Tema 3

Descripción y control de procesos

La unidad fundamental de un sistema operativo es el proceso. Cuando un usuario ejecuta un programa se convierte en un proceso. Éste es el que va a ejecutarse en la CPU y va a hacer uso de los distintos recursos del sistema. En este tema veremos cómo se representa un proceso en el sistema, qué estructuras se necesitan para gestionarlo, y la evolución que sufre desde que se crea hasta que termina. Finalmente, se estudiará una nueva tendencia en los sistemas operativos: los hilos.

3.1. ¿Qué es un proceso?

El término proceso fue utilizado por primera vez por los diseñadores del sistema Multics en los años sesenta. Desde entonces, del término proceso, que

a veces se utiliza como sinónimo de **tarea**[1], se han dado muchas definiciones. Nosotros nos quedaremos con una primera definición muy simple que es la siguiente: un proceso en un programa en ejecución. En ella estamos resaltando la diferencia entre programa (entidad estática) y proceso (entidad dinámica). Un usuario del sistema no ejecuta procesos, sino programas. Un programa es una colección de instrucciones y datos que se encuentran almacenados en un fichero, es decir, es algo pasivo. Si queremos ejecutarlo, el programa debe pasar previamente por una serie de fases, que pueden verse en la figura 3.1.

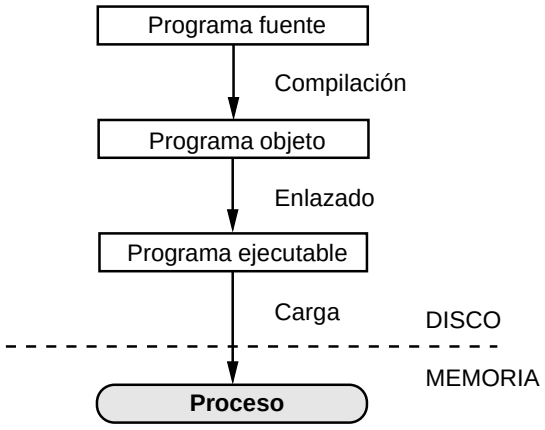

Figura 3.1: Conversión de un programa fuente en proceso

En ella podemos ver que el programa fuente debe ser previamente compilado para obtener un módulo objeto. Éste debe ser enlazado con otros módulos objeto para formar un único módulo de carga ejecutable. Para poderlo ejecutar es necesario cargarlo en memoria, labor realizada por el cargador. Es en ese momento cuando el programa se convierte en proceso.

[1] En la bibliografía sobre sistema operativos no existe unanimidad para designar el concepto de una secuencia de código en ejecución. Muchos libros utilizan el término proceso mientras que los manuales de las empresas y los programadores de sistemas se refieren a tarea. La razón de esta dualidad puede ser debida a que el término proceso puede ser algo confuso cuando se utiliza para describir un entorno de multiprogramación, ya que un sistema de multiprocesamiento es entendido normalmente como un sistema con varios procesadores, en vez de un sistema con un solo procesador en el que se ejecutan varios procesos concurrentemente. Con el término multitarea se designa un sistema operativo que soporta ejecución concurrente de programas sobre un solo procesador sin soportar necesariamente formas elaboradas de gestión de memoria y gestión de ficheros. La multiprogramación es un concepto más general que designa a un sistema operativo que proporciona ejecución concurrente de programas, gestión de memoria y gestión de ficheros. Por tanto, podemos decir que un sistema operativo de multiprogramación es también un sistema operativo multitarea, mientras que lo inverso no siempre es cierto.

Un proceso no sólo consiste en una copia del programa (conjunto de instrucciones y datos), sino que además el sistema operativo le añade cierta información adicional para poder gestionarlo.

A medida que vayamos profundizando en nuestro estudio de los procesos veremos que este concepto es más complejo y sutil de lo que parece a primera vista, y de hecho, abarca dos conceptos separados y potencialmente independientes, uno relativo a la propiedad de los recursos y el otro relativo a la ejecución.

La definición de sistema operativo dada por Denning, nos da idea de la importancia del concepto de proceso: "Sistema operativo es el software de un sistema informático que asiste al hardware para realizar funciones de gestión de procesos." Analizando esta definición, podemos ver que los requisitos principales que debe cumplir un sistema operativo multiprogramado están relacionados con la gestión de procesos. Así, una de las funciones principales del sistema operativo es la creación de procesos de usuario. Además, es responsabilidad del sistema operativo la ejecución intercalada de los procesos con objeto de obtener un mayor rendimiento del procesador, siempre que proporcione un tiempo de respuesta razonable (en el tema 4 se estudiará en más profundidad este aspecto). Por otro lado, el sistema operativo también se encarga de la asignación de recursos a los procesos con objeto de evitar la aparición de problemas como el interbloqueo (éstos se estudiarán en el tema 6). Igualmente, otra de las funciones del sistema operativo es facilitar la comunicación entre los distintos procesos de usuario (como se verá en el tema 5).

Dado que los procesos son una pieza clave dentro de los sistemas operativos, vamos a empezar el estudio en detalle de estos últimos, examinando cómo representan y controlan a los procesos.

3.2. Vida de un proceso

La vida de un proceso viene delimitada por su creación y su terminación. Durante el período de vida de un proceso, éste pasa por una serie de estados; el cambio de un estado a otro se denomina **transición de estado** o **cambio de estado**, y puede producirse por necesidad del sistema o del propio proceso. De este modo, se puede definir la ejecución de un proceso como la progresión a través de un conjunto de estados.

En cualquier momento, los estados de todos los procesos del sistema y de

los recursos (asignados o libres) definen lo que se conoce como **estado global del sistema**. A partir de esta información, el sistema operativo utiliza sus algoritmos de planificación para decidir cómo asignar los recursos disponibles a los procesos que los solicitan, de tal forma que el rendimiento resultante satisfaga los objetivos de diseño.

3.2.1. Creación de un proceso

Para la creación de un proceso nuevo, el sistema operativo tiene que realizar una serie de acciones que se pueden dividir en dos etapas:

1. Creación de las estructuras de datos que se necesitan para la administración del proceso.
2. Asignación del espacio de direcciones que va a utilizar el proceso.

La creación de un nuevo proceso puede ser debida a diferentes motivos, que incluso pueden variar de un sistema a otro. Algunos sucesos que pueden dar lugar a la creación de un nuevo proceso pueden ser:

- En un sistema por lotes se crea un nuevo proceso cuando se envía un trabajo a ejecutar.
- En un sistema interactivo se crea un proceso cuando un nuevo usuario quiere acceder a él.
- El sistema operativo puede crear un proceso para proporcionar un servicio a otro, por ejemplo, para imprimir un fichero. Así el proceso que hace la petición puede proseguir su ejecución.
- Un proceso existente puede querer crear otro, por ejemplo para realizar dos acciones en paralelo. Cuando el sistema operativo crea un proceso a instancias de otro, la acción se conoce como **generación de procesos**. Cuando un proceso genera a otro, el generador se conoce como **proceso padre** y el generado como **hijo**. De esta forma se establece una jerarquía entre los procesos.

Hay que hacer notar que en los tres primeros casos, el nuevo proceso se crea por iniciativa del sistema operativo y es transparente al programa de usuario, mientras que en el último caso es un proceso de usuario el que lo causa.

3.2.2. Terminación de un proceso

En la terminación de un proceso también pueden distinguirse dos etapas: en primer lugar, el sistema operativo le retira el espacio de direcciones del que disponía el proceso, y posteriormente destruye las estructuras de datos que creó para su manejo.

Hay muchas razones por las que un proceso puede terminar, la más inmediata es porque termina su ejecución de forma normal; en este caso, el proceso ejecutará un servicio para indicar que ha terminado. Las otras razones son debidas a errores que se producen durante la ejecución del proceso. Algunos de éstos pueden ser:

- El proceso intenta acceder a una localización de memoria a la cual no tiene permitido el acceso.
- El proceso intenta acceder a un recurso al que no tiene acceso, o intenta hacerlo de una forma improcedente, por ejemplo, intenta escribir en un fichero de sólo lectura.
- El proceso intenta realizar una operación aritmética no permitida.
- El proceso intenta ejecutar una instrucción no válida.
- El sistema operativo o el usuario pueden dar por terminado un proceso, por ejemplo, por la existencia de un interbloqueo.

Algunos sistemas operativos están diseñados de forma que al terminar un proceso terminan todos los que han sido generados por él. De este modo un proceso puede terminar porque lo pide el proceso que lo generó. Esto es útil ya que permite el cierre ordenado del sistema de una forma fácil, matando al proceso inicial que es el padre de todos los demás.

