPLOS DE DESCRIPCION Y CONTROL DE PROCESOS**

**Sistema UNIX, versión V**

UNIX utiliza un servicio de procesos simple pero potente, que es muy visible para el usuario. Todos los procesos del sistema, excepto dos procesos básicos, son creados por órdenes de programas del usuario.

*Estados de un proceso*

Un total de nueve estados de proceso son los reconocidos por el sistema operativo UNIX; éstos están reflejados en La tabla 3.13 y un diagrama de transición de estados se muestra en La figura 3.16. Esta figura es bastante similar ala figura 3.7, con los dos estados de Dormido de UNIX correspondientes a los dos estados de Bloqueado. Las diferencias pueden resumirse rápidamente a continuación:

• UNIX emplea dos estados de Ejecución, que indican si el proceso está ejecutando en modo de usuario o en modo núcleo.

• Se hace una distinción entre los estados: Listo para Ejecutar y en Memoria, frente al estado de Expulsado. Estos dos estados son, básicamente, el mismo, como se indica por la línea de puntos que los une. Se hace la distinción para enfatizar la forma en que se pasa al estado Expulsado. Cuando un proceso esté ejecutándose en modo núcleo (como resultado de una llamada del supervisor, una interrupción de reloj, o una interrupción de E/S), llegara un momento en el que el núcleo termine su trabajo y esté listo para devolver el control al programa de usuario. En este punto, el núcleo puede decidir expulsar el proceso en curso en favor de alguno que esté listo y tenga mayor prioridad. En tal caso, el proceso en curso se pasa al

**TABLA 3.13 Estados de un Proceso en UNIX**

Ejecución en modo de usuario Ejecutando en modo usuario.

Ejecución en modo del núcleo Ejecutando en modo del núcleo.

Listo para ejecutar y en memoria Listo para ejecutar tan pronto como el núcleo lo planifique.

Dormido y en memoria No dispuesto para ejecutar hasta que se produzca un suceso; el proceso está en memoria principal.

Listo para ejecutar y descargado El proceso está listo para ejecutar, pero se debe cargar el

proceso en memoria principal antes de que el núcleo pueda

planificarlo para su ejecución.

Dormido y descargado El proceso está esperando un suceso y ha sido expulsado al almacenamiento secundario.

Expulsado El proceso retorna del modo núcleo al modo usuario pero el n6cleo lo expulsa y realiza un cambio de contexto para

para planificar otro proceso.

Creado El proceso está recién creado y aún no está aún listo para ejecutar.

ombie El proceso ya no existe, pero deja un registro para que lo recoja el proceso padre.

Z

*Digitalización con propósito académico*

*Sistemas Operativos*

**142 Descripción y control de procesos**

**FIGURA 3.16 Diagrama de transición de estados de los procesos en UNIX [ACH86]**

estado Expulsado. Sin embargo, a efectos de la expedición, aquellos procesos que están en estado Expulsado y aquellos que están en Listos para Ejecutar y en Memoria forman una sola cola.

La expulsión puede producirse solo cuando va a moverse un proceso del modo núcleo al modo usuario. Mientras un proceso esté ejecutándose en modo núcleo no puede ser expulsado. Esto hace que UNIX sea inadecuado para el procesamiento en tiempo real. El capitulo 9 discute los requisitos del procesamiento en tiempo real.

Hay dos procesos que son únicos en UNIX. El proceso 0 que se crea cuando el sistema arranca; en realidad, es una estructura de datos predefinida que se carga en el momento del arranque y que se denomina proceso de intercambio. Además, el proceso 0 genera el proceso 1, conocido como proceso *init;* todos los demás procesos del sistema tienen al proceso 1 como antepasado. Cuando se conecta un nuevo usuario interactivo en el sistema, es el proceso 1 el que crea un nuevo proceso para dicho usuario. A continuación, el programa de usuario puede crear procesos hijos en forma de árbol, de forma que una aplicación en particular puede constar de un conjunto de procesos relacionados.

*Descripción de procesos*

Un proceso en UNIX es un conjunto más bien complejo de estructuras de datos que propor-cionan al sistema operativo toda la información necesaria para administrarlos y expedirlos.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Ejemplos de descripción y control de procesos 143**

La tabla 3.14 resume los elementos de la imagen de un proceso, los cuales se organizan en tres partes: contexto del usuario, contexto de los registros y contexto del sistema.

