5. Estructura de un proceso

5.1 Programas y procesos

Un programa es una colección de instrucciones y de datos que se encuentran almacenados en un archivo ordinario. Este archivo tiene en su nodo-i un atributo que lo identifica como ejecutable. Puede ser ejecutado por el propietario, el grupo y el resto de los usuarios, dependiendo de los permisos que tenga el archivo.

Los usuarios pueden crear archivos ejecutables de varias formas. Una de las sencillas es mediante la escritura de programa para el intérprete de órdenes (scripts de shell). Mediante este mecanismo se deben realizar dos pasos para obtener un programa. El primero es editar un archivo de texto que contenga una serie de líneas que puedan ser interpretadas por un intérprete de órdenes (`sh`, `csh` o `ksh`). Una vez editado el programa, hay que cambiar sus atributos para que sea ejecutable; esto se consigue con la orden `chmod`. Para poder ejecutar un archivo así creado, tendremos que arrancar un intérprete de órdenes, pasándole como parámetro el nombre de nuestro archivo de órdenes. Como, por defecto, al iniciar una sesión bajo UNIX, se ejecuta un intérprete para dar servicio al usuario, bastará con escribir el nombre del archivo de órdenes para que empiece a ejecutarse.

Trabajar con archivos de órdenes presenta grandes ventajas a la hora de realizar programas cortos que no sean de gran complejidad, pero es una seria limitación a la hora de afrontar el desarrollo de una aplicación de mayor tamaño. Para ello, en la mayor parte de las ocasiones, vamos a generar archivos ejecutables mediante la ayuda de lenguajes de alto o bajo nivel. En nuestro caso estamos empleando el compilador de lenguaje C.

Este mecanismo de generación ya ha sido estudiado, pero para ser exhaustivos lo vamos a exponer. Primero se debe crear un archivo de texto que contenga el código fuente de nuestro programa (este archivo tendría una extensión `.c`). El compilador de C (`gcc`) se va a encargar de traducir el código fuente a código objeto que entiende nuestra computadora.

La estructura de todo programa ejecutable creado por el compilador de C viene impuesta por el sistema. Para ver una descripción detallada de esta estructura, podemos consultar la entrada `a.out` del manual de UNIX. A grosso modo, un programa consta de las siguientes partes:

- Un conjunto de encabezados que describen los atributos del archivo.
- Un bloque donde se encuentran las instrucciones en lenguaje máquina del programa. Este bloque se conoce en UNIX como texto del programa.
- Un bloque dedicado a la representación en lenguaje máquina de los datos que deben ser inicializados cuando arranca la ejecución del programa. Aquí está incluida la indicación de cuánto espacio de memoria debe reservar el kernel para estos datos. Tradicionalmente, este bloque se conoce como `bss` (seudo-operador del ensamblador de IBM 7090 que significa `block started by symbol`). El kernel inicializa, en tiempo de ejecución, esta zona a valor 0.
- Otras secciones, tales como tablas de símbolos.

Cuando un programa es leído del disco por el kernel y es cargado en memoria para ejecutarse, se convierte en un proceso. En un proceso no sólo hay una copia del programa, sino que además el kernel le añade información adicional para poder manejarlo.

Un proceso se compone de tres bloques fundamentales que se conocen como segmentos. Estos bloques son:

- El segmento de texto. Contiene las instrucciones que entiende el CPU. Este bloque es una copia del bloque de texto del programa.
- El segmento de datos. Contiene los datos que deben ser inicializados al arrancar el proceso. Si el programa ha sido generado por un compilador de C, en este bloque estarán las variables globales y las estáticas. Se corresponde con el bloque `bss` del programa.
- El segmento de pila. Lo crea el kernel al arrancar el proceso y su tamaño es gestionado dinámicamente por el kernel. La pila se compone de una serie de bloques lógicos, llamados marcos de pila, que son introducidos cuando se llama a una función y son sacados cuando se vuelve de la función. Un marco de pila se compone de los parámetros de la función, las variables locales de la función y la información necesaria para restaurar el marco de pila anterior a la llamada a la función (dentro de esta información se incluyen el contador de programa y el apuntador de pila anteriores a la llamada a la función). En los programas fuente no se incluye código para gestionar la pila (a menos que estén escritos en ensamblador); es el compilador quien incluye el código necesario para controlarlo.

Debido a que los procesos se ejecutan en dos modos: usuario y supervisor (también conocido como modo kernel); el sistema maneja dos pilas por separado. La pila del modo usuario contiene los argumentos, variables locales y otros datos relativos a funciones que se ejecutan en modo usuario, y la pila del modo supervisor contiene los marcos de pila de las funciones que se ejecutan en modo supervisor (estas funciones son las llamadas al sistema).