3.2.3. Estados de un proceso

A la hora de diseñar un sistema operativo hay que definir un modelo claro del comportamiento que van a tener los procesos en este sistema. Esto es lo que se conoce como **modelo de estados**; que nos indica en qué estados puede encontrarse un proceso, y qué razones pueden hacer que un proceso pase de un estado a otro. Estos cambios de estado vienen impuestos, en gran medida, por la competencia que se establece entre los procesos a la hora de

compartir los recursos del sistema. A continuación se describirán dos modelos de estados posibles.

3.2.3.1. Modelo de cinco estados

Podemos definir un modelo simple en el que los procesos pueden estar en cinco estados diferentes. Éstos podrían ser:

Nuevo Un proceso que acaba de ser creado pero no ha sido admitido al conjunto de procesos ejecutables por el sistema operativo.

Listo Están en este estado aquellos procesos que poseen todos los recursos necesarios para ejecutarse excepto el procesador.

Ejecución Se encuentra en este estado el proceso que tiene el control del procesador. En un ordenador con un solo procesador sólo podemos tener un proceso en este estado en un instante dado.

Bloqueado Los procesos en este estado no pueden ejecutarse hasta que no ocurra un cierto suceso, como por ejemplo la terminación de una operación de E/S.

Terminado Un proceso que ha sido sacado del conjunto de procesos ejecutables por el sistema operativo, bien porque se ha detenido o porque ha sido abortado por alguna razón.

Los estados nuevo y terminado son construcciones útiles para el manejo de procesos. Si consideramos que tanto la creación como la terminación de un proceso se realizan en dos etapas, tal como apuntamos en el apartado 3.2.1; el estado nuevo corresponderá a un proceso que acaba de ser definido, es decir, el sistema operativo ha creado las estructuras de datos necesarias para manejarlo, pero todavía no le ha asignado un espacio de direcciones, es decir, no se ha comprometido a ejecutarlo. De forma similar, un proceso sale del sistema en dos etapas; primero, el proceso termina su ejecución de forma normal o anormal y pasa al estado de terminado. En este punto, el proceso ya no se puede elegir para ser ejecutado. Sin embargo, el sistema operativo mantiene durante un tiempo las estructuras y otra información asociada al proceso. Esto proporciona tiempo para que ciertos procesos extraigan información sobre el proceso terminado. Una vez pasado este tiempo, el sistema operativo puede borrar los datos correspondientes al proceso. Ejemplos de

programas que pueden necesitar información sobre los procesos que terminan son los de contabilidad, que controlan el tiempo de CPU y otros recursos utilizados por el proceso con fines de facturación, asimismo programas que monitorizan el uso y rendimiento de la máquina pueden tomar información sobre cada proceso que se ejecuta.

Un **diagrama de transición de estados** es un grafo dirigido, cuyos nodos representan los estados que pueden alcanzar los procesos y las ramas representan los sucesos que hacen que un proceso cambie de un estado a otro. El correspondiente a nuestro modelo se representa en la figura 3.2, en ella podemos ver que las transiciones que pueden darse entre estados son las siguientes:

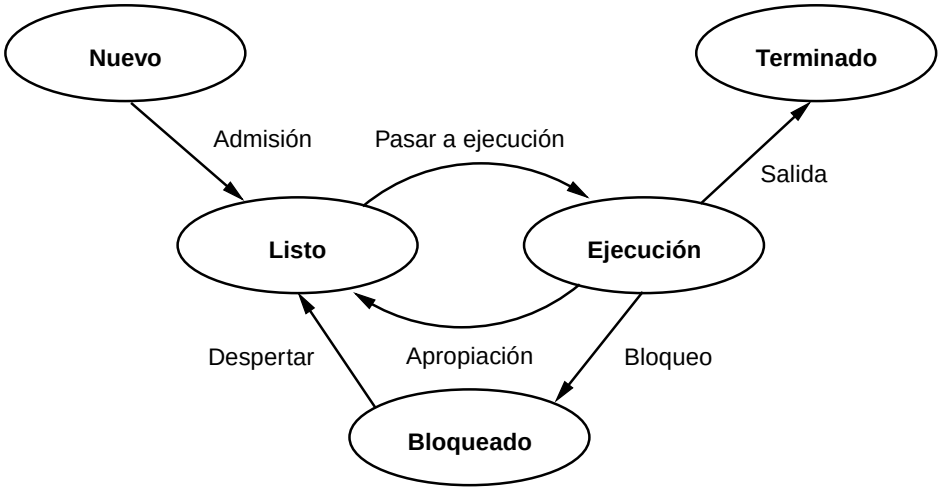

Figura 3.2: Modelo de procesos de cinco estados

Nada → nuevo Se crea un nuevo proceso para ejecutar un programa. Este suceso puede ocurrir porque un nuevo usuario inicia una sesión en un sistema de tiempo compartido, un proceso existente crea un proceso hijo, etc.

Nuevo → listo El sistema operativo pasará un proceso del estado de nuevo al de listo cuando esté preparado para encargarse de él. La cantidad de memoria principal disponible en un sistema representa un límite para el número de procesos activos que puede haber en él, de ahí, que el sistema no pueda admitir cualquier número de procesos.

Listo → ejecución Cuando el procesador queda libre habrá que seleccionar, entre los que se encuentran en estado listo, un nuevo proceso para ejecutarse. La cuestión de qué proceso es el que se elige se verá en el tema 4 en el que se tratará la planificación de procesos.

Ejecución → terminado El proceso que se está ejecutando actualmente finaliza su ejecución o bien es abortado debido a que se ha producido algún tipo de error.

Ejecución → listo El proceso que se está ejecutando actualmente pierde el control de la CPU para cedérsela a otro proceso. La razón más común para que se produzca esta transición es que el proceso en ejecución haya consumido el tiempo máximo de uso ininterrumpido del procesador. Si en un sistema el criterio para que un proceso tome el control de la CPU es su prioridad, el hecho de que llegue al sistema un proceso con mayor prioridad que el que se está ejecutando actualmente puede hacer que éste pierda el control de la CPU.

Ejecución → bloqueado Un proceso pasa al estado bloqueado si necesita algo sin lo que no puede continuar su ejecución. Por ejemplo, un proceso puede requerir un servicio del sistema operativo que éste no está preparado para realizar de inmediato. Puede pedir un recurso que no esté disponible inmediatamente, o bien, puede necesitar que se realice una operación de E/S que tardará un tiempo en llevarse a cabo.

Bloqueado → listo Un proceso en estado bloqueado pasa al estado listo cuando ocurre el suceso que estaba esperando.

Con este modelo simple ya podemos empezar a apreciar algunos de los elementos de diseño del sistema operativo. Éste debe ser capaz de seguirle la pista a cada proceso del sistema, es decir, debe mantener información sobre cada proceso, como por ejemplo el estado actual y su localización en memoria. Los procesos en estado listo y bloqueado deberían mantenerse en colas, esperando su turno para ejecutarse. En la figura 3.3 aparecen estas dos colas y las razones por las que los procesos se mueven entre ellas.

Como se puede observar tenemos dos colas, una corresponde a los procesos listos y la otra a los procesos bloqueados. Cuando el sistema admite un nuevo proceso, éste se coloca en la cola de procesos listos. Si el sistema tiene que elegir un proceso para ejecutarlo, selecciona uno de la cola de procesos listos. Cuando a un proceso que se está ejecutando se le retira el procesador, puede ser debido a varias causas: ha terminado su ejecución, por lo que saldrá del

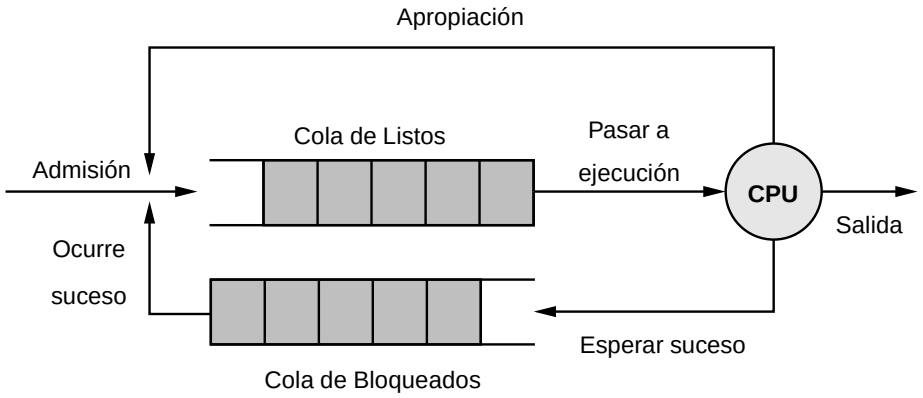

Figura 3.3: Modelo de colas para el diagrama de la figura 3.2

sistema, ha finalizado el tiempo máximo de uso de la CPU, en este caso pasará a la cola de procesos listos, o bien espera que ocurra algún suceso, en este caso será colocado en la cola de procesos bloqueados. Finalmente, cuando se produzca el suceso que está esperando, pasarán de la cola de procesos bloqueados a la de procesos listos todos aquellos procesos que estuvieran esperando dicho suceso.