El contexto del usuario contiene los elementos básicos de un programa de usuario y puede ser generado directamente a partir un archivo de código compilado. El programa del usuario se separa en zonas de texto y datos; la zona de texto es solo de lectura y está destinada a albergar las instrucciones del programa. Mientras que el proceso está ejecutando, el procesador utiliza la zona de pila del usuario para las llamadas a procedimientos, los retornos y el paso de parámetros. La región de memoria compartida es una zona de datos que es compartida con otros procesos. Hay una sola copia física de una región de memoria compartida, pero gracias a la memoria virtual, a cada proceso le parece que esta región compartida está en su propio espacio de direcciones.

Cuando un proceso no está ejecutándose, la información del estado del procesador se al-macena en la zona de contexto de los registros.

Por último, el contexto del sistema contiene la información restante que el sistema operativo necesita para administrar el proceso. Está formada por una parte estática, que tiene un tamaño fijo y permanece asociada a un proceso durante toda su existencia y una parte dinámica que varia de tamaño durante la vida del proceso. Un elemento de la parte estática es la entrada de la tabla de procesos. Esta es, en realidad, una parte de la tabla de procesos que

**TABLA 3.14 Imagen de un proceso en UNIX**

Contexto del Usuario

Texto del Proceso Instrucciones de máquina ejecutables del programa.

Datos del proceso Datos del proceso accesibles por el programa.

Pila del usuario Contiene los argumentos, las variables locales, y los punteros de las

funciones que ejecutan en modo usuario.

Memoria compartida Memoria compartida con otros procesos, utilizada para la comunicación

interproceso.

Contexto de los Registros

Contador de programa Dirección de la próxima instrucción a ejecutar; puede estar

en el espacio de memoria del núcleo o del usuario, de este proceso.

Registro de estado Contiene el estado del hardware en el momento de la expulsión;

del procesador el formato y el contenido dependen del hardware.

Puntero de pila Señala a la cima de la pila del núcleo o de la pila del usuario,

dependiendo del modo de operación en el momento de a expulsión.

Registros de propósito Dependientes del hardware.

general

Contexto del Sistema

Entrada de la tabla Define el estado de un proceso; esta información está siempre

de procesos accesible para el sistema operativo.

Área U (de usuario) La información de control del proceso a la que se necesita acceder solo

en el contexto del proceso.

Tabla de regiones Define una traducción de direcciones virtuales a físicas; también contiene

de preproceso un campo de permiso que indica el tipo de acceso permitido para el proceso: solo lectura, lectura/escritura o lectura/ejecución.

Pila del núcleo Contiene los marcos de pila de los procedimientos del núcleo cuando

el proceso ejecuta en modo del núcleo.

*Digitalización con propósito académico*

*Sistemas Operativos*

**144 Descripción y control de procesos**

mantiene el sistema operativo, que dispone de una entrada por proceso. La entrada de la tabla de procesos contiene la información de control accesible para el núcleo en todo momento; de haí que en un sistema de memoria virtual todas las entradas de la tabla de procesos se mantengan en memoria principal. La tabla 3.15 enumera los contenidos de una entrada de La tabla de procesos. La zona de usuario o zona U, contiene información de control adicional del proceso que es necesaria para el núcleo solo cuando está ejecutando en el contexto del proceso. La tabla 3.16 muestra el contenido de esta tabla.

La distinción entre la entrada de la tabla de procesos y la zona U refleja el hecho de que el núcleo de UNIX siempre ejecuta en el contexto de algún proceso. Durante una gran parte del tiempo, el núcleo tendrá que estar tratando con todo lo relativo al proceso. Sin embargo, durante algún tiempo, como cuando el núcleo está ejecutando un algoritmo de planificación para preparar la expedición de otro proceso, el núcleo necesitará acceder a la información de otros procesos.

La tercera parte estática del contexto del sistema es la tabla de regiones de los procesos, que es utilizada por el sistema gestor de memoria. Por último, La pila núcleo es La parte dinámica del contexto del sistema. Esta pila se utiliza cuando el proceso está ejecutando en el modo núcleo. Contiene la información que se debe salvar y restaurar cada vez que se producen interrupciones y llamadas a procedimientos.

***Control de procesos***

Como ya se mencionó, el Sistema UNIX, versión v, sigue el modelo de la figura 3.13b, en el que la mayor parte del sistema operativo ejecuta en el entorno de un proceso de usuario.

