Lecturas 11 de Octubre

- **Descripción de procesos: Stalling pp. 108-143**
- **Descripción de procesos: Carretero pp. 80-91**
- **Concepto y tipos de Hebras (hilos) en Stalling pp. 158-172**

Descripción de procesos (Stalling 108-143)

Los SO se construyen en torno al concepto de proceso. Requisitos:

- El sistema operativo debe intercalar la ejecución de múltiples procesos, para maximizar la utilización del procesador mientras se proporciona un tiempo de respuesta razonable.
- El sistema operativo debe reservar recursos para los procesos conforme a una política específica (por ejemplo, ciertas funciones o aplicaciones son de mayor prioridad) mientras que al mismo tiempo evita interbloqueos.
- Un sistema operativo puede requerir dar soporte a la comunicación entre procesos y la creación de procesos, mediante las cuales ayuda a la estructuración de las aplicaciones.

PROCESOS Y PCB

Un **proceso** es: Un programa en ejecución. Una instancia de un programa ejecutado en un computador. La entidad que se puede asignar y ejecutar en un procesador. Una unidad de actividad que se caracteriza por la ejecución de una secuencia de instrucciones, un estado actual, y un conjunto de recursos del sistema asociados.

Mientras el proceso está en ejecución, este proceso se puede caracterizar por una serie de elementos que se almacenan en el ***PCB ("Process Control Blokc")***, incluyendo los siguientes: * **Identificador**. Único asociado al proceso. * **Estado**. Si el proceso está actualmente corriendo, está en el estado en ejecución. * **Prioridad**. Nivel de prioridad relativo al resto de procesos. * **Contador de programa**. La dirección de la siguiente instrucción del programa que se

ejecutará. * **Punteros a memoria**. Incluye los punteros al código de programa y los datos asociados a dicho proceso, además de cualquier bloque de memoria compartido con otros procesos. * **Datos de contexto**. Estos son datos que están presenten en los registros del procesador cuando el proceso está corriendo. * **Información de estado de E/S**. Incluye las peticiones de E/S pendientes, dispositivos de E/S,... * **Información de auditoría**. Puede incluir la cantidad de tiempo de procesador y de tiempo de reloj utilizados, así como los límites de tiempo, registros contables, etc.

CREACIÓN Y TERMINACIÓN DE PROCESOS

Creación de un proceso. Cuando se va a crear un nuevo proceso el SO crea las estructuras de datos que se usan para manejar el proceso. La razones principales para la creación de un proceso son: nuevo proceso de lotes, sesión interactiva, creado por el sistema operativo para proporcionar un servicio y creado por un proceso existente. Cuando un proceso lanza otro, al primero se le denomina proceso padre, y al proceso creado se le denomina proceso hijo.

Terminación de procesos. Todo sistema debe proporcionar los mecanismos mediante los cuales un proceso indica su finalización, o que ha completado su tarea. Un trabajo por lotes debe incluir una instrucción HALT o unallamada a un servicio de sistema operativo especifica para su terminación. Las razones comunes para la finalización de un proceso son: finalización normal, memoria no disponible, límite de tiempo, error de protección, violación de memoria, fallo en la E/S. La cola de los procesos es una lista de tipo FIFO.

PROCESOS SUPENDIDOS

La necesidad de intercambio o swapping. Existe una buena justificación para añadir otros estados al modelo. Para ver este beneficio de nuevos estados, vamos a suponer un sistema que no utiliza memoria virtual. Cada proceso que se ejecuta debe cargarse completamente en memoria principal. Recuérdese que la razón de toda esta compleja maquinaria es que las operaciones de E/S son mucho más lentas que los procesos de cómputo y, por tanto, el procesador en un sistema monoprogramado estaría ocioso la mayor parte del tiempo. La diferencia de velocidad entre el procesador y la E/S es tal que sería muy habitual que todos los procesos en memoria se encontrasen a esperas de dichas operaciones. Por tanto, incluso en un sistema multiprogramado, el procesador puede estar ocioso la mayor parte del tiempo.

Swapping (memoria de intercambio), que implica mover parte o todo el proceso de memoria principal al disco. Cuando ninguno de los procesos en memoria principal se encuentra en estado Listo, el sistema operativo intercambia uno de los procesos bloqueados a disco, en la cola de Suspendidos. Esta es una lista de procesos existentes que han sido temporalmente expulsados de la

memoria principal, o suspendidos. El sistema operativo trae otro proceso de la cola de Suspendidos o responde a una solicitud de un nuevo proceso. La ejecución continúa con los nuevos procesos que han llegado. El swapping, sin embargo, es una operación de E/S, y por tanto existe el riesgo potencial de hacer que el problema empeore. Pero debido a que la E/S en disco es habitualmente más rápida que la E/S sobre otros sistemas (por ejemplo, comparado con cinta o impresora), el swapping habitualmente mejora el rendimiento del sistema. Debe añadirse un nuevo estado.

Cuando el SO ha realizado la operación de swap (transferencia a disco), tiene dos **opciones para seleccionar un nuevo proceso para traerlo a memoria principal:** ***puede admitir un nuevo proceso que se haya creado*** o ***puede traer un proceso que anteriormente se encontrase en estado de Suspendido.*** Parece que sería preferible traer un proceso que anteriormente estuviese suspendido, para proporcionar dicho servicio en lugar de incrementar la carga total del sistema. Todos los procesos que fueron suspendidos se encontraban previamente en el estado de Bloqueado en el momento de su suspensión. Claramente no sería bueno traer un proceso bloqueado de nuevo a memoria porque podría no encontrarse todavía listo para ejecución. Cuando el evento sucede, el proceso no está Bloqueado y está disponible para su ejecución. Hay dos conceptos independientes: ***si un proceso está esperando a un evento (Bloqueado o no)*** y ***si un proceso está transferido de memoria a disco (suspendido o no)***. Para describir estas combinaciones de 2x2 necesitamos 4 estados: * **Listo.** El proceso está en memoria principal disponible para su ejecución. * **Bloqueado.** El proceso está en memoria principal y esperando un evento. * **Bloqueado/Suspendido.** El proceso está en almacenamiento secundario y esperando un evento. * **Listo/Suspendido.** El proceso está en almacenamiento secundario pero está disponible para su ejecución tan pronto como sea cargado en memoria principal.