Esto último implica que, cuando ocurre un suceso, el sistema operativo debe rastrear la cola de procesos bloqueados, en busca de aquellos que estaban esperándolo. En sistemas donde puede haber cientos de procesos bloqueados, podría ser más eficiente tener una cola por cada suceso. Así cuando ocurre cierto suceso, todos los procesos que están en la cola asociada a dicho suceso pueden pasar a estado listo.

Respecto al orden de los procesos en las colas, depende tanto del tipo de cola como de otros aspectos. Así, en el caso de la cola de listos, estará ordenada en función del criterio de selección de procesos, del que se hablará en el tema 4. Por el contrario, la cola de bloqueados no necesita un orden especial entre los procesos, ya que no se sabe en qué orden van a ocurrir los sucesos que están esperando.

La implementación de las colas de procesos listos y bloqueados se puede realizar mediante una lista enlazada de bloques de datos que mantiene el sistema operativo sobre cada proceso, esto se verá con más detenimiento en el apartado 3.4.

3.2.3.2. Estados suspendidos

El modelo de cinco estados visto anteriormente puede servir para construir un sistema operativo basado en él. Pero podemos considerar un modelo más complejo al que se añaden nuevos estados: los estados suspendidos (suspendido-listo y suspendido-bloqueado).

Un proceso suspendido es aquel que no está disponible de inmediato para su ejecución. Las razones por las que puede llegarse a esta situación pueden ser varias:

- Se pueden suspender procesos si se considera que la carga del sistema es alta. En este caso, se puede optar por sacar procesos de memoria principal y llevarlos a memoria secundaria (intercambio). Cuando la carga del sistema disminuya estos procesos podrán reanudarse, volviendo a memoria principal.

- Si un sistema está funcionando mal y es posible que falle, se puede querer suspender los procesos activos para reanudarlos cuando se haya corregido el problema.

- El sistema operativo puede suspender un proceso si sospecha que es el causante de un problema.

- Un proceso padre puede suspender un proceso hijo para coordinar su actividad con la de otros.

- Un usuario puede suspender un proceso si sospecha que los resultados parciales que está produciendo no son correctos. Cuando verifique si el proceso está funcionando correctamente o no, podrá proceder a reanudarlo o abortarlo.

- Consideraciones de temporización también pueden conducir a suspender un proceso. Por ejemplo, si un proceso se activa periódicamente pero está ocioso la mayor parte del tiempo, podría suspenderse mientras no esté activo. Un ejemplo de procesos de este tipo podría ser los que monitorizan la actividad de los usuarios.

En todos los casos, la activación de un proceso suspendido la pide el mismo agente que inicialmente solicitó la suspensión. Esta es una diferencia clara entre los procesos suspendidos y bloqueados, en este último caso el

proceso pasa al estado listo cuando ocurre el suceso que estaba esperando, no existe una activación.

Debido a esta diferencia entre suspensión y bloqueo, los procesos suspendidos pueden encontrarse en dos estados diferentes:

Suspendido-bloqueado Un proceso que está esperando a que ocurra un suceso y que además ha sido suspendido por alguna de las razones citadas. Ambas condiciones son independientes; si el suceso esperado ocurre, esto no habilita al proceso para su ejecución.

Suspendido-listo Un proceso que estaba en estado listo y ha sido suspendido, o bien, estaba en estado suspendido-bloqueado y ha ocurrido el evento que esperaba.

Al introducir los dos nuevos estados aparecen nuevas transiciones importantes (figura 3.4):

Bloqueado → suspendido-bloqueado Un proceso bloqueado es suspendido por algún agente: el sistema operativo, el proceso padre, etc.

Suspendido-bloqueado → suspendido-listo Esta transición se produce cuando ocurre el suceso que estaba esperando el proceso. Hay que hacer notar que esto requiere que la información de estado referente a los procesos suspendidos esté disponible para el sistema operativo.

Suspendido-listo → listo El agente que suspendió al proceso lo devuelve al estado listo.

Listo → suspendido-listo Un proceso que está en estado listo es suspendido por alguna de las razones citadas anteriormente.

Ejecución → suspendido-listo Se suspende el proceso que se está ejecutando actualmente.

3.3. Imagen de un proceso

Un proceso consta como mínimo de:

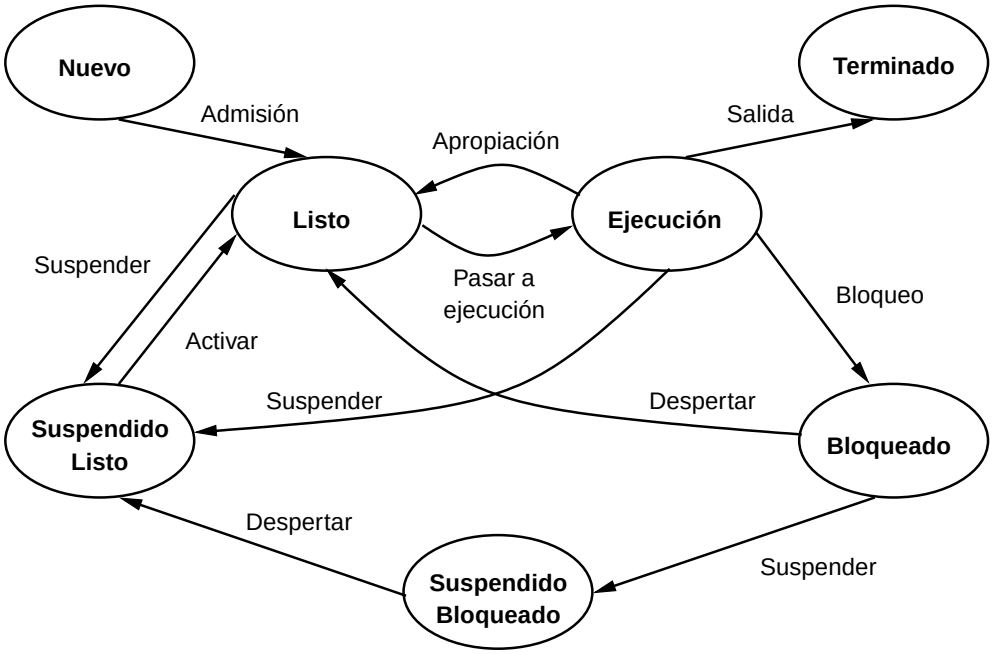

Figura 3.4: Diagrama de transición de estados con estados suspendidos

Programa de usuario El programa o conjunto de programas que van a ser ejecutados.

Datos del usuario Conjunto de localizaciones de memoria para almacenar las variables locales y globales, y las constantes definidas.

Pilas Cada proceso dispone de una pila del usuario que se utiliza para almacenar argumentos y otros datos relativos a funciones que se ejecutan en modo usuario. También tiene una pila del núcleo que se emplea en las llamadas al sistema y siempre que éste pasa de modo usuario a modo núcleo.

Bloque de control del proceso [2] Es el conjunto de atributos asociados al proceso que utiliza el sistema operativo para su control.

El conjunto formado por el programa, los datos, la pila y los atributos se suele denominar **imagen del proceso** (figura 3.5).

[2]Suele aparecer en la bibliografía de forma abreviada como BCP, o en su nomenclatura inglesa PCB (*Process Control Block*). También lo podemos encontrar con el nombre de descriptor del proceso.

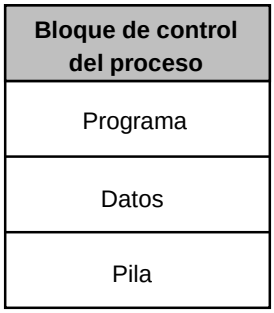

Figura 3.5: Imagen de un proceso

La forma en que se almacena en memoria la imagen de un proceso depende del esquema de gestión de la memoria que se esté usando. En el caso más simple, la imagen del proceso se mantiene como un bloque contiguo de memoria. Este bloque estará en memoria secundaria o en memoria principal, dependiendo del estado del proceso. En los sistemas operativos modernos la imagen del proceso consta de un conjunto de bloques que no necesitan estar cargados de forma contigua, permitiéndose que el proceso se ejecute sin que todos estos bloques estén cargados en memoria principal simultáneamente (sistemas de memoria virtual), es decir, sólo una parte de la imagen reside en memoria principal mientras el resto se encuentra en memoria secundaria. En cualquier caso, para que el sistema operativo pueda encargarse de controlar el proceso, al menos una pequeña porción de la imagen debe residir siempre en memoria principal, ésta es el bloque de control del proceso. El sistema debe mantener la información de dónde se encuentran en cada momento las distintas porciones de la imagen del proceso.