UNIX es un sistema que permite multiproceso (ejecución de varios procesos simultáneamente). El planificador es la parte del kernel encargada de gestionar el CPU y determinar qué proceso ocupa el tiempo de CPU en un determinado instante.

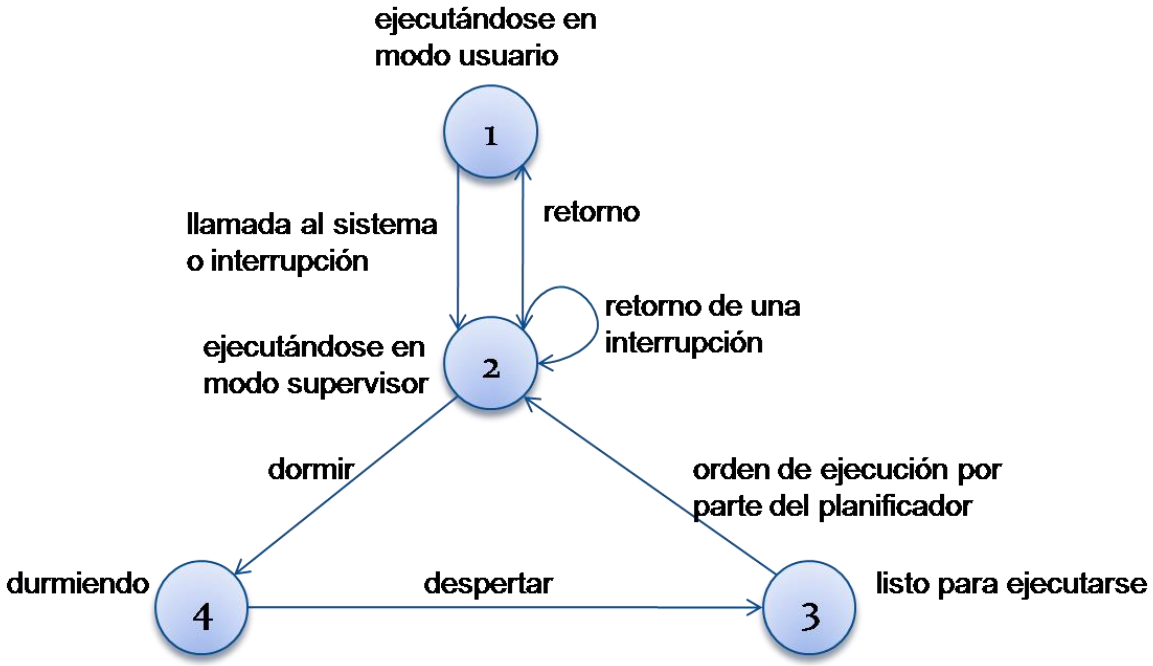
Un mismo programa puede estar siendo ejecutado en un instante determinado por varios procesos a la vez.

Desde un punto de vista funcional, un proceso en UNIX es la entidad que se crea tras la llamada `fork`. Todos los procesos, excepto el primero (proceso número 0), son creados mediante una llamada a `fork`. El proceso que llama a `fork` se conoce como proceso padre y el proceso creado es el proceso hijo. Todos los procesos tienen un único proceso padre, pero pueden tener varios procesos hijos. El kernel identifica cada proceso mediante su `PID` (process identification), que es un número asociado a cada proceso y que no cambia durante el tiempo de vida de éste.

El proceso 0 es especial; es creado cuando arranca el sistema, y después de hacer una llamada `fork` se convierte en el proceso intercambiador (encargado de la gestión de la memoria virtual). El proceso hijo creado se llama `init` y su `PID` vale 1. Este proceso es el encargado de arrancar los demás procesos del sistema según la configuración que se indica en el archivo `/etc/inittab`.

5.2 Estado de un Proceso

El tiempo de vida de un proceso se puede dividir en un conjunto de estados, cada uno con unas características determinadas. En la siguiente figura podemos ver, en un primer nivel de aproximación, los estados por los que evoluciona un proceso.