**TABLA 3.15 Entradas de la labia de Procesos en UNIX**

Estado del Proceso Estado actual del proceso.

Punteros Al área U y a la zona de memoria del proceso (texto, datos, pila).

Tamaño del proceso Permite que el sistema operativo sepa cuánto espacio *asignar* al proceso.

Identificadores El ID real de usuario identifica al usuario que es responsable del proceso que de usuario se está ejecutando. El ID efectivo de usuario sirve para que un proceso disponga temporalmente de los privilegios asociados a un programa particular; mientras se ejecuta el programa como parte del proceso, éste opera con el ID efectivo de usuario.

Identificadores ID del proceso; ID del proceso padre.

de proceso

Descriptor de suceso Válido cuando el proceso está en un estado dormido cuando se produce el

suceso, el proceso pasa al estado de listo para ejecutar.

Prioridad Empleada en la planificación del proceso.

Señal Enumera las señales enviadas a un proceso y que aún no se han tratado.

Temporizadores Incluye el tiempo de ejecución del proceso, el uso de los recursos

del núcleo y un temporizador de usuario empleado para enviar señales

de alarma al proceso.

Enlace-P Puntero al siguiente enlace en la cola de listos (valido si el proceso está listo para ejecutarse).

Estado de la memoria Indica si la imagen del proceso está en memoria principal o se ha decargado

al disco. Si está en memoria, este campo también indica si el proceso se

puede descargar o si está temporalmente bloqueado en memoria principal.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Ejemplos de descripción y control de procesos 145**

**TABLA 3.16 Área U de UNIX**

Puntero a la Tabla de Procesos Indica la entrada que corresponde al área U.

Identificadores de usuario Los IDs de usuario reales y efectivos.

Temporizadores Registro del tiempo que el proceso (y sus descendientes) dedicaron ejecutando en modo usuario y en modo núcleo.

Vector de tratamiento de señales Para cada tipo de señal definida en el sistema, indica cOmo

reaccionará el proceso al recibirla (terminar, ignorar, ejecutar una función especificada por el usuario).

Terminal de control Indica el terminal de conexión para el proceso (si procede).

Campo de error Registra los errores producidos durante una llamada al sistema.

Valor de retorno Contiene el resultado de las llamadas al sistema.

Parámetros de E/S Describen la cantidad de datos a transferir, la dirección origen

(o destino) del vector de datos del espacio de usuario y los

desplazamientos en los archivos para la EJS.

Parámetros de archivos El directorio actual y el directorio raíz actual describen el entorno del sistema de archivos para el proceso.

Tabla de descriptores Registra los archivos que el proceso tiene abiertos.

de los archivos del usuario

Campos de limites Restringe el tamaño del proceso y el tamaño de un archivo en que éste puede escribir.

Campos de modos de protección Mascara con los modos de protección de los archivos creados por el proceso.

Por tanto, hacen falta dos modos, el de usuario y el del núcleo. Algunas partes del sistema operativo, en especial el proceso de intercambio *(swapper),* operan como procesos separados y se les conoce como *procesos del núcleo.*

La creación de procesos en UNTX se hace por medio de la llamada *fork* al núcleo del sis-tema. Cuando un proceso emite una petición de *fork,* el sistema operativo realiza las siguientes operaciones ACH86J:

1. Asigna una entrada en la tabla de procesos para el nuevo proceso.

2. Asigna un ID único de proceso al proceso hijo.

3. Hace una copia de La imagen del proceso padre, a excepción de la memoria compartida.

4. Incrementa los contadores de los archivos que son propiedad del padre, para reflejar el hecho que hay un nuevo proceso ahora que también es propietario de esos archivos.

*5.* Pone al proceso hijo en el estado Listo para Ejecutar.

6. Devuelve al proceso padre el número de ID del hijo y devuelve un valor cero al proceso hijo.

Todo este trabajo se acomete en el modo núcleo en el proceso padre. Cuando el núcleo haya terminado con estas funciones, podrá realizar alguna de las siguientes como parte de la rutina distribuidora:

1. Permanecer en el proceso padre. El control vuelve al modo de usuario en el punto de la llamada fork del padre.

2. Transferir eL control al proceso hijo. El proceso hijo comienza ejecutando en el mismo punto del código que el padre, es decir, en el punto de retomo de la llamada fork.