3.3.1. Bloque de control del proceso

En un sistema de tiempo compartido, se requiere una gran cantidad de información sobre cada proceso para su manejo. Podemos considerar que ésta reside en el bloque de control del proceso. Dado que esta información se puede organizar de distintas formas en diferentes sistemas, sólo nos ocuparemos del tipo de información que utiliza el sistema operativo y no de cómo se organiza ésta.

La información contenida en el bloque de control del proceso se puede agrupar en tres tipos:

- Identificación del proceso.
- Información del estado del procesador.
- Información de control del proceso.

Con respecto a la identificación del proceso, en casi todos los sistemas operativos se asigna a cada proceso un identificador numérico único. Éste se suele utilizar como índice de la tabla de procesos y también puede ser utilizado en las tablas de memoria, E/S y ficheros para indicar en qué zona de memoria reside el proceso, o bien qué dispositivos de E/S o ficheros tiene asignados. Cuando los procesos se comunican entre sí, su identificador sirve para informar al sistema operativo del destino del mensaje. Cuando a los procesos se les permite generar otros, los identificadores se usan para indicar quién es el padre y quiénes los descendientes.

Además de su propio identificador, a un proceso se le puede asignar un identificador de usuario que indica quién es el usuario responsable de él.

Otro conjunto importante de información es la referente al estado del procesador, que está constituida fundamentalmente por el contenido de los registros de éste. Mientras el proceso está en ejecución, los registros mantienen información; cuando el proceso deja de ejecutarse, toda la información contenida en ellos debe guardarse para que pueda ser recuperada cuando el proceso reanude su ejecución. Esta información incluye normalmente, los registros visibles al usuario, los registros de control y estado, y los punteros a las pilas.

El tercer conjunto de información dentro del bloque de control del proceso, está compuesto por la información adicional que necesita el sistema operativo para controlar y coordinar los distintos procesos activos; ésta incluye el estado del proceso, su prioridad, los recursos que tiene asignados, punteros que permiten crear listas enlazadas de los BCP, etc. La podemos denominar información de control del proceso. A medida que avancemos en el estudio de los sistemas operativos, se verá más clara la necesidad de estos elementos.

La tabla 3.1 lista los distintos tipos de información que requiere el sistema operativo para cada proceso.

Resumiendo, podemos decir que el bloque de control del proceso es la estructura de datos central y más importante en un sistema operativo. Cada uno de ellos contiene toda la información que éste necesita sobre un proceso.

Identificación del proceso

- Identificador del propio proceso[3].
- Identificador del proceso que lo creó (proceso padre)[4].
- Identificador del usuario responsable del proceso[5].

Información de estado del procesador

- Registros visibles al usuario.
- Registros de control y estado: contador de programa (PC), palabra de estado del proceso (PSW), ...
- Punteros a las pilas.

Información de control del proceso

- Información sobre la planificación del proceso: la información concreta a guardar dependerá del algoritmo de planificación utilizado.
- Comunicación entre procesos.
- Privilegios del proceso.
- Manejo de memoria: puntero al espacio de direcciones que ocupa el proceso.
- Propiedad de los recursos y utilización.
- Punteros a otros procesos.

Cuadro 3.1: Elementos típicos de un bloque de control de un proceso

[3]En la bibliografía en inglés suele aparecer como PID (*Process IDentifier*).
[4]En la bibliografía en inglés suele aparecer como PPID (*Parent Process IDentifier*).
[5]También suele aparecer como UID (*User IDentifier*)

Podríamos decir que el conjunto de bloques de control de los procesos define el **estado del sistema operativo**.

3.4. Control de los procesos

Como ya hemos comentado anteriormente, el sistema operativo se encarga de asignar el procesador a los procesos para su ejecución, les asigna recursos y responde a las peticiones de servicios que hacen éstos. Para realizar estas funciones el sistema operativo necesita mantener información acerca de los procesos que existen en el sistema. Para ello construye y mantiene tablas de información sobre cada una de las entidades que maneja. En general, suele mantener cuatro tipos de tablas con información sobre la memoria, E/S, ficheros y procesos. Aunque los detalles pueden variar de un sistema operativo a otro, prácticamente todos ellos mantienen estos cuatro tipos de información.

El sistema operativo debe mantener **tablas de procesos** para administrarlos. En la figura 3.6 se representa una posible implementación de ésta, con una entrada por cada proceso donde se almacena su bloque de control. El identificador del proceso constituye el índice en dicha tabla. El contenido de cada entrada puede variar de un sistema a otro. En algunos, en lugar de almacenar el bloque de control, cada entrada sólo contiene un puntero a la zona de memoria donde está ubicado.

El bloque de control del proceso puede contener punteros que permiten enlazarlos entre sí. Así, las colas que se describieron en el apartado 3.2 pueden ser implementadas como listas enlazadas de bloques de control de procesos, como se muestra en la figura 3.6, donde se refleja la cola de procesos listos, y la de bloqueados.

En la figura 3.6 también aparecen los registros del procesador. Así, el registro de índice del proceso contiene el identificador del proceso que está actualmente ejecutándose, es decir, su PID. El contador de programa apunta a la próxima instrucción a ejecutar. Los registros base y límite definen la región de memoria que ocupa el proceso. Tanto el contador de programa como las referencias a datos que se realizan se interpretan como relaciones relativas al registro base y no deben exceder el valor del registro límite. Esto permite que no existan interferencias entre los procesos. Todos estos registros se almacenan en el BCP cuando el proceso abandona el estado de ejecución.

Las **tablas de memoria** le permiten conocer el estado de la la memoria

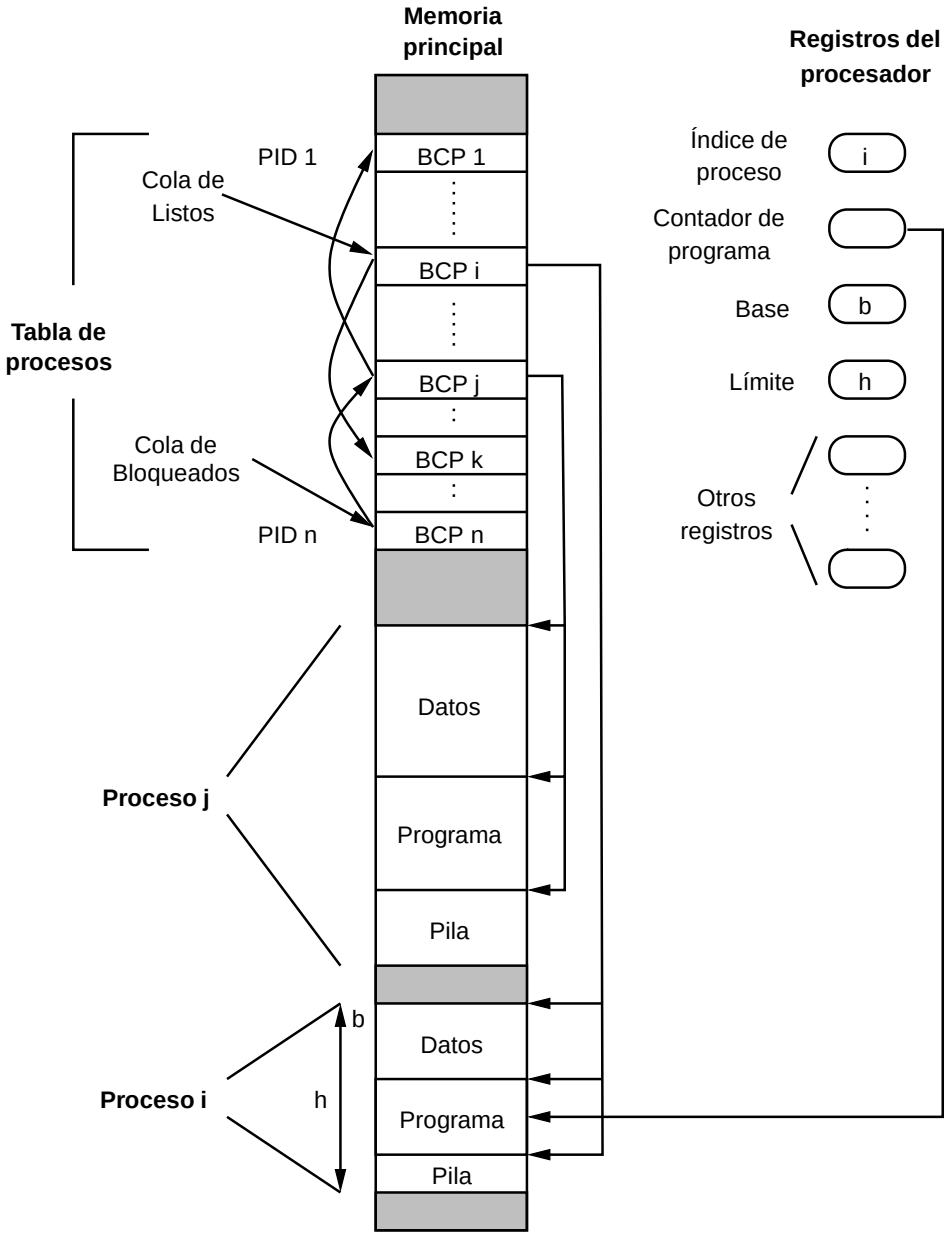

Figura 3.6: Implementación de la tabla de procesos

principal y de la secundaria. Suelen incluir la siguiente información:

- Memoria principal asignada a los procesos.
- Memoria secundaria asignada a los procesos.
- Atributos de protección de zonas de memoria principal y secundaria, por ejemplo, puede indicar qué procesos tienen acceso a una zona de memoria compartida.
- Cualquier otra información que se necesite para administrar la memoria.

Estas estructuras de información para el manejo de la memoria se verán con más detalle en el tema 7 y en la asignatura Sitemas Operativos II.

El sistema operativo utiliza las **tablas de E/S** para llevar el control de los dispositivos. En un instante dado, un dispositivo de E/S puede estar disponible o asignado a un proceso particular. Si se está realizando una operación de E/S, el sistema operativo necesita saber el estado de ésta y la localización en memoria principal que se está usando como fuente o destino de la transferencia de E/S. La gestión de la E/S se estudiará en la asignatura de Sistemas Operativos II.

Por último, las **tablas de ficheros** proporcionan información sobre qué ficheros existen, su localización en memoria secundaria, estado actual, y otros atributos.

Todas las tablas de las que hemos hablado (memoria, ficheros, E/S y procesos) deben estar, de alguna forma, relacionadas entre sí, ya que la memoria, los ficheros y los dispositivos de E/S se administran en nombre de los procesos. Las tablas de procesos deben contener referencias directas o indirectas a los recursos que están utilizando éstos.

Anteriormente se ha dicho que el sistema operativo crea y mantiene estas tablas. Nos podríamos preguntar ¿cómo sabe el tamaño deben tener? El sistema operativo debe tener algún conocimiento del ambiente básico, por ejemplo, cuánta memoria principal existe, qué dispositivos de E/S hay, de qué tipo son éstos, etc. En algunos sistemas, algunos de estos datos pueden ser especificados durante el proceso de configuración por el administrador, mientras que en otros están establecidos en el código fuente.

3.5. Gestión de procesos

Una de las funciones del núcleo de un sistema operativo es la gestión de los procesos. Las tareas que se engloban dentro de esta función son las siguientes:

- Creación y terminación de procesos.
- Cambio de procesos.
- Gestión de los bloques de control de los procesos.
- Planificación y despacho de procesos.
- Sincronización y soporte para la comunicación entre los procesos.

Una vez estudiados los estados de un proceso y las estructuras que mantiene el sistema para su control vamos a describir de forma más detallada los pasos que da el sistema para la creación de procesos. A continuación se tratará las operaciones implicadas en el cambio de proceso. La gestión de los bloques de control de los procesos hace referencia a todas las operaciones implicadas desde que se crean hasta que se destruyen, tales como el cambio de cola, modificación de la información referente al estado del procesador o al control del proceso. La función de planificación y despacho será tratada en el tema 4, mientras que la sincronización y comunicación entre procesos se estudiará en el tema 5.

3.5.1. Creación de procesos

En el apartado 3.2.1 hablamos de los sucesos que podían conducir a la creación de un nuevo proceso, a continuación vamos a describir más detalladamente los pasos que da el sistema para crearlos:

1. Asignar un identificador único al nuevo proceso. Se añade una nueva entrada a la tabla de procesos, que contiene una entrada por proceso.
2. Asignar un espacio de direcciones al proceso. Se debe asignar espacio para todos los elementos de la imagen del proceso. Por tanto, el sistema operativo debe conocer cuánto ocupará el espacio de direcciones privado del usuario (programa y datos) y la pila del usuario. Estos valores

se pueden asignar por omisión, teniendo en cuenta el tipo de proceso, o se pueden establecer teniendo en cuenta la petición del usuario cuando se crea el trabajo. Si un proceso es generado por otro, el proceso padre puede pasar al sistema operativo los valores necesarios como parte de la petición de creación del proceso. Si el nuevo proceso va a compartir algún espacio de direcciones ya existente, se deben establecer los enlaces apropiados. Por último, se debe asignar espacio para el bloque de control del proceso.

3. Inicializar el bloque de control del proceso. La porción de identificación del proceso contiene el identificador de éste más otros identificadores necesarios, tal como el del proceso padre. La porción de información del estado del procesador normalmente se inicializará con la mayor parte de sus entradas a cero, excepto el contador del programa (se establece al punto de entrada del programa) y los punteros a la pila del sistema. La porción de información de control del proceso se inicializa con los valores por omisión o con los valores pedidos por el proceso. Por ejemplo, el estado del proceso será inicializado como listo. La prioridad se puede establecer inicialmente al valor más bajo, a menos que se haga una petición explícita de un valor más alto. Inicialmente, el proceso no poseerá ningún recurso (dispositivos de E/S y ficheros), a no ser que se pidan explícitamente o sean heredados del padre.

4. Se debe introducir en la cola de procesos correspondiente a su estado inicial, estableciéndose los enlaces adecuados.

5. Si es necesario se crearán o expandirán otras estructuras de datos. Por ejemplo, si el sistema operativo mantiene un fichero de contabilidad para cada proceso con fines de facturación o de estudio del rendimiento del sistema, se deberá crear el fichero correspondiente.

3.5.2. Cambio de proceso

De una forma simple podemos decir que el **cambio de proceso** consiste en que el proceso que se está ejecutando actualmente pierde el control del procesador, lo toma el sistema operativo y éste le da el control a otro proceso. A la hora de estudiar el cambio de proceso hemos de tener en cuenta varias dos cuestiones:

1. ¿Qué sucesos pueden provocar un cambio de proceso?

2. ¿Qué debe hacer el sistema operativo con las distintas estructuras de datos que controla, para llevar a cabo un cambio de proceso?

Nos ocuparemos en primer lugar de los sucesos que pueden dar lugar a que un proceso pierda el control de la CPU y lo tome el sistema operativo. La tabla 3.2 muestra un resumen.

Suceso	Mecanismo
Externo a la ejecución de la instrucción actual.	Interrupción.
Error asociado a la ejecución de la instrucción actual.	Excepción.
Petición de un servicio.	Llamada al sistema o paso de mensajes.

Cuadro 3.2: Sucesos que interrumpen la ejecución de un proceso

Las **interrupciones** son uno de los sucesos que pueden provocar un cambio de proceso. Se pueden distinguir dos clases de interrupciones, las causadas por algún suceso externo e independiente del proceso que se está ejecutando, como por ejemplo una interrupción del reloj; y las que son debidas a que el proceso que se está ejecutando produce un error o condición de excepción. Las primeras se suelen conocer como interrupciones y las segundas como **excepciones**.

Cuando se produce una interrupción ordinaria, se transfiere el control en primer lugar a un manejador de interrupciones del sistema, que hace algunas tareas internas y luego le da el control a la rutina del sistema operativo relacionada con ese tipo particular de interrupción.

Cuando se produce una excepción, el sistema operativo determina si el error es fatal. Si es así, el proceso que se está ejecutando actualmente pasa al estado de terminado y continuaría con la ejecución de otro proceso. Si no, la acción del sistema operativo dependerá de la naturaleza del error y el diseño de aquel.

Por último, el sistema operativo puede ser activado por la petición de un servicio realizada por el programa que se está ejecutando. Por ejemplo, un proceso de usuario ejecuta una instrucción que pide una operación de E/S, como la apertura de un fichero. Esta petición produce una transferencia a una rutina que se encarga de proporcionar el servicio.