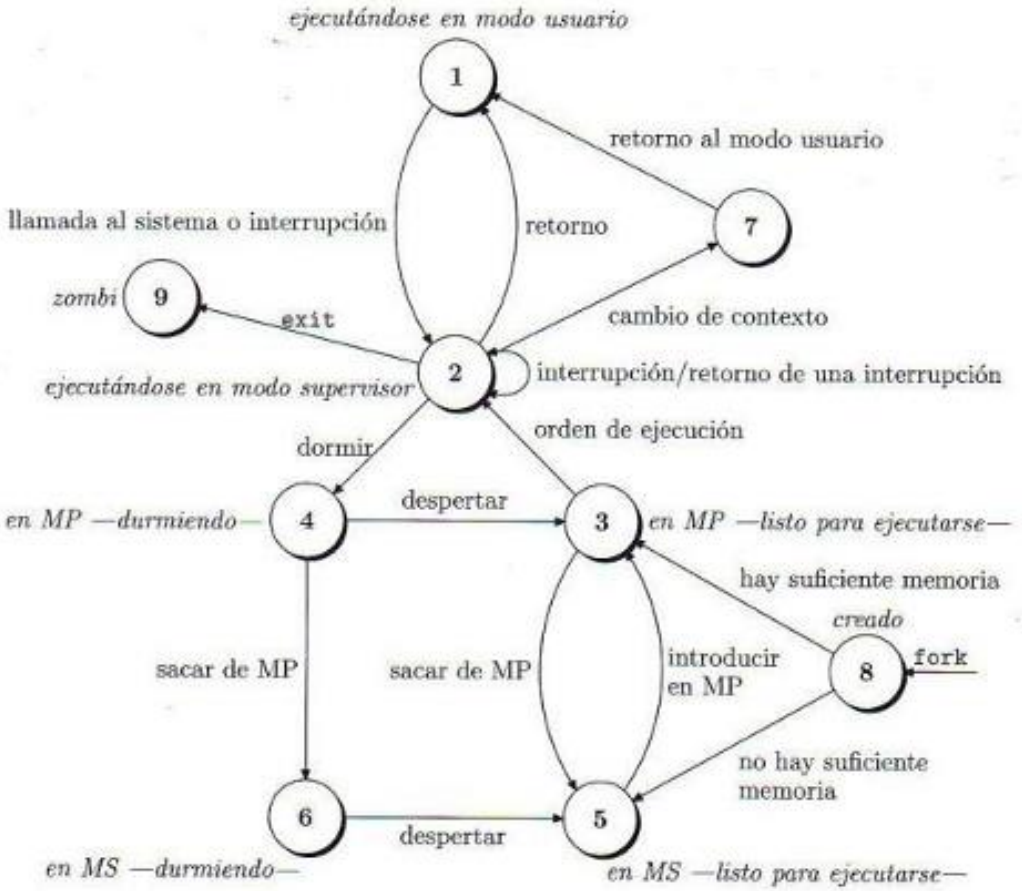
Estos son:

1. El proceso se está ejecutando en modo usuario.
2. El proceso se está ejecutando en modo supervisor.
3. El proceso no se está ejecutando, pero está listo para ser ejecutado cuando lo indique el planificador de tareas. Puede haber varios procesos simultáneos en este estado.
4. El proceso está durmiendo. Un proceso entra en este estado cuando no puede proseguir su ejecución porque está esperando a que se complete una operación de entrada/salida.

En un sistema monoprocesador no puede haber más de un proceso ejecutándose a la vez. Así, en los dos modos de ejecución (usuario y supervisor: estados 1 y 2) sólo podrá haber un proceso en cada estado.

Un proceso no permanece siempre en un mismo estado, sino que está continuamente cambiando de acuerdo con unas reglas bien definidas. Estos cambios de estado vienen impuestos por la competencia que existe entre los procesos para compartir un recurso escaso como es el CPU. La transición entre los cuatro estados descritos también queda definida en la figura anterior, que en este sentido es un diagrama de transición de estados. Un diagrama de transición de estados es un grafo dirigido, cuyos nodos representan los eventos que hacen que un proceso cambie de un estado a otro.

Si bien la figura anterior muestra los cuatro estados básicos por los que puede pasar un proceso, el diagrama de estados para los procesos de UNIX es bastante más complejo. En la siguiente figura podemos ver el diagrama completo y los estados que en él se reflejan son:



1. El proceso se está ejecutando en modo usuario.
2. El proceso se está ejecutando en modo supervisor.
3. El proceso no se está ejecutando, pero está listo para ejecutarse tan pronto como el kernel lo ordene.
4. El proceso está durmiendo cargado en memoria.
5. El proceso está listo para ejecutarse, pero el intercambiador (proceso 0) debe cargar el proceso en memoria antes de que el kernel pueda ordenar que pase a ejecutarse.
6. El proceso está durmiendo y el intercambiador ha descargado el proceso hacia una memoria secundaria (área de intercambio del disco) para crear espacio en la memoria principal donde poder cargar otros procesos.
7. El proceso está regresando del modo supervisor a modo usuario, pero el kernel se apropia del proceso y hace un cambio de contexto, pasando otro proceso a ejecutarse en modo usuario.
8. El proceso acaba de ser creado y está en un estado de transición; el proceso existe, pero ni está preparado para ejecutarse (estado 3), ni durmiendo (estado 4). Este estado es el inicial para todos los procesos, excepto el proceso 0.