*Digitalización con propósito académico*

*Sistemas Operativos*

**146 Descripción y control de procesos**

3. Transferir el control a otro proceso. Tanto el padre como el hijo se dejan en el estado Listo para Ejecutar.

Puede que sea difícil hacer ver este método de creación de procesos, porque tanto el padre como el hijo están ejecutando el mismo trozo de código. La diferencia es la siguiente:

Cuando se retorna del *fork,* se pregunta por el parámetro de retomo. Si el valor es cero, en-tonces se está en el proceso hijo y se puede ejecutar una bifurcación hacia el programa de usuario apropiado para continuar la ejecución. Si el valor es diferente de cero, entonces se está en el proceso padre y puede continuar la línea principal de ejecución.

**WINDOWS NT**

El diseño de los procesos de Windows NT está dirigido por la necesidad de dar soporte a varios entornos de sistemas operativos. Los procesos aportados por los distintos sistemas operativos son diferentes en varios aspectos, incluyendo los siguientes:

• Cómo se les denomina a los procesos

• Si hay hilos disponibles dentro de los procesos

• Cómo se representan los procesos

• Cómo se protegen los recursos de los procesos

• Qué mecanismos se emplean para la comunicación y la sincronización entre procesos

• Cómo están relacionados los procesos entre sí

Por consiguiente, la estructura nativa de los procesos y de los servicios que brinda el núcleo de NT es relativamente simple y de propósito general, permitiendo a cada subsistema emular la estructura y la funcionalidad particular de los procesos de un sistema operativo. Las características más importantes de los procesos de NT son las siguientes:

• Los procesos de NT se implementan como objetos.

• Un proceso ejecutable puede tener uno o más hilos.

• Los objetos proceso y los objetos hilo tienen capacidades predefinidas de sincronización.

• El núcleo de NT no conserva ninguna relación entre los procesos que crea, incluyendo las relaciones padre-hijo.

La figura 3.17 ilustra la forma en que un proceso se refiere a los recursos que controla o utiliza. La señal de acceso *(access token),* descrita en el capitulo 2, controla si el proceso puede cambiar sus propios atributos. En tal caso, el proceso no tendrá un descriptor abierto de su señal de acceso: Si el proceso intenta abrir un descriptor, el sistema de seguridad determinará si se le permite y, por tanto, si el proceso puede cambiar sus propios atributos.

También tienen que ver con el proceso una serie de bloques que definen el espacio de di-recciones virtuales asignado. El proceso no puede modificar directamente estas estructuras, sino que debe depender del administrador de memoria virtual, quien le proporciona al proceso un servicio de asignación de memoria.

Finalmente, el proceso incorpora una tabla de objetos, con los descriptores de otros objetos que conoce. Existe un descriptor para cada hilo del proceso. En la figura se muestra un solo hilo. Además, el proceso tiene acceso a un objeto archivo y a un objeto sección que define una sección de memoria compartida.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Ejemplos de descripción y control de procesos 147**

**FIGURA 3.17 Un proceso de NT y sus recursos**

*Objetos proceso y objetos hilo*

La estructura orientada a objetos de NT facilita el desarrollo de un servicio de procesos de propósito general. NT hace uso de dos tipos de objetos relacionados con los procesos: los procesos y los hilos. Un proceso es una entidad correspondiente a un trabajo de usuario o una aplicación, que dispone de sus propios recursos, tales como memoria y archivos abiertos. Un hilo es una unidad de trabajo que se puede expedir para su ejecución secuencial y que es interrumpible, de forma que el procesador pueda pasar de un hilo a otro.

Cada proceso de NT está representado por un objeto, cuya estructura general se muestra en la figura 3.1 8a. Cada proceso queda definido por una serie de atributos y encapsula una serie de acciones o servicios que puede desempeñar. Un proceso realizará un servicio cada vez que reciba un mensaje apropiado; la única manera de invocar a dichos servicios es por medio de mensajes al objeto proceso que ofrece el servicio.

Cada vez que NT crea un proceso nuevo, utiliza la clase de objeto o tipo de objeto definido para los procesos de NT como plantilla para generar nuevos casos o instancias de objetos. En el momento de la creación, se asignan unos valores a los atributos. La tabla 3.17 da una definición breve de cada uno de los atributos de un objeto proceso.

Un proceso de NT debe tener, al menos, un hilo para ejecutar. Dicho hilo puede crear en-tonces otros hilos: En un sistema multiprocesador, pueden ejecutarse en paralelo varios hi-los de un mismo proceso.