TRANSICIONES:

- **Bloqueado → Bloqueado/Suspendido.** Si no hay procesos listos, entonces al menos uno de los procesos bloqueados se transfiere al disco para hacer espacio para otro proceso que no se encuentra bloqueado. Esta transición puede realizarse incluso si hay procesos listos disponibles, si el sistema operativo determina que el proceso actualmente en ejecución o los procesos listos que desea ejecutar requieren más memoria principal para mantener un rendimiento adecuado.
- **Bloqueado/Suspendido → Listo/Suspendido.** Cuando sucede un evento al que estaba esperando. Nótese que esto requiere que la información de estado concerniente a un proceso suspendido sea accesible para el sistema operativo.
- **Listo/Suspendido → Listo.** Cuando no hay más procesos listos en memoria principal, el sistema operativo necesitará traer uno para continuar

la ejecución. Adicionalmente, puede darse el caso de que un proceso en estado Listo/Suspendido tenga mayor prioridad que cualquiera de los procesos en estado Listo. En este caso, el sistema operativo puede haberse diseñado para determinar que es más importante traer un proceso de mayor prioridad que para minimizar el efecto del swapping.

- **Listo → Listo/Suspendido.** El SO preferirá suspender procesos bloqueados que un proceso Listo, porque un proceso Listo se puede ejecutar en ese momento, mientras que un proceso Bloqueado ocupa espacio de memoria y no se puede ejecutar. Sin embargo, puede ser necesario suspender un proceso Listo si con ello se consigue liberar un bloque suficientemente grande de memoria.

(DIAGRAMA DE ESTADOS)

Otros usos para la suspensión de procesos. Hace poco, hemos definido el concepto de procesos suspendidos como el proceso que no se encuentra en memoria principal. Un proceso que no está en memoria principal no está disponible de forma inmediata para su ejecución, independientemente de si está a la espera o no de un evento. Podemos analizar el concepto de procesos suspendidos, definiendo un proceso suspendido como el que cumple las siguientes características: 1. El proceso no está inmediatamente disponible para su ejecución. 2. El proceso puede estar o no a la espera de un evento, si es así, la condición de bloqueo es independiente de la condición estar suspendido, y si sucede el evento que lo bloquea, eso no habilita al proceso para su ejecución inmediata. 3. El proceso fue puesto en estado suspendido por un agente: bien el proceso mismo, el proceso padre o el sistema operativo, con el propósito de prevenir su ejecución. 4. El proceso no puede ser recuperado de este estado hasta que el agente explícitamente así lo indique.

Una de las razones que hemos discutido anteriormente es proporcionar espacio de memoria para traer procesos en estado Listos/Suspendidos o para incrementar el espacio de memoria disponible para los procesos Listos. El SO puede tener otros ***motivos para suspender un proceso***. Si el sistema operativo detecta o sospecha que puede haber un problema, puede suspender procesos. Otro tipo de razones están relacionadas con el uso interactivo del sistema. Por ejemplo, si los usuarios sospechan de un programa con algún tipo de bug, se puede detener la ejecución de dicho programa, examinando y modificando el programa y sus datos y reanudándolo de nuevo.

Por último, un proceso padre puede suspender un proceso descendiente. Por ejemplo, el proceso A lanza al proceso B para realizar una lectura sobre un fichero. Posteriormente, el proceso B encuentra un error en la lectura del fichero y se lo indica al proceso A. El proceso A suspende al proceso B e investiga la causa. En todos estos casos, la activación de un proceso Suspendido se solicita por medio del agente que inicialmente puso al proceso en suspensión. Por último, un proceso padre puede suspender un proceso descendiente. Por ejemplo, el proceso A lanza al proceso B para realizar una lectura sobre un fichero. Posteriormente, el proceso B encuentra un error en la lectura del fichero y se lo indica al proceso

A.

ESTRUCTURAS DE CONTROL DEL SISTEMA OPERATIVO

Si el SO se encarga de la gestión de procesos y recursos, debe disponer de información sobre el estado actual de cada proceso y cada recurso. El mecanismo universal para proporcionar esta información es el siguiente: el SO construye y mantiene tablas de información sobre cada entidad que gestiona. Cuatro diferentes tipos de tablas: memoria, E/S, ficheros y procesos.

Las **tablas de memoria** se usan para mantener un registro tanto de la memoria principal (real) como de la secundaria (virtual). Parte de la memoria principal está reservada para el uso del SO; el resto está disponible para el uso de los procesos. Los procesos se mantienen en memoria secundaria utilizando algún tipo de memoria virtual o técnicas de swapping. Las tablas de memoria deben incluir la siguiente información: * *Las reservas de memoria principal por parte de los procesos.* * *Las reservas de memoria secundaria por parte de los procesos.* * *Todos los atributos de protección que restringe el uso de la memoria principal y virtual, de forma que los procesos puedan acceder a ciertas áreas de memoria compartida.* * *La información necesaria para manejar la memoria virtual.*

El SO debe utilizar las **tablas de E/S** para gestionar los dispositivos de E/S y los canales del computador. Pero, en un instante determinado, un dispositivo E/S puede estar disponible o asignado a un proceso en particular. Si la operación de E/S se está realizando, el sistema operativo necesita conocer el estado de la operación y la dirección de memoria principal del área usada como fuente o destino de la transferencia de E/S. El sistema operativo también puede mantener las **tablas de ficheros.** Estas tablas proporcionan información sobre la existencia de ficheros, su posición en almacenamiento secundario, su estado actual, y otros atributos. La mayoría de, o prácticamente toda, esta información se puede gestionar por el sistema de ficheros, en cuyo caso el SO tiene poco o ningún conocimiento sobre los ficheros. En otros sistemas operativos, la gran mayoría del detalle de la gestión de ficheros sí que gestiona en el propio sistema operativo.

Tablas de procesos para gestionar los procesos. Las tablas se encuentran entrelazadas y referenciadas entre sí de alguna manera. Memoria, E/S, y ficheros se gestionan por parte de los procesos, de forma que debe haber algunas referencias en estos recursos, directa o indirectamente, desde las tablas de procesos. Los ficheros indicados en las tablas de ficheros son accesibles mediante dispositivos E/S y estarán, en algún caso, residentes en memoria virtual o principal. Las tablas son también accesibles para el SO, y además están controladas por la gestión de memoria. ***¿Cómo puede el SO crear las tablas iniciales?*** Ciertamente, debe tener alguna información sobre el entorno básico, tal como: cuánta memoria física existe, cuáles son los dispositivos de E/S y cuáles son sus identificadores. Cuando el SO se inicializa, debe tener acceso a algunos datos de

configuración que definen el entorno básico y estos datos se deben crear fuera del sistema operativo, con asistencia humana o por medio de algún software de autoconfiguración.