Cuando se produce una interrupción, una excepción o la petición de un servicio, se llevan a cabo las siguientes acciones:

1. El procesador se prepara para transferir el control a la rutina correspondiente. Para empezar, debe guardarse toda aquella información que pueda ser alterada por la ejecución de la rutina y que necesitaremos recuperar cuando se reanude la ejecución del proceso que hemos interrumpido, es decir, el **contexto del proceso**. Éste estará constituido por los registros del procesador y será guardado en la pila del núcleo. Esto es lo que se conoce como **cambio de contexto**.

2. A continuación establecerá el valor del contador del programa a la dirección de comienzo de la rutina adecuada, para así proceder a la lectura de la primera instrucción de ésta.

En la figura 3.7 un proceso de usuario es interrumpido después de la instrucción N. El contenido de todos los registros se colocan en la pila, se actualiza el puntero a la pila y el contador de programa apunta a la dirección de la rutina.

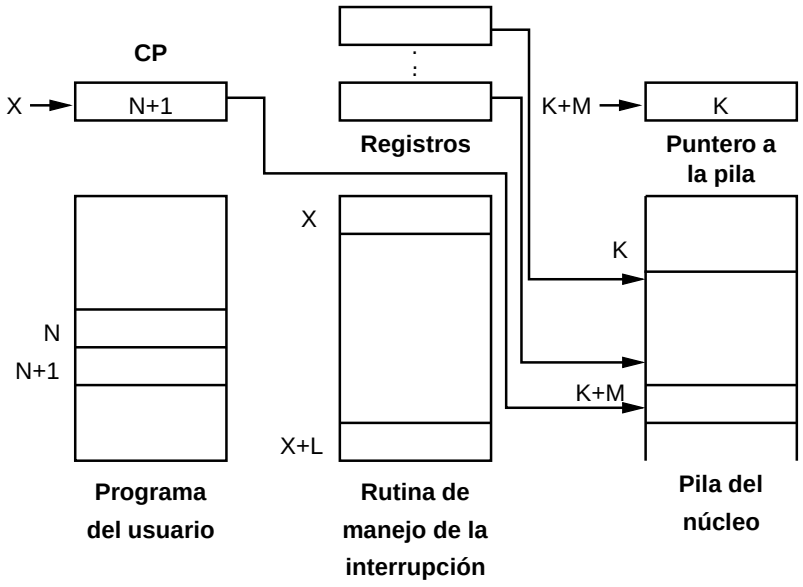

Figura 3.7: Cambios que se producen al atender una interrupción

¿Qué ocurre cuando termina la ejecución de la rutina? El sistema operativo debe decidir entre continuar la ejecución del proceso que se estaba ejecutando o sustituirlo por otro. Esto va a depender de la causa que motivó la interrupción de la ejecución del proceso.

Si se continúa con la ejecución del proceso interrumpido, se recuperarán los valores de los registros del procesador guardados en la pila, entre ellos el contador de programa. Esto va a permitir continuar con la ejecución del proceso en la instrucción siguiente donde fue interrumpido.

Si se sustituye el proceso que se estaba ejecutando por otro, se produce lo que se conoce como cambio de proceso. En este caso, habrán de realizarse las siguientes acciones:

1. Guardar el contexto del proceso, que se encuentra en la pila del núcleo, en su bloque de control.
2. Actualizar el estado del proceso interrumpido en su bloque de control. También deben ser actualizados otros campos importantes, incluyendo la razón por la que deja el estado de ejecución y la información de contabilidad.
3. Mover el bloque de control del proceso a la cola apropiada.
4. Seleccionar otro proceso para su ejecución. Esto se estudiará con más profundidad en el tema 4.
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 manejo de la memoria.
7. Restaurar el contexto del proceso seleccionado tal como estaba en el momento en que salió del estado de ejecución por última vez, cargando los valores previos del contador de programa y otros registros.

Vemos pues que el cambio de proceso requiere considerablemente más esfuerzo que el cambio de contexto. Cuando se produce un cambio de proceso, lo primero que se realiza es el cambio de contexto, y lo último, justo antes de otorgar el control del procesador al otro proceso, es restaurar su contexto.

Podemos decir, por tanto, que no es lo mismo un cambio de proceso que un cambio de contexto. Estos dos conceptos son tratados a menudo en la bibliografía como uno solo, denominándose cambio de contexto al cambio de proceso y no dándose ninguna denominación especial al primero.

Otra operación relacionada con las anteriores es el cambio de estado. Siempre que se produce un cambio de proceso se llevan a cabo dos cambios de estado, el proceso que estaba ejecutándose pasa a un estado distinto, y uno de los procesos en estado listo pasa a ejecución.

3.6. Hilos de ejecución

Hasta ahora se ha presentado el concepto de proceso como una entidad que reune las dos características siguientes:

Unidad propietaria de recursos Un proceso tiene asignado un espacio de direcciones para mantener su imagen, así como el control de otros recursos, tales como dispositivos de E/S y ficheros.

Unidad de despacho Un proceso es un camino o traza de ejecución a través de uno o más programas. Es la entidad que toma el control del procesador cuando el sistema operativo se lo cede. Un proceso tiene un estado y una prioridad de despacho.

En los sistemas operativos tradicionales estas dos características van unidas, hablándose de un **proceso pesado**. En sistemas más recientes se han separado, pasando a denominarse **hilo**[6] a la unidad de despacho, mientras que la unidad propietaria de los recursos se sigue denominando proceso. Esta separación permite que un proceso pueda tener asociados múltiples hilos, al contrario que en el enfoque tradicional donde podemos decir que el proceso pesado consta de único hilo. Sistemas operativos tales como OS/2, la versión de UNIX que hace Sun, Windows NT, el sistema Mach, o LINUX utilizan el modelo de múltiples hilos por proceso, denominándose sistemas multihilo.

En un ambiente multihilo, el proceso lleva asociado el espacio de direcciones de la imagen y los recursos asignados, tales como ficheros, dispositivos de E/S. Dentro de un proceso puede haber uno o más hilos, cada uno con su estado, su contexto, su pila, almacenamiento estático para variables locales y derechos de acceso a la memoria y a los recursos del proceso.

En la figura 3.8 se muestra la diferencia entre un proceso pesado y un proceso con múltiples hilos. Un proceso pesado se representa mediante su bloque de control y lleva asociados un espacio de direcciones y las pilas de usuario y núcleo. Sin embargo, en un modelo multihilo, el proceso tiene su bloque de control y su espacio de direcciones, mientras que cada uno de los hilos tiene también su propio bloque de control y las pilas de usuario y núcleo. La existencia de un bloque de control por hilo es necesaria debido a que cada hilo tiene su propio estado, prioridad y contexto.

[6] Traducción de la palabra inglesa *thread*. También se emplea el término **proceso ligero** para denominarlo.

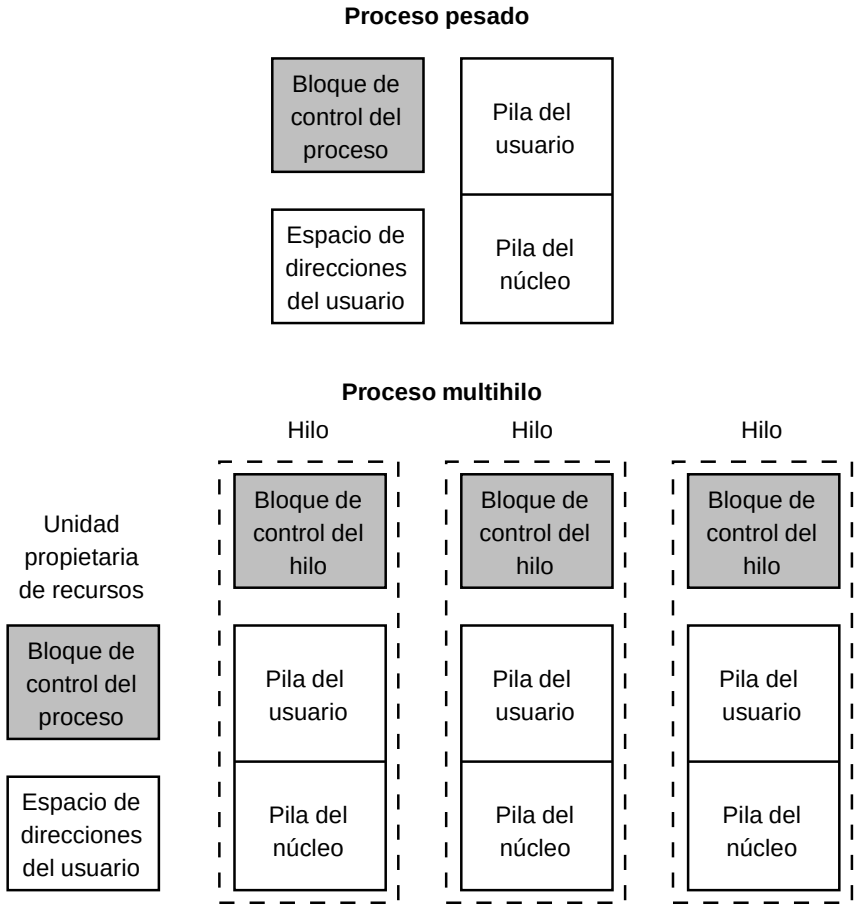

Figura 3.8: Proceso pesado frente a proceso multihilo

Los posibles estados en que puede encontrarse un hilo son ejecución, bloqueado y listo. Normalmente, no tiene sentido asociar estados suspendidos con los hilos, sino que éstos van asociados a los procesos. Por ejemplo, si se intercambia un proceso, todos sus hilos tendrán que estar necesariamente intercambiados puesto que comparten el espacio de direcciones del proceso. De forma similar, la terminación de un proceso implica la terminación de todos los hilos que lo componen.