La figura 3.1 8b representa la estructura de objeto para un objeto hilo y la tabla 3.18 define los atributos de un objeto hilo: nótese que alguno de los atributos de un hilo se parecen a los de un proceso. En tales casos, el valor del atributo del hilo se obtiene a partir del valor del atributo del proceso. Por ejemplo, la afinidad que tiene el hilo con los procesadores es el conjunto de procesadores de un sistema multiprocesador que pueden ejecutar al hilo; este conjunto es igual o es un subconjunto de la afinidad que tiene el proceso con los procesadores.

*Digitalización con propósito académico*

*Sistemas Operativos*

**148 Descripción y control de procesos**

*Digitalización con propósito académico*

*Sistemas Operativos*

**Ejemplos de descripción y control de procesos 149**

**TAB LA 3.17 Atributos de un Objeto Proceso en Windows NT**

ID de proceso Un valor único que identifica al proceso ante el sistema operativo.

Señal de acceso Un objeto del ejecutor que contiene información de seguridad sobre el usuario conectado al sistema y representado por el proceso.

Prioridad de base Prioridad de partida para los hilos del proceso.

Afinidad por omisión Conjunto de procesadores por omisión en los cuales se puede ejecutarse los hilos del proceso.

Limites de cuota La cantidad máxima de memoria paginada y no paginada del sistema,

el espacio para los archivos de paginación y el tiempo de procesador que un proceso de usuario puede emplear.

Tiempo de ejecución La cantidad total de tiempo que todos los hilos del proceso han consumido en ejecución

Contadores de E/S Variables que registran el número y el tipo de las operaciones de E/S que han llevado a cabo los hilos del proceso.

Contadores de Variables que registran el número y el tipo de las operaciones de memoria operaciones de MV virtual que los hilos del proceso han realizado.

Puertos de Canales de comunicación entre procesos a los que el administrador de procesos excepción/depuración envía mensajes cuando uno de los hilos del proceso provoca una excepción.

Estado de terminación La razón de la terminación de un proceso.

Nótese que uno de los atributos del objeto hilo es el contexto. Esta información permite que los hilos puedan ser suspendidos y reanudados. Además, se puede alterar el comportamiento de un hilo modificando su contexto cuando esté suspendido.

***Hilos múltiples***

NT da soporte para la concurrencia entre procesos, pues los hilos de procesos diferentes pueden ejecutarse concurrentemente. Es más, pueden asignarse varios hilos del mismo proceso a distintos procesadores y así ejecutarse concurrentemente. Un proceso con hilos múltiples puede lograr la concurrencia sin la sobrecarga del empleo de varios procesos. Los hilos de un mismo proceso pueden intercambiar información a través de la memoria compartida y tener acceso a los recursos compartidos del proceso.

Un proceso orientado a objetos con hilos múltiples es una forma eficiente de construir aplicaciones servidoras. La figura 3.19 describe el concepto general. Un único proceso servidor puede dar servicio a varios clientes. Cada demanda de un cliente provoca la creación de un nuevo hilo en el servidor.

***Soporte para subsistemas***

Los servicios generales de procesos e hilos deben dar soporte a las estructuras de hilos y procesos particulares de varios clientes del SO. Es responsabilidad de cada subsistema el aprovechar las características de los procesos e hilos de NT para emular los servicios de pro-cesos e hilos del sistema operativo correspondiente.

Observando cómo es la creación de un proceso, se puede esbozar una idea de cómo se proporciona soporte para hilos y procesos. La creación de un proceso comienza con la petición de un proceso nuevo desde una aplicación del SO. La petición de crear un proceso se emite desde una aplicación hacia el subsistema protegido correspondiente. El subsistema, a su vez, emite una petición de un nuevo proceso al ejecutor de NT. NT crea un objeto proceso

*Digitalización con propósito académico*

*Sistemas Operativos*

**150 Descripción y control de procesos**

**TABLA 3.18 Atributos de un Objeto Hilo en Windows NT**

ID del cliente Valor único que identifica a un hilo cuando llama al servidor.

Contexto hilo El conjunto de valores de registros y otros datos volátiles que definen el estado de ejecución de un hilo.

Prioridad dinámica La prioridad de ejecución del hilo en un momento dado.

Prioridad de base El limite inferior de la prioridad dinámica del hilo.

Afinidad del hilo El conjunto de procesadores en los que puede ejecutar el hilo, que es con el procesador un subconjunto o coincide con la afinidad con el procesador del proceso que contiene el hilo.