ESTRUCTURAS DE CONTROL DE PROCESO

El sistema operativo si quiere manejar y controlar los procesos debe conocer dónde están localizados los procesos, y los atributos de los procesos que quiere gestionar.

Localización de los procesos. Un proceso debe incluir un programa o un conjunto de programas a ejecutar. Asociados con estos programas existen unas posiciones de memoria para los datos de variables locales y globales y de cualquier constante definida. Así, un proceso debe consistir en una cantidad suficiente de memoria para almacenar el programa y datos del mismo. Adicionalmente, se incluye una pila utilizada para registrar las llamadas a procedimientos y los parámetros pasados entre dichos procedimientos.

Por último, cada proceso está asociado a un número de atributos que son utilizados por el SO para controlar el proceso. El conjunto de estos atributos es el BCP. Nos podemos referir al conjunto de programa, datos, pila, y atributos, como la imagen del proceso. La posición de la imagen del proceso dependerá del esquema de gestión de memoria que se utilice. En el caso más simple, la imagen del proceso se mantiene como un bloque de memoria contiguo, o continuo. Este bloque se mantiene en memoria secundaria, habitualmente disco. Para que el SO pueda gestionar el proceso, al menos una pequeña porción de su imagen se debe mantener en memoria principal. Para ejecutar el proceso, la imagen completa se debe cargar en memoria principal o al menos en memoria virtual. Asimismo, el SO necesita conocer la posición en disco de cada proceso y, para cada proceso que se encuentre en memoria principal, su posición en dicha memoria.

Los sistemas operativos modernos suponen la existencia de un hardware de **paginación** que permite el uso de la memoria física no contigua, para dar soporte a procesos parcialmente residentes en la memoria principal. En esos casos, una parte de la imagen del proceso se puede encontrar en dicha memoria principal, con las restantes en memoria secundaria. De esta forma, las tablas mantenidas por sistema operativo deben mostrar la localización de cada página de la imagen del proceso. Hay una tabla primaria de procesos con una entrada por cada proceso. Cada entrada contiene, al menos, un puntero a la imagen del proceso. Si la imagen del proceso contiene múltiples bloques, esta información se mantiene directamente en la tabla principal del proceso o bien está disponible por referencias cruzadas entre las entradas de las tablas de memoria.

Atributos de proceso. Cualquier sistema operativo multiprogramado de la actualidad requiere una gran cantidad de información para manejar cada proceso. Como quedó explicado, esta información se puede considerar que reside en el ***bloque de control del proceso (BCP).*** Los diferentes sistemas organizarán

esta información de formas diferentes. Por ahora, exploraremos simplemente el tipo de información que el SO puede utilizar, sin considerar cualquier detalle de cómo esta información está organizada.

TABLA ELEMENTOS DE BCP

IDENTIFICADORES DEL PROCESO

Identificadores numéricos que se pueden guardar dentro del BCP: * identificadores del proceso * identificador del proceso que creó a este proceso (proceso padre) * identificador del usuario

INFORMACIÓN DE ESTADO DEL PROCESADOR

Registros visibles por el usuario Un registro visible por el usuario es aquel al que se puede hacer referencia por medio del lenguaje máquina que ejecuta el procesador cuando está en modo usuario.

Registros de estado y control * Contador de programa. * Códigos de condición: signo, cero, acarreo, igual, desbordamiento. * Información de estado: flags de interrupciones habilitadas/deshabilitadas, modo ejecución.

Puntero de pila Cada proceso tiene una o más pilas de sistema (LIFO) asociadas a él. Una pila se utiliza para almacenar los parámetros y las direcciones de retorno de los procedimientos y llamadas a sistema.

INFORMACIÓN DE CONTROL DE PROCESO

Información de estado y de planificación Esta es la información que el sistema operativo necesita para analizar las funciones de planificación. * Estado del proceso. * Prioridad. * Información relativa a planificación: esto dependerá del algoritmo de planificación utilizado. Por ejemplo, la cantidad de tiempo que el proceso ha ejecutado la última vez. * Evento: identificar el evento al cual el proceso está esperando para continuar su ejecución.

Estructuración de datos Un proceso puede estar enlazado con otro proceso en una cola, anillo o con cualquier otra estructura. Por ejemplo, todos los procesos en estado de espera por un nivel de prioridad en particular pueden estar enlazados en una cola. Un proceso puede mostrar una relación padre-hijo (creador-creado) con otro proceso. El BCP puede contener punteros a otros procesos.

Comunicación entre procesos Se pueden asociar flags, señales, y mensajes relativos a la comunicación entre dos procesos. Privilegios de proceso Los procesos adquieren privilegios de acuerdo con la memoria a la que van a acceder y los tipos de instrucciones que se pueden ejecutar.

Gestión de memoria Esta sección puede incluir punteros a tablas de segmentos y/o páginas que describen la memoria virtual asignada a este proceso.

Propia de recursos y utilización Se deben indicar los recursos controlados por el proceso, por ejemplo, ficheros abiertos. Se puede incluir un histórico de utilización del procesador o de otros recursos.

En prácticamente todos los sistemas operativos, a cada proceso se le asocia un identificador numérico único, que puede ser un índice en la tabla de procesos principal. Muchas de las otras tablas controladas por el sistema operativo incluyen información que referencia a otro proceso por medio del identificador de proceso. Pueden aparecer referencias similares en las tablas de E/S o de ficheros. Cuando un proceso se comunica con otro proceso, el identificador de proceso informa al SO del destino de la comunicación. Cuando los procesos pueden crear otros procesos, los identificadores indican el proceso padre y los descendientes. A un proceso se le puede asignar un identificador de usuario que indica qué usuario es responsable del trabajo.

La información de estado de proceso indica los contenidos de los registros del procesador. Cuando un proceso está ejecutando, esta información está, por supuesto, en los registros. Cuando un proceso se interrumpe, toda la información de los registros debe salvaguardarse de forma que se pueda restaurar cuando el proceso continúe con su ejecución. Normalmente, el conjunto de registros incluye registros visibles por usuario, registros de control y de estado, y punteros de pila. El diseño de todos los procesadores incluye un registro o conjuntos de registros, habitualmente conocidos como ***palabras de estado de programa (PSW).***

El papel del bloque de control de proceso. Es la más importante de las estructuras de datos del SO. Cada BCP contiene toda la información sobre un proceso que necesita el sistema operativo. Los bloques se leen y/o se modifican por la práctica totalidad de los módulos del SO. Se puede decir que el conjunto de los BCP definen el estado del sistema operativo.