3.7. Procesos en LINUX

El concepto de proceso en LINUX es el mismo que para cualquier otro sistema operativo. Al ser un sistema multiprogramado, puede haber muchos procesos residentes en el sistema de forma simultánea.

3.7.1. Imagen de un proceso

En LINUX, la imagen de un proceso está compuesta por los siguientes elementos:

El segmento de datos Contiene los datos y es exclusivo de cada proceso. Se divide en datos inicializados, es decir, datos cuya situación no cambia a lo largo de la ejecución del proceso y datos no inicializados. El tamaño de este segmento puede variar en tiempo de ejecución.

El segmento de texto o código Es una copia del bloque de texto del programa. Puede ser compartido por varios procesos y es de solo lectura. El texto, al igual que los datos inicializados, se toman del fichero ejecutable.

El segmento de pila Lo crea el núcleo al arrancar el proceso, gestionando dinámicamente su tamaño. La pila se compone de una serie de bloques lógicos, llamados **marcos de pila**, que se introducen cuando se llama a una función y se sacan cuando se vuelve de la función. Un marco de pila se compone de los parámetros de la función, las variables locales a ésta y la información necesaria para restaurar el marco de pila anterior a la llamada a la función (contador de programa y puntero de pila). El segmento de pila es exclusivo para cada proceso. Dado que los procesos se pueden ejecutar en dos modos: modo usuario y modo núcleo, el sistema maneja dos pilas por separado, la pila del usuario y la pila del núcleo.

Bloque de control del proceso Esta estructura de datos es la que contiene toda la información sobre el proceso, y permanece en memoria, incluso cuando éste no está residente en ella.

3.7.2. El bloque de control de procesos

En LINUX es una estructura compleja que contiene información sobre la identidad del proceso, estado, tiempos de ejecución, zona de memoria para el intercambio, punteros a la tabla de memoria, los ficheros y sistemas de ficheros asociados, hilos, procesador donde se ejecuta, etc. La información que contiene es la siguiente:

Identificadores Son los siguientes:

- Identificación del proceso (PID y GID) y del padre (PPID), que especifican las relaciones entre procesos.
- Identificación del usuario (UID y GID, reales y efectivos) para determinar los privilegios del proceso.

Información del procesador En el caso de ejecutarse en un entorno multiprocesador.

El contador de programa Contiene la dirección de la siguiente instrucción que debe ejecutar la CPU.

Registros de propósito general Contiene información útil para el funcionamiento del programa como mensajes de error, señales, etc.

Otros registros Almacenan el valor de los registros hardware en los cambios de contexto del proceso.

Puntero a la pila Contiene la dirección de la siguiente entrada en la pila del usuario o del núcleo, dependiendo del modo de ejecución en que estuviese en el momento de la apropiación.

Punteros a la imagen Sirve para localizar el proceso, tanto en memoria principal como en memoria secundaria.

Punteros a otras estructuras del sistema Contiene la información útil sobre los ficheros y sistemas de ficheros.

Parámetros para la planificación del proceso Tales como prioridad, tiempo consumido de CPU, tiempo de E/S, estado de espera, etc., y demás parámetros que se utilizan para determinar cuál es el siguiente proceso a ejecutarse.

Señales Enumera las señales que ha recibido el proceso y no ha tratado todavía. Las señales pueden estar activas o ser ignoradas.

Miscelánea Entre ellos se encuentran:

- Tamaño del proceso.
- Punteros a los bloques de control anterior y posterior en la tabla de procesos.
- Descriptores de señales a la espera.
- Punteros al proceso padre y al proceso hijo más joven.
- Indicador para saber si es intercambiable.
- Parámetros de entrada y de salida.

3.7.3. La tabla de procesos

El núcleo del sistema ve a un proceso como una entrada en la tabla de procesos que apunta a su bloque de control. El tamaño de ésta es fijo, por tanto, el número máximo de procesos que puede haber en el sistema está limitado. También se limita el número de procesos que puede crear un usuario; este límite se establece en la mitad del número de procesos totales. Por motivos de seguridad, mantiene como mínimo, cuatro entradas libres para el administrador del sistema. De este modo, si el sistema está colapsado en cuanto al número de procesos, el superusuario podrá crear cuatro procesos para poder realizar sus labores de administración.

3.7.4. Estados de un proceso

LINUX reconoce seis estados, empleando dos estados de ejecución para indicar si el proceso se está ejecutando en modo usuario o modo supervisor. La figura 3.9 muestra el diagrama de estados de LINUX, donde aparecen los siguientes estados:

Nuevo El proceso acaba de ser creado y está en un estado de transición; el proceso existe, pero no está preparado para ejecutarse. Este estado es el inicial para todos los procesos, excepto para el primero que se crea en el sistema, llamado proceso init.

Listo El proceso está esperando su turno para poder ejecutarse en la CPU. El planificador antes de escoger un proceso comprueba si alguno de los bloqueados puede pasar a este estado, y después escoge uno entre los procesos listos.

Ejecutándose en modo usuario

Ejecutándose en modo supervisor

Bloqueado El proceso se encuentra esperando una señal o evento para poder continuar con su ejecución. Diferencia dos tipos de bloqueos, uno causado por el funcionamiento del proceso y otro por algo ajeno a él, como por ejemplo cuando el usuario ordena pararlo. En este último estado, el proceso puede ser reiniciado externamente, o bien, terminar completamente cuando el proceso padre lo haga.

Terminado Es un estado intermedio en la terminación del proceso. Durante el tiempo que el proceso está en este estado envía al proceso padre un registro que contiene el código de salida y algunos datos estadísticos tales como los tiempos de ejecución.

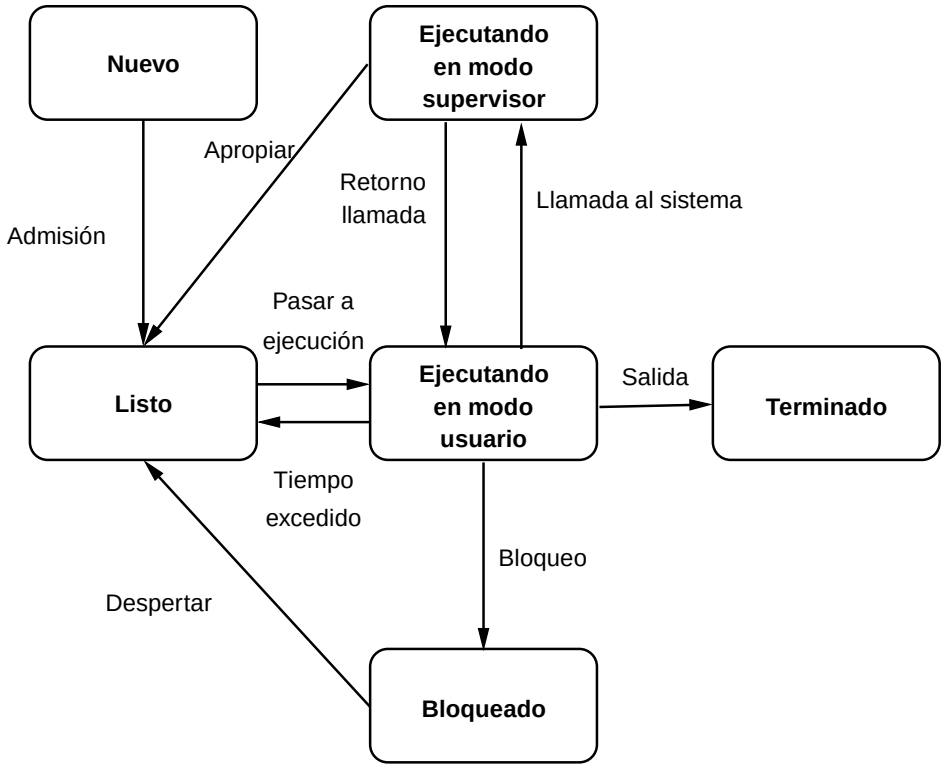

Figura 3.9: Diagrama de estados de un proceso en LINUX

3.7.5. Llamadas al sistema

LINUX dispone de una serie de llamadas al sistema relacionadas con la gestión de procesos, éstas aparecen en la tabla 3.3.