Tiempo de ejecución El tiempo acumulado que el hilo ha ejecutado en modo usuario y en modo del hilo núcleo.

Estado de alerta Un indicador que dice si el hilo debe ejecutar una llamada a un procedimiento asíncrono.

Contador El número de veces que la ejecución del hilo se ha suspendido sin reanudarse de suspensión.

Señal de imitación Una señal de acceso temporal que permite a un hilo realizar operaciones en

nombre de otro proceso (utilizado por los subsistemas).

Puerto de terminación Un canal de comunicación entre procesos al que el administrador de procesos envía un mensaje cuando el hilo termina (utilizado por los subsistemas).

Estado de terminación La razón de terminación del hilo.

y le devuelve un descriptor de dicho objeto al subsistema. Ahora, cuando NT crea un proceso, no crea automáticamente un hilo. En los casos de Win32 y OS/2, siempre se crea un hilo junto con el nuevo proceso. Por tanto, para estos sistemas operativos, el subsistema llamará de nuevo al administrador de procesos de NT para crear un hilo para el nuevo proceso, recibiendo de NT el descriptor del hilo. La información correspondiente sobre el hilo y el

**FIGURA 3.19 Servidor de hilos múltiples en Windows NT**

*Digitalización con propósito académico*

*Sistemas Operativos*

**Ejemplos de descripción y control de procesos 151**

proceso es devuelta a la aplicación. Windows de 16 bits y POSIX no tienen soporte de hilos. Por tanto, para estos sistemas operativos el subsistema obtiene de NT un hilo para el nuevo proceso, de forma que se pueda activar el proceso, pero solo se devuelve a la aplicación la información del proceso. El hecho de que el proceso de la aplicación se implemente mediante un hilo no es visible para la aplicación.

Cuando se crea un proceso nuevo en Win32 o en OS/2, éste hereda muchos de sus atributos del proceso creador. Sin embargo, en el entorno de NT, esta creación del proceso se hace indirectamente. Un proceso cliente de aplicación envía su petición de crear un proceso al subsistema del SO; después, un proceso del subsistema enviará a su vez una solicitud de creación de proceso al ejecutor de NT. Puesto que el efecto deseado es que el nuevo proceso herede las características del proceso cliente y no las del proceso servidor, NT permite que el subsistema especifique el padre del nuevo proceso. El nuevo proceso hereda entonces la señal de acceso del padre, los limites de cuota, la prioridad de base y la afinidad por omisión con los procesadores.

**MVS**

*Tareas de MVS*

El sistema operativo MYS emplea un entorno de procesos muy estructurado. En MYS, la me-moria virtual se divide en grandes regiones llamadas *espacios de direcciones.* A grandes rasgos, a cada aplicación le corresponde un solo espacio de direcciones. Dentro del espacio de direc-ciones, se pueden generar varias tareas. La tarea de MVS constituye la unidad de expedición.

Normalmente, existen a la vez más de un centenar de espacios de direcciones, cada uno de los cuales puede tener muchas tareas. De este modo, el MVS puede manejar y, a menudo, maneja miles de tareas concurrentemente.

La figura 3.20, basada en otra de Leban y Arnold [LEBA84a], ofrece un ejemplo de la forma en la que se generan varias tareas en un mismo espacio de direcciones. La aplicación da soporte para transacciones de consulta desde una serie de terminales. El programa está dividido en cuatro módulos, un programa principal y tres programas para las tres clases de consulta: de clientes, de pedidos y de productos. Cuando se carga la aplicación en el sistema, el espacio de direcciones comienza con una parte de la memoria reservada para MVS y con el programa principal cargado (figura 3.20a). Después, un usuario introduce una consulta de clientes desde un terminal. Como resultado, el programa principal realiza una solicitud de creación de una nueva tarea para ejecutar el módulo cliente y éste se carga en el espacio de direcciones (figura 3.20b). Mientras esta transacción está en curso, un segundo terminal realiza un pedido y se carga el modulo correspondiente como una tarea separada (figura 3.20c). A continuación, mientras ambas transacciones siguen en curso, un tercer terminal realiza otro pedido. El código del modulo solicitado ya está en el espacio de direcciones y puede ser utilizado por más de una tarea. Sin embargo, cada tarea necesita su propia región privada de memoria para las variables locales y el tratamiento de la pita, como se ilustra en la figura 3.20d.