Un gran número de rutinas dentro del SO necesitarán acceder a información de los BCP. Proporcionar acceso directo a estas tablas no es difícil. El PID puede utilizarse para indexar dentro de una tabla de punteros a BCP. La dificultad no reside en el acceso sino en la protección. Dos posibles problemas: * Un fallo en una simple rutina, como un manejador de interrupción, puede dañar los BCP de proceso, que puede destruir la capacidad del sistema para manejar los procesos afectados. * Un cambio en la estructura o en la semántica de los BCP puede afectar a un gran número de módulos del sistema operativo.

MODOS DE EJECUCIÓN

Muchos procesadores proporcionan al menos dos modos de ejecución. Ciertas instrucciones se pueden ejecutar en modos privilegiados únicamente. Éstas incluirían lectura y modificación de los registros de control, por ejemplo la palabra de estado de programa; instrucciones de E/S primitivas; e instrucciones relacionadas con la gestión de memoria. Adicionalmente, ciertas regiones de memoria sólo se pueden acceder en los modos más privilegiados. El modo menos

privilegiado a menudo se denomina **modo usuario**, porque los programas de usuario típicamente se ejecutan en este modo. El modo más privilegiado se denomina **modo sistema, modo control o modo núcleo**. Este último término se refiere al núcleo del sistema operativo, que es la parte del sistema operativo que engloba las funciones más importantes del sistema.

El motivo por el cual se usan los otros modos es claro. Se necesita proteger al sistema operativo y a las tablas clave del sistema, de la interferencia con programas de usuario. En modo núcleo, el software tiene control completo del procesador y de sus instrucciones, registros, y memoria. Este nivel de control no es necesario y por seguridad no es recomendable para los programas de usuario. Aparecen dos cuestiones: ***¿cómo conoce el procesador en que modo está ejecutando y cómo este modo puede modificarse?*** En lo referente la primera cuestión, existe típicamente un bit en la palabra de estado de programa (PSW) que indica el modo de ejecución. Este bit se cambia como respuesta a determinados eventos. Habitualmente, cuando un usuario realiza una llamada a un servicio del sistema operativo o cuando una interrupción dispara la ejecución de una rutina del sistema operativo, este modo de ejecución se cambia a modo núcleo y; tras la finalización del servicio, el modo se fija de nuevo a modo usuario.

CREACIÓN DE PROCESOS

Pasos que llevan a la verdadera creación de un proceso. Una vez que el sistema operativo decide, por cualquier motivo, crear un proceso procederá de la siguiente manera: 1. **Asignar un identificador de proceso único al proceso.** En este instante, se añade una nueva entrada a la tabla primaria de procesos, que contiene una entrada por proceso. 2. **Reservar espacio para proceso.** Para ello, el sistema operativo debe conocer cuánta memoria se requiere para el espacio de direcciones privado (programas y datos) y para la pila de usuario. Si existe una parte del espacio direcciones compartido por este nuevo proceso, se fijan los enlaces apropiados. Por último, se debe reservar el espacio para el bloque de control de proceso (BCP). 3. **Inicialización del bloque de control de proceso.** La parte de identificación de proceso del BCP contiene el identificador del proceso así como otros posibles identificadores. En la información de estado de proceso del BCP, habitualmente se inicializa con la mayoría de entradas a 0, excepto el contador de programa (fijado en el punto entrada del programa) y los punteros de pila de sistema (fijados para definir los límites de la pila del proceso). La parte de información de control de procesos se inicializa en base a los valores por omisión, considerando también los atributos que han sido solicitados para este proceso. Inicialmente, el proceso no debe poseer ningún recurso (dispositivos de E/S, ficheros) a menos que exista una indicación explícita de ello o que haya sido heredados del padre. 4. **Establecer los enlaces apropiados.** 5. **Creación o expansión de otras estructuras de datos.**

CAMBIO DE PROCESO

Cuándo se realiza el cambio de proceso. Un cambio de proceso puede ocurrir en cualquier instante en el que el sistema operativo obtiene el control sobre el proceso actualmente en ejecución.

Primero, consideremos las ***interrupciones del sistema.*** Podemos distinguir dos tipos diferentes de interrupciones de sistema, unas simplemente denominadas **interrupciones**, y las otras denominadas **traps**. Las primeras se producen por causa de algún tipo de evento que es externo e independiente al proceso actualmente en ejecución, por ejemplo la finalización de la operación de E/S. Las otras están asociadas a una condición de error o excepción generada dentro del proceso que está ejecutando, como un intento de acceso no permitido a un fichero.

Dentro de una interrupción ordinaria, el control se transfiere inicialmente al manejador de interrupción, que realiza determinadas tareas internas y que posteriormente salta a una rutina del sistema operativo, encargada de cada uno de los tipos de interrupciones en particular. Algunos ejemplos son: * **Interrupción de reloj.** El sistema operativo determinar si el proceso en ejecución ha excedido o no la unidad máxima de tiempo de ejecución, denominada *rodaja de tiempo (time slice)*. Esto es, una rodaja de tiempo es la máxima cantidad de tiempo que un proceso puede ejecutar antes de ser interrumpido. En dicho caso, este proceso se puede pasar al estado de Listo y se puede activar otro proceso. * **Interrupción de E/S.** El sistema operativo determina qué acción de E/S ha ocurrido. Si la acción de E/S constituye un evento por el cual están esperando uno o más procesos, el sistema operativo mueve todos los procesos correspondientes al estado de Listos (y los procesos en estado Bloqueado/Suspendido al estado Listo/Suspendido). El sistema operativo puede decidir si reanuda la ejecución del proceso actualmente en estado Ejecutando o si lo expulsa para proceder con la ejecución de un proceso Listo de mayor prioridad. * **Fallo de memoria.** El procesador se encuentra con una referencia a una palabra que no se encuentra en memoria principal. El sistema operativo debe traer el bloque (página o segmento) que contiene la referencia desde memoria secundaria a memoria principal. Después de que se solicita la operación de E/S para traer el bloque a memoria, el proceso que causó el fallo de memoria se pasa al estado de Bloqueado; el sistema operativo realiza un cambio de proceso y pone a ejecutar a otro proceso. Después de que el bloque de memoria solicitado se haya traído, el proceso pasará al estado Listo.