Nombre	Descripción
clone	Crea un contexto de ejecución.
fork	Crea un proceso.
execve	Ejecuta un programa.
wait	Espera la terminación de un proceso.
exit	Termina un proceso.

Cuadro 3.3: Llamadas al sistema relacionadas con procesos

La llamada fork sirve para crear un nuevo proceso. Ésta crea una copia exacta del proceso que realiza la llamada, éste es el proceso padre y el nuevo proceso que se crea es el hijo. Cuando se realiza una llamada fork el sistema realiza las siguientes operaciones:

- El núcleo busca una entrada libre en la tabla de procesos y la reserva.
- Si la encuentra, copia toda la información del proceso padre en la entrada del proceso hijo.
- Asigna un identificador al proceso hijo.
- Comparte inicialmente todos los segmentos del proceso padre bajo la técnica de **copia-en-escritura**[7].

La llamada execve ejecuta un programa, es decir, sustituye el código y los datos copiados del proceso padre por la nueva imagen contenida en el fichero que se le pasa como argumento. Es decir, carga un programa en la zona de memoria del proceso que la ejecuta, sobrescribiendo los segmentos del programa antiguo con los del nuevo.

Cuando un proceso termina voluntariamente hace la llamada al sistema exit, ésta tiene las siguientes consecuencias:

- Se descarga de memoria el contexto del proceso.

[7] Cuando se intente realizar una modificación en alguno de los segmentos, se reserva memoria para éste, se copia su contenido y posteriormente se modifica la copia.

- Se liberan los recursos asignados al proceso.

En la figura 3.10 se ve de forma gráfica la utilización de las llamadas fork, execve, wait y exit.

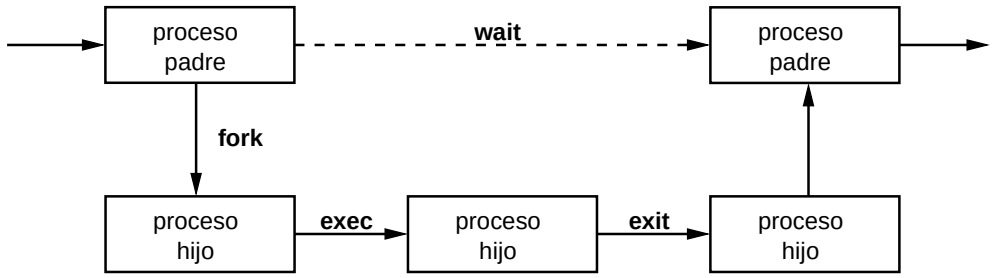

Figura 3.10: Sincronización entre proceso padre e hijo

3.7.6. Hilos

LINUX es un sistema multiproceso y multihilo, siguiendo el modelo visto en el apartado 3.6. Para hacer uso de los hilos se dispone de una biblioteca de funciones que nos permite, entre otras cosas, crearlos, sincronizarlos, etc. La tabla 3.4 nos muestra algunas de las funciones que proporciona.

Nombre	Descripción
pthread_create	Crea un hilo.
pthread_join	Espera la terminación del hilo dado como argumento.

Cuadro 3.4: Algunas funciones de LINUX relacionadas con los hilos

La función pthread_create crea un bloque de control para el nuevo hilo mediante la llamada clone. El nuevo hilo comparte todos los recursos asignados al proceso mediante punteros a las estructuras del proceso (figura 3.11).

3.8. Resumen

La unidad de funcionamiento del sistema es el proceso. Éste, al contrario que el programa, es dinámico, pasando durante su vida por una serie de

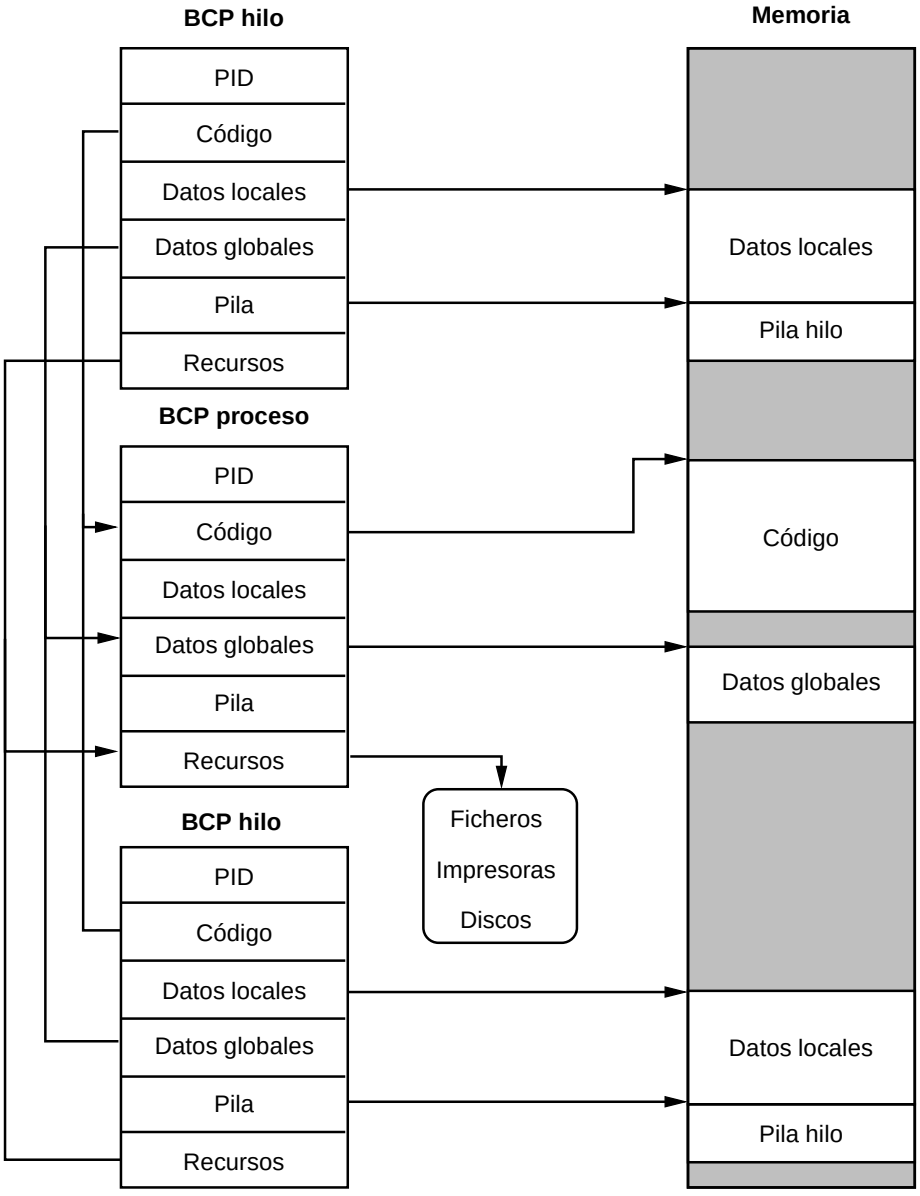

Figura 3.11: Relación entre el proceso y el hilo

estados. El conjunto de estados posibles en un sistema y las transiciones que pueden darse entre ellos definen el modelo de estados.

El sistema operativo se tiene que encargar de la gestión de los procesos, es decir, de su creación, planificación, terminación, cambios de estado, asignación de recursos, etc. Para ello requiere de una serie de tablas, de procesos, memoria, E/S y ficheros. La estructura que mantiene toda la información que necesita el sistema sobre cada proceso es el bloque de control.

El sistema operativo se encarga de realizar dos operaciones fundamentales: el cambio de contexto y el cambio de proceso. El primero es el elemento principal en la multiprogramación encargándose de guardar los registros de la CPU, para posteriormente cuando vuelva a tomar la CPU pueda continuar con su ejecución en el mismo punto por donde estaba. El segundo permite que cuando un proceso esté esperando algún evento se le pueda retirar la CPU y asignarla a otro.

Los sistemas operativos más recientes han introducido un nuevo concepto, el hilo. Éste permite separar las dos características englobadas en el concepto tradicional de proceso, la unidad propietaria de los recursos y la unidad de despacho.