Explícitamente, el MVS hace uso de solo tres estados para una tarea: Listo, Activo (en ejecución) y Esperando. Sin embargo, se puede descargar a memoria secundaria un espacio de direcciones entero. Por tanto, según se ha venido usando el término, todas las tareas de dicho espacio de direcciones van a quedar suspendidas.

*Digitalización con propósito académico*

*Sistemas Operativos*

**152 Descripción y control de procesos**

F**IGURA 3.20 Esquema del espacio de dirección para una aplicación MVS**

Cada tarea está representada por un bloque de control de tarea (TCB, *Task Control Block).* Además, existe otro tipo de entidad que se puede expedir, la petición de servicio. La *petición de servicio* es una “tarea del sistema”, de la que existen dos clases. Algunas tareas del sistema ofrecen servicios a una aplicación específica y se ejecutan en el espacio de direcciones de dicha aplicación; otras están involucradas en las operaciones realizadas entre espacios de direcciones y no ejecutan en ningún espacio de direcciones de usuario, sino en el espacio de direcciones reservado para el sistema. Estas tareas pueden considerarse en conjunto como el núcleo del sistema operativo MVS. Cada tarea del sistema tiene su propio bloque de petición de servicio (SRB, *Service Request Block)* y, cuando llega el momento de expedir una nueva tarea, el distribuidor escoge de entre todos los SRBs y TCBs que estén listos.

De esta forma, en MVS se pueden hallar, a su manera, dos conceptos aparentemente tan modernos como son el de hilos dentro de un proceso y el de contemplar al sistema operativo como un conjunto de procesos (figura 3. 13c).

*Estructuras de control de tareas*

La tabla 3.19 enumera las estructuras principales de control que utiliza el MYS para administrar las tareas de usuario y las tareas del sistema. En conjunto, estas estructuras agrupan gran parte de lo que se ha venido en llamar bloque de control de proceso. Los bloques globales de petición de servicio se usan para administrar las tareas del sistema que no ejecutan en un espacio de direcciones de usuario. Las estructuras restantes tienen que ver con la gestión del espacio de direcciones. El bloque de control de espacio de direcciones (ASCB, *Address Space Control Block) y* el bloque ampliado de espacio de direcciones (ASXB, *Address Space eXtension Block)* contienen información sobre los espacios de direcciones. La distinción es que MVS necesita la información del ASCB cuando está ejecutando en el espacio de direcciones reservado del sistema, mientras que el ASXB contiene información adicional necesaria solo cuando MVS está tratando con un espacio de direcciones específico de un usuario. Por último, los SRBs y TCBs locales contienen información sobre las entidades que se pueden expedir en un espacio de direcciones.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Ejemplos de descripción y control de procesos 153**

La figura 3.21 muestra como se organizan estas estructuras y propone la disciplina de expe-dición aplicada en MVS. Los SRBs globales se mantienen en una cola de prioridades descen-dentes, en la zona de colas del sistema, que es una región de la memoria principal que no puede descargarse al disco. De este modo, los SRBs globales siempre están disponibles en la memoria principal. Cuando un procesador está disponible, MVS mira primero en esta cola en busca de algún SRB que esté Listo. Si no encuentra ninguno, entonces busca un espacio de direcciones que tenga al menos un TCB o un SRB Listo. Con tal propósito se mantiene una cola de ASCB en la zona de colas del sistema. A cada espacio de direcciones se le asigna un nivel de prioridad global y, por tanto, los ASCBs se pueden ordenar por prioridades descendentes.

**TABLA 3.19 Estructuras de Control de Procesos en MVS**

Bloque Global de Petición Representa a una tarea del sistema que es independiente de cual-

de Servicio (SRB) quier espacio de direcciones del usuario. Incluye:

• Dirección del ASCB al que atenderá la rutina

• Punto de entrada a la rutina

• Dirección de la lista de parámetros que pasar a la rutina

• Nivel de prioridad

Bloque de Control de Espacio Contiene información del sistema sobre un espacio de direc de direcciones (ASCB) ciones, es decir, la información que necesita MVS del espacio de direcciones cuando no está ejecutando en ningún espacio de

direcciones particular. La información incluye:

• Punteros a las colas de creación de ASCBs

• Si el espacio de dirección ha sido descargado a disco.