Con un **trap**, el sistema operativo conoce si una condición de error o de excepción es irreversible. Si es así, el proceso en ejecución se pone en el estado Saliente y se hace un cambio de proceso. De no ser así, el sistema operativo decidirá como actuar.

Por último, el sistema operativo se puede activar por medio de una llamada al sistema procedente del programa en ejecución. Por ejemplo, si se está ejecutando

un proceso y se ejecuta una operación que implica una llamada de E/S, como abrir un archivo. Esta llamada implica un salto a una rutina que es parte del código del sistema operativo. La realización de una llamada al sistema puede implicar en algunos casos que el proceso que la realiza pase al estado de Bloqueado.

Cambio de modo. En la fase de interrupción, el procesador comprueba que no exista ninguna interrupción pendiente, indicada por la presencia de una señal de interrupción. Si hay una interrupción pendiente, el proceso actúa de la siguiente manera: 1. Coloca el contador de programa en la dirección de comienzo de la rutina del programa manejador de la interrupción. 2. Cambia de modo usuario a modo núcleo de forma que el código de tratamiento de la interrupción pueda incluir instrucciones privilegiadas.

El procesador, acto seguido, pasa a la fase de búsqueda de instrucción y busca la primera instrucción del programa de manejo de interrupción, que dará servicio a la misma. En este punto, habitualmente, el contexto del proceso que se ha interrumpido se salvaguarda en el bloque de control de proceso del programa interrumpido (toda la información que se puede ver alterada por la ejecución de la rutina de interrupción y que se necesitará para la continuación del proceso que ha sido interrumpido.)

El manejador de interrupción es habitualmente un pequeño programa que realiza unas pocas tareas básicas relativas a la interrupción.

¿Qué pasa con el resto de información del bloque de control de proceso? Si a esta interrupción le sigue un cambio de proceso a otro proceso, se necesita hacer algunas cosas más. Pero si no hay cambio de proceso sólo se necesita salvaguardar la información del estado del procesador cuando se produce la interrupción y restablecerlo cuando se reanude la ejecución del proceso.

Habitualmente, las operaciones de salvaguarda y recuperación se realizan por hardware.

Cambio del estado del proceso. Esta claro, por tanto, que el cambio de modo es un concepto diferente del cambio de proceso. Un cambio de modo puede ocurrir sin que se cambie el estado del proceso actualmente en estado Ejecutando. En dicho caso, la salvaguarda del estado y su posterior restauración comportan sólo una ligera sobrecarga. Sin embargo, si el proceso actualmente en estado Ejecutando, se va a mover a cualquier otro estado (Listo, Bloqueado, etc.), entonces el sistema operativo debe realizar cambios sustanciales en su entorno. Los pasos que se realizan para un cambio de proceso completo son:

1. Salvar el estado del procesador, incluyendo el contador de programa y otros registros.
2. Actualizar el bloque de control del proceso que está actualmente en el estado Ejecutando. Esto incluye cambiar el estado del proceso a uno de los otros estados (Listo, Bloqueado, Listo/Suspendido, o Saliente). También se tienen que actualizar otros campos importantes, incluyendo la razón por

la cual el proceso ha dejado el estado de Ejecutando y demás información de auditoría.
3. Mover el bloque de control de proceso a la cola apropiada (Listo, Bloqueado en el evento i, Listo/Suspendido).
4. Selección de un nuevo proceso a ejecutar.
5. Actualizar el bloque de control del proceso elegido.
6. Actualizar las estructuras de datos de gestión de memoria.
7. Restaurar el estado del procesador al que tenía en el momento en el que el proceso seleccionado salió del estado Ejecutando por última vez.

Por tanto, el cambio de proceso, que implica un cambio en el estado, requiere un mayor esfuerzo que un cambio de modo.

EJECUCIÓN DEL SISTEMA OPERATIVO

Si el sistema operativo es simplemente una colección de programas y si son ejecutados por el procesador, tal y como cualquier otro programa, ¿es el sistema operativo un proceso del sistema? y si es así ¿cómo se controla?.

Núcleo sin procesos. Una visión tradicional es la ejecución del sistema operativo fuera de todo proceso. Con esta visión, cuando el proceso en ejecución se interrumpe o invoca una llamada al sistema, el contexto se guarda y el control pasa al núcleo. El sistema operativo tiene su propia región de memoria y su propia pila de sistema para controlar la llamada a procedimientos y sus retornos. El sistema operativo puede realizar todas las funciones que necesite y restaurar el contexto del proceso interrumpido, que hace que se retome la ejecución del proceso de usuario afectado. De forma alternativa, el sistema operativo puede realizar la salvaguarda del contexto y la activación de otro proceso diferente.

En cualquier caso, el punto clave en este caso es que el concepto de proceso se aplica aquí únicamente a los programas de usuario. El código del sistema operativo se ejecuta como una entidad independiente que requiere un modo privilegiado de ejecución.

Ejecución dentro de los procesos de usuario. Una alternativa que es común en los sistemas operativos de máquinas pequeñas (PC, estaciones de trabajo) es ejecutar virtualmente todo el software de sistema operativo en el contexto de un proceso de usuario. Esta visión es tal que el sistema operativo se percibe como un conjunto de rutinas que el usuario invoca para realizar diferentes funciones, ejecutadas dentro del entorno del proceso de usuario.

Cuando ocurre una interrupción, trap o llamada al sistema, el procesador se pone en modo núcleo y el control se pasa al sistema operativo. Para este fin, el contexto se salva y se cambia de modo a una rutina del sistema operativo. Sin embargo, la ejecución continúa dentro del proceso de usuario actual. De esta forma, no se realiza un cambio de proceso, sino un cambio de modo dentro del mismo proceso.

Si el sistema operativo, después de haber realizado su trabajo, determina que el proceso actual debe continuar con su ejecución, entonces el cambio de modo continúa con el programa interrumpido, dentro del mismo proceso. Ésta es una de las principales ventajas de esta alternativa: se ha interrumpido un programa de usuario para utilizar alguna rutina del sistema operativo, y luego continúa, y todo esto se ha hecho sin incurrir en un doble cambio de proceso.

Sistemas operativos basados en procesos. Otra alternativa, es implementar un sistema operativo como una colección de procesos de sistema. Como en las otras opciones, el software que es parte del núcleo se ejecuta en modo núcleo. En este caso, las principales funciones del núcleo se organizan como procesos independientes. De nuevo, debe haber una pequeña cantidad de código para intercambio de procesos que se ejecuta fuera de todos los procesos.

Esta visión tiene diferentes ventajas. Impone una disciplina de diseño de programas que refuerza el uso de sistemas operativos modulares con mínimas y claras interfaces entre los módulos. Adicionalmente, otras funciones del sistema operativo que no sean críticas están convenientemente separadas como otros procesos.

Descripción de procesos: Carretero pp. 80-91

Un **sistema operativo monotarea o monoproceso**, solamente permite que exista un proceso en cada instante. La ventaja de estos sistemas operativos es que son muy sencillos.

Por el contrario, un **sistema operativo multitarea o multiproceso** permite que coexistan varios procesos activos a la vez. El sistema operativo se encarga de ir repartiendo el tiempo del procesador entre estos procesos, para que todos ellos vayan avanzando en su ejecución.

Un **sistema monousuario** está previsto para dar soporte a un solo usuario. Estos sistemas pueden ser monoproceso o multiproceso.

El **sistema operativo multiusuario o de tiempo compartido** da soporte a varios usuarios que trabajan simultáneamente desde varios terminales. A su vez, cada usuario puede tener activos más de un proceso, por lo que el sistema, obligatoriamente, ha de ser multitarea.

La **multitarea** se basa en las tres características siguientes: * Paralelismo real entre E/S y procesador. * Alternancia en los procesos de fases de E/S y de procesamiento. * Memoria principal capaz de almacenar varios procesos.

En un sistema monotarea el procesador no tiene nada que hacer durante las fases de entrada/salida, por lo que desperdicia su potencia de procesamiento.

En un sistema multitarea se aprovechan las fases de entrada/salida de unos procesos para realizar las fases de procesamiento de otros.

Finalmente es importante destacar que la multitarea exige tener más de un proceso activo y cargado en memoria principal. Por tanto, hay que disponer de suficiente memoria principal para albergar a estos procesos.

Proceso nulo El procesador no para de ejecutar nunca, pero, ¿qué hacer cuando no hay procesos disponibles? Para evitar esta contradicción, los sistemas operativos incluyen el denominado proceso nulo. Este proceso consiste en un bucle infinito que no realiza ninguna operación útil. El objetivo de este proceso es «entretener» al procesador cuando no hay ninguna otra tarea.

El ***planificador (scheduler)*** forma parte del núcleo del sistema operativo. Entra en ejecución cada vez que se activa el sistema operativo y su misión es seleccionar el proceso que se ha de ejecutar a continuación.

El ***activador (dispatcher)*** también forma parte del sistema operativo y su función es poner en ejecución el proceso seleccionado por el planificador.

Ventajas de la multitarea

- Facilita la programación. Permite dividir las aplicaciones en varios procesos, lo que beneficia a su modularidad.
- Permite prestar un buen servicio, puesto que se puede atender a varios usuarios de forma eficiente, interactiva y simultánea.
- Aprovecha los tiempos muertos que los procesos pasan esperando a que se completen sus operaciones de E/S.
- Aumenta el uso de la UCP, al aprovechar los espacios de tiempo que los procesos están bloqueados.

Se denomina **grado de multiprogramación** al número de procesos activos que mantiene un sistema. Influye en el rendimiento. Mientras más procesos activos haya en un sistema, mayor es la probabilidad de encontrar siempre un proceso en estado de listo para ejecutar, por lo que entrará a ejecutar menos veces el proceso nulo. Sin embargo, se tiene el inconveniente de que, a mayor grado de multiprogramación, se tienen mayores necesidades de memoria.

En un ***sistema sin memoria virtual*** los procesos activos han de residir totalmente en memoria principal. Por tanto, el grado de multiprogramación viene limitado por el tamaño de los procesos y por la memoria disponible.

En los ***sistemas con memoria virtual*** los procesos sólo tienen en memoria principal su conjunto residente, lo que hace que quepan mas procesos. Sin embargo, al aumentar el número de procesos disminuye el conjunto residente de cada uno y al rebasar un mínimo, ya no representa adecuadamente al proceso lo que tiene como consecuencia que se produzcan muchos fallos de página. Se denomina ***hiperpaginación (trashing)*** a esta situación.

Cuando la memoria principal disponible es pequeña, se llega a la situación de hiperpaginación antes de alcanzar una cota alta de utilización del procesador.

Para aumentar el rendimiento de un sistema que esté en esta situación es necesario añadir más memoria principal. Cuando la memoria es grande se llega a saturar el procesador con menos procesos de los que caben en memoria. En este caso se puede aumentar el rendimiento del sistema manteniendo la memoria pero aumentando la potencia del procesador o añadiendo otro procesador.

Información de un proceso

El proceso tiene asociado una serie de elementos de información, divididos en:

INFORMACIÓN DEL ESTADO DEL PROCESADOR

- Registros generales.
- Contador de programa.
- Puntero de pila.
- Registro o registros de estado.
- Registros especiales.

El estado del procesador de un proceso reside en los registros del procesador, cuando el proceso está en ejecución, o en el bloque de control de proceso (BCP), cuando el proceso no está en ejecución.

IMAGEN DE MEMORIA DEL PROCESO

La imagen de memoria del proceso está formada por los espacios de memoria que está autorizado a utilizar.

- El proceso solamente puede tener información en su imagen de memoria y no fuera de ella. Si genera una dirección que esté fuera de ese espacio, el hardware de protección deberá detectarlo y levantar una excepción.
- La imagen de memoria estará referida a memoria virtual o a memoria física.
- Los procesos suelen necesitar asignación dinámica de memoria.
- No hay que confundir la asignación de memoria con la asignación de marcos de memoria.

El sistema operativo asigna la memoria al proceso, para lo cual puede emplear distintos modelos de imagen de memoria:

- Imagen de memoria con un único segmento de tamaño fijo
- Proceso con un único segmento de tamaño variable
- Proceso con un número fijo de segmentos de tamaño variable en el que un proceso contiene varios tipos de información, pero en el que el sistema operativo deja la función de gestionar esta información:
  - Texto o código. Bajo este nombre se considera el programa máquina que ha de ejecutar el proceso.
  - Datos. Este bloque de información depende mucho de cada proceso (dinamicos, estaticos, inicializados,. . .)