• Prioridad de expedición

• Memoria real y virtual asignada al espacio de direcciones

• Número de TCBs listos en este espacio de direcciones

• Puntero al ASXB

Bloque Ampliado de Espacio Contiene información sobre un espacio de direcciones que no es de direcciones (ASXB) de interés global. Incluye:

• N6mero de TCBs en el espacio de direcciones

• Puntero a la cola de expedición de TCB para este espacio de direcciones

• Puntero a la cola de SRB locales para el espacio de direcciones

• Puntero a la zona de salvaguarda del gestor de interrupciones

Bloque Local de direcciones Representa a una tarea del sistema que se ejecuta en un espacio (SRB) de direcciones del usuario. Incluye:

• Dirección del ASCB al que atenderá la rutina

• Punto de entrada a la rutina

• Dirección de la lista de parámetros que pasará a la rutina

• Nivel de prioridad

Bloque de Control de Tareas Representa a un programa de usuario en ejecución. Contiene (TCB) a formación necesaria para administrar una tarea en un espacio

de direcciones. Incluye:

• Información del estado del procesador

• Punteros a los programas que forman parte de la tarea.

*Digitalización con propósito académico*

*Sistemas Operativos*

**154 Descripción y control de procesos**

**FIGURA 3.21 Colas de planificación de MVS**

Una vez que se selecciona un espacio de direcciones, MVS puede trabajar con las estructuras de dicho espacio, conocidas como la zona local de colas del sistema. MVS expide a! SRB que tenga prioridad mayor y, si n**O** hay ninguno, elige al TCB de prioridad mayor. En este ejemplo se muestra la estructura de colas de los TCB para un tote de trabajos, que consta de las Siguientes tareas:

*• Tarea de Control de Regiones (RCT, Region Control Task):* Responsable de la adminis-tración del espacio de direcciones en nombre de todas las tareas que ejecutan en él.

*• Volcado (dump):* Responsable de volcar a disco el espacio de direcciones si éste termina de una forma anormal.

*• Tarea de Control Iniciada (STC, Started Control Task):* El TCB del programa que da inicio al espacio de direcciones.

*• Iniciador:* Responsable de cargar el flujo de trabajos por lotes.

*• Tarea de usuario:* La aplicación puede estar formada por una o más tareas de usuario.

La ventaja de dividir la información de control en global y local es que, si es necesario, se puede descargar al disco tanta información de un espacio de direcciones como sea posible, reservando así la memoria principal.

*Digitalización con propósito académico*

*Sistemas Operativos*

**Lecturas recomendadas 155**

**3.6**

**RESUMEN**

La piedra angular de los sistemas operativos modernos es el proceso. La función principal del sistema operativo es crear, administrar y terminar los procesos. Mientras que haya procesos activos, el sistema operativo debe velar por que se le asigne a cada uno un tiempo de ejecución en el procesador, por coordinar sus actividades, gestionar los conflictos en las solicitudes y asignar recursos del sistema a los procesos.

Para llevar a cabo las funciones de gestión de procesos, el sistema operativo debe disponer de una descripción de cada proceso. Cada proceso está representado por una imagen de proceso, que incluye el espacio de direcciones en el que ejecuta el proceso y un bloque de control del proceso. Este último contiene toda la información necesaria para que el sistema operativo administre el proceso, incluyendo su estado actual, los recursos que le han sido asignados, la prioridad y otros datos relevantes.

Durante su existencia, un proceso transita por varios estados. Los más importantes son: Listo, en Ejecución y Bloqueado. Un proceso Listo es aquel que no está ejecutándose en un momento dado, pero que está preparado para ejecutar tan pronto como el sistema operativo lo decida. Un proceso en Ejecución es aquel que está ejecutándose en el procesador. En un sistema multiprocesador puede haber más de un proceso en este estado. Un proceso bloqueado es el que está esperando a que termine algún suceso (como una operación de E/S).

Un proceso en Ejecución puede verse cortado por una interrupción, que es un suceso que se produce fuera del proceso y que es reconocido por el procesador o bien puede ser interrumpido por una llamada del supervisor al sistema operativo. En ambos casos, el procesador lleva a cabo un cambio de contexto y pasa el control a una rutina del sistema operativo:

Después de que ésta haya terminado su trabajo, el sistema operativo podrá reanudar al pro-ceso interrumpido o cambiar a otro proceso.

Algunos sistemas operativos hacen una distinción entre los conceptos de proceso e hilo. El primero se refiere ala propiedad de los recursos y el segundo se refiere ala ejecución de programas. Este enfoque puede llevar a una mejora de la eficiencia y hacer más cómoda la programación.