  - Pila. A través del puntero de pila, los programas utilizan una estructura dc pila residente en memoria.
- Proceso con un número variable de segmentos de tamaño variable

INFORMACIÓN DE BCP

El BCP contiene la información básica del proceso, entre la que cabe destacar la siguiente: * Información de identificación (del proceso, del padre del proceso, del usuario, etc.) * Estado del procesador * Información de control del proceso (planificación y estado del proceso, segmentos de memoria asignados al proceso, recursos asignados, punteros para estructurar los procesos y la comunicación entre procesos).

TABLAS DEL SISTEMA OPERATIVO

La información asociada a cada proceso se encuentra parcialmente en el BCP y parcialmente fuera de él. La decisión de incluir o no una información en el BCP se toma según dos argumentos: eficiencia y necesidad de compartir información.

En las tablas de entrada/salida el sistema operativo mantiene la información asociada a los periféricos y a las operaciones de entrada/salida.

Concepto y tipos de Hebras (hilos) (Stalling 158-172)

Ya se ha presentado el concepto de un proceso como poseedor de dos características:

- **Propiedad de recursos**. Un proceso incluye un espacio de direcciones virtuales para el manejo de la imagen del proceso (la colección de programa, datos, pila y atributos definidos en el bloque de control del proceso).
- **Planificación/ejecución**. La ejecución de un proceso sigue una ruta de ejecución (traza) a través de uno o más programas. De esta manera, un proceso tiene un estado de ejecución (Ejecutando, Listo, etc.) y una prioridad de activación y ésta es la entidad que se planifica y activa por el sistema operativo.

En la mayor parte de los sistemas operativos tradicionales, estas dos características son, realmente, la esencia de un proceso. Para distinguirlas, la unidad que se activa se suele denominar **hilo (thread), o proceso ligero**, mientras que la unidad de propiedad de recursos se suele denominar **proceso o tarea**.

Multihilo se refiere a la capacidad de un sistema operativo de dar soporte a múltiples hilos de ejecución en un solo proceso. En un entorno multihilo, un

proceso se define como la unidad de asignación de recursos y una unidad de protección. Se asocian con procesos los siguientes:

- Un espacio de direcciones virtuales que soporta la imagen del proceso.
- Acceso protegido a procesadores, otros procesos (para comunicación entre procesos), archivos y recursos de E/S (dispositivos y canales).

Dentro de un proceso puede haber uno o más hilos, cada uno con:

- Un estado de ejecución por hilo (Ejecutando, Listo, etc.).
- Un contexto de hilo que se almacena cuando no está en ejecución.
- Una pila de ejecución.
- Por cada hilo, espacio de almacenamiento para variables locales.
- Acceso a la memoria y recursos de su proceso, compartido con todos los hilos de su mismo proceso.

Vemos ahora la diferencia entre hilos y procesos desde el punto de vista de gestión de procesos:

- En un modelo de proceso monohilo (es decir, no existe el concepto de hilo), la representación de un proceso incluye su bloque de control de proceso y el espacio de direcciones de usuario, además de las pilas de usuario y núcleo para gestionar el comportamiento de las llamadas/retornos en la ejecución de los procesos. Mientras el proceso está ejecutando, los registros del procesador se controlan por ese proceso y, cuando el proceso no se está ejecutando, se almacena el contenido de estos registros.
- En un entorno multihilo, sigue habiendo un único bloque de control del proceso y un espacio de direcciones de usuario asociado al proceso, pero ahora hay varias pilas separadas para cada hilo, así como un bloque de control para cada hilo que contiene los valores de los registros, la prioridad, y otra información relativa al estado del hilo.

De esta forma, todos los hilos de un proceso comparten el estado y los recursos de ese proceso, residen en el mismo espacio de direcciones y tienen acceso a los mismos datos. Cuando un hilo cambia determinados datos en memoria, otros hilos ven los resultados cuando acceden a estos datos. Si un hilo abre un archivo con permisos de lectura, los demás hilos del mismo proceso pueden también leer ese archivo.

Los mayores beneficios de los hilos provienen de las consecuencias del rendimiento:

1. Lleva mucho menos tiempo crear un nuevo hilo en un proceso existente que crear un proceso totalmente nuevo.
2. Lleva menos tiempo finalizar un hilo que un proceso.
3. Lleva menos tiempo cambiar entre dos hilos dentro del mismo proceso.
4. Los hilos mejoran la eficiencia de la comunicación entre diferentes programas que están ejecutando. En la mayor parte de los sistemas operativos, la comunicación entre procesos independientes requiere la intervención del

núcleo. Sin embargo, ya que los hilos dentro de un mismo proceso comparten memoria y archivos, se pueden comunicar entre ellos sin necesidad de invocar al núcleo.

En un sistema operativo que soporte hilos, la planificación y la activación se realizan a nivel de hilo. Existen, sin embargo, diversas acciones que afectan a todos los hilos de un proceso y que el sistema operativo debe gestionar a nivel de proceso. Ejemplo: Suspender un proceso implica expulsar el espacio de direcciones de un proceso de memoria principal para dejar hueco a otro espacio de direcciones de otro proceso. Ya que todos los hilos de un proceso comparten el espacio de direcciones, todos los hilos se suspenden al mismo tiempo. De forma similar, la finalización de un proceso finaliza todos los hilos de ese proceso.

Estados de los hilos. Igual que con los procesos, los principales estados de los hilos son: Ejecutando, Listo y Bloqueado. Operaciones básicas de cambio de estado de hilo:

- **Creación.** Cuando se crea un nuevo proceso, también se crea un hilo de dicho proceso. Posteriormente, un hilo del proceso puede crear otro hilo dentro del mismo proceso, proporcionando un puntero a las instrucciones y los argumentos para el nuevo hilo. Al nuevo hilo se le proporciona su propio registro de contexto y espacio de pila y se coloca en la cola de Listos.
- **Bloqueo.** Cuando un hilo necesita esperar por un evento se bloquea, almacenando los registros de usuario, contador de programa y punteros de pila. El procesador puede pasar a ejecutar otro hilo en estado Listo, dentro del mismo proceso o en otro diferente.
- **Desbloqueo.** Cuando sucede el evento por el que el hilo está bloqueado, el hilo se pasa a la cola de Listos.
- **Finalización.** Cuando se completa un hilo, se liberan su registro de contexto y pilas.

Existen dos amplias categorías de implementación de hilos: **hilos de nivel de usuario *(user-level threads, ULT)*** e **hilos de nivel de núcleo *(kernel-level threads, KLT)***. Los últimos son también conocidos en la literatura como hilos soportados por el núcleo (kernel-supported threads) o procesos ligeros (lightweight processes).

Hilos de nivel de usuario.

En un entorno ULT puro, la aplicación gestiona todo el trabajo de los hilos y el núcleo no es consciente de la existencia de los mismos. Cualquier aplicación puede programarse para ser multihilo a través del uso de una biblioteca de hilos que contiene código para la creación y destrucción de hilos, para paso de

mensajes y datos entre los hilos, para planificar la ejecución de los hilos, y para guardar y restaurar el contexto de los hilos.

Por defecto, una aplicación comienza con un solo hilo y ejecutando en ese hilo. Esta aplicación y su hilo se localizan en un solo proceso gestionado por el núcleo. En cualquier momento que la aplicación esté ejecutando, la aplicación puede crear un nuevo hilo a ejecutar dentro del mismo proceso. El núcleo continúa planificando el proceso como una unidad y asigna al proceso un único estado (Listo, Ejecutando, Bloqueado, etc.).

El uso de ULT en lugar de KLT, presenta las siguientes ventajas:

1. El cambio de hilo no requiere privilegios de modo núcleo porque todas las estructuras de datos de gestión de hilos están en el espacio de direcciones de usuario de un solo proceso. Por consiguiente, el proceso no cambia a modo núcleo para realizar la gestión de hilos. Esto ahorra la sobrecarga de dos cambios de modo (usuario a núcleo; núcleo a usuario).
2. El algoritmo de planificación se puede hacer a medida sin tocar el planificador del sistema operativo.
3. Los ULT pueden ejecutar en cualquier sistema operativo.

Hay dos desventajas de los ULT en comparación con los KLT: 1. En un sistema operativo típico muchas llamadas al sistema son bloqueantes. Como resultado, cuando un ULT realiza una llamada al sistema, no sólo se bloquea ese hilo, sino que se bloquean todos los hilos del proceso.

2. En una estrategia pura ULT, una aplicación multihilo no puede sacar ventaja del multiproceso. El núcleo asigna el proceso a un solo procesador al mismo tiempo. Por consiguiente, en un determinado momento sólo puede ejecutar un hilo del proceso.

Una forma de solucionar el problema de hilos que se bloquean es una técnica denominada jacketing (revestimiento). El objetivo de esta técnica es convertir una llamada al sistema bloqueante en una llamada al sistema no bloqueante.

Hilos a nivel de núcleo.

En un entorno KLT puro, el núcleo gestiona todo el trabajo de gestión de hilos. No hay código de gestión de hilos en la aplicación, solamente una interfaz de programación de aplicación (API) para acceder a las utilidades de hilos del núcleo. Windows es un ejemplo de este enfoque.

Cualquier aplicación puede programarse para ser multihilo. Todos los hilos de una aplicación se mantienen en un solo proceso. El núcleo mantiene información de contexto del proceso como una entidad y de los hilos individuales del proceso. La planificación realizada por el núcleo se realiza a nivel de hilo. Este enfoque resuelve los dos principales inconvenientes del enfoque ULT. Primero, el núcleo

puede planificar simultáneamente múltiples hilos de un solo proceso en múltiples procesadores. Segundo, si se bloquea un hilo de un proceso, el núcleo puede planificar otro hilo del mismo proceso. Otra ventaja del enfoque KLT es que las rutinas del núcleo pueden ser en sí mismas multihilo.

La principal desventaja del enfoque KLT en comparación con el enfoque ULT es que la transferencia de control de un hilo a otro del mismo proceso requiere un cambio de modo al núcleo.

Enfoques combinados.

Algunos sistemas operativos proporcionan utilidades combinadas ULT/KLT, Solaris es el principal ejemplo de esto. En un sistema combinado, la creación de hilos se realiza por completo en el espacio de usuario, como la mayor parte de la planificación y sincronización de hilos dentro de una aplicación. Los múltiples ULT de una aplicación se asocian en un número (menor o igual) de KLT. El programador debe ajustar el número de KLT para una máquina y aplicación en particular para lograr los mejores resultados posibles.

En los enfoques combinados, múltiples hilos de la misma aplicación pueden ejecutar en paralelo en múltiples procesadores, y una llamada al sistema bloqueante no necesita bloquear el proceso completo. Si el sistema está bien diseñado, este enfoque debería combinar las ventajas de los enfoques puros ULT y KLT, minimizando las desventajas.

Relación muchos-a-muchos.

Un **dominio** es una entidad estática, que consiste en un espacio de direcciones y «puertos» a través de los cuales se pueden enviar y recibir mensajes. Un hilo es una ruta de ejecución, con una pila de ejecución, estado del procesador e información de planificación. Al igual que en el enfoque multihilo visto hasta el momento, múltiples hilos podrían ejecutar en un solo dominio, proporcionando las ventajas discutidas anteriormente. Sin embargo, también es posible realizar la actividad de usuario o ejecutar aplicaciones en múltiples dominios. En este caso hay un hilo que se puede mover entre dominios.

Hay varias formas de implementar esta aplicación: 1. El programa completo puede implementarse como un solo proceso. Existen desventajas relativas a la gestión de memoria. 2. El programa principal y el subprograma de E/S podrían implementarse como dos procesos in dependientes. Este enfoque presenta la sobrecarga de la creación del proceso subordinado. 3. Tratar al programa principal y al subprograma de E/S como una sola actividad que se puede implementar en un solo hilo. Sin embargo, se podría crear un espacio de direcciones (dominio) para el programa principal y otro para el subprograma de

E/S. De esta forma, el hilo se podría mover entre los dos espacios de direcciones a lo largo de la ejecución. El sistema operativo puede gestionar los dos espacios de direcciones de forma independiente, y no se genera una sobrecarga de creación de procesos. Adicionalmente, el espacio de direcciones utilizado por el subprograma de E/S también podría ser compartido por otros programas de E/S.

Relación uno-a-muchos.

En el campo de los sistemas operativos distribuidos (diseñados para controlar sistemas de computadores distribuidos), ha habido interés en el concepto de un hilo como una entidad que se puede mover entre espacios de direcciones.