9. El proceso ejecuta la llamada `exit` y pasa el estado zombi. El proceso ya no existe, pero le deja a su proceso padre un registro que contiene el código de salida y algunos datos estadísticos tales como los tiempos de ejecución. El estado de zombi es el estado final de un proceso.

5.3 Tabla de Procesos y Área de Usuario

Todo proceso tiene asociadas una entrada en la tabla de procesos y un área de usuario. Estas son dos estructuras que van a describir el estado del proceso y que le van a permitir al kernel su control.

La tabla de procesos tiene campos que van a ser accesibles desde el kernel, pero los campos del área de usuario sólo necesitan ser visibles por el proceso.

Las áreas de usuario se reservan cuando se crea un proceso y no es necesario que una entrada de la tabla de procesos que no aloja ningún proceso tenga reservada un área de trabajo.

Los campos que tiene cada una de las entradas de la tabla de procesos son los siguientes:

- Campo de estado que identifica el estado del proceso.
- Campos para localizar el proceso y su área de usuario, tanto en memoria principal como en memoria secundaria. El kernel usa esta información para realizar los cambios de contexto cuando el proceso pasa de un estado a otro. También utiliza esta información cuando traslada el proceso de la memoria principal al área de intercambio, y viceversa. En estos campos también hay información sobre el tamaño del proceso, para que el kernel sepa cuánto espacio de memoria debe reservar para él.
- Algunos identificadores de usuario (`UID`) para determinar los privilegios del proceso. Por ejemplo, el campo `UID` delimita qué procesos puede enviar señales al proceso actual y qué procesos pueden enviárselas a él.
- Identificadores de proceso (`PID`) que especifican las relaciones entre procesos. Estos identificadores son fijados cuando se crea el proceso mediante una llamada a `fork`.
- Descriptores de eventos, cuando el proceso está durmiendo y que serán utilizados al despertar.
- Parámetros de planificación que permiten al kernel determinar el orden en que los procesos pasan del estado ejecutándose en modo supervisor a ejecutándose en modo usuario.
- Un campo de señales que enumera las señales que han sido recibidas, pero que no han sido tratadas todavía.
- Algunos temporizadores que indican el tiempo de ejecución del proceso y el tiempo de uso de los recursos del kernel. Estos campos se usan para llevar control de los procesos y calcular la prioridad dinámica que se le asigna. Aquí también hay un temporizador que puede programar el usuario y que se usa para enviarle al proceso la señal de `SIGALRM`.

El área de usuario contiene información que es necesaria sólo cuando el proceso se está ejecutando. Algunos de los campos de que consta esta zona son los siguientes:

- Apuntador a la entrada de la tabla de procesos correspondiente al proceso al cual pertenece el área de usuario.

- Los identificadores de usuario real y efectivo (`UID`), que te determinan algunos privilegios del proceso, tales como el permiso de acceso a un archivo.
- Temporizadores que registran el tiempo empleado por el proceso ejecutándose en modo usuario y en modo supervisor.
- Un arreglo que indica cómo va a responder el proceso a las señales que recibe.
- Un registro que indica los errores que se han producido durante alguna llamada al sistema.
- Un campo de valor de retorno que contiene el resultado de las llamadas efectuadas por el proceso.
- Parámetros de entrada/salida que describen la cantidad de datos a transferir, la dirección del arreglo origen o destino de la transferencia y los apuntadores de lectura/escritura del archivo al que se refiere la operación de entrada.
- El directorio de trabajo actual y el directorio raíz asociados al proceso.
- La tabla de descriptores de archivo que identifica los archivos que el proceso tiene abiertos.
- Campos de límite que restringen el tamaño del proceso y de algún archivo sobre el que puede escribir.
- Una máscara de permisos que va a ser usada cada vez que se crea un nuevo archivo.

5.4 Contexto de un Proceso

El contexto de un proceso es su estado, definido por su código, los valores de sus variables globales y sus estructuras de datos, el valor de los registros del CPU, los valores almacenados en su entrada de la tabla de procesos y en su área de usuario, y el valor de sus pilas de usuario y supervisor. El código del sistema operativo y sus estructuras de datos globales, son compartidos por todos los procesos, pero no son considerados parte del contexto del proceso.

Cuando se ejecuta un proceso, se dice que el sistema se está ejecutando en el contexto de un proceso. Cuando el kernel decide que debe ejecutar otro proceso, realiza un cambio de contexto, lo que da lugar a que el kernel guarde la información necesaria para poder continuar con la ejecución del proceso interrumpido en el mismo punto donde se quedó. De igual manera, cuando el proceso cambia del modo usuario al modo supervisor, el kernel guarda información para cuando el proceso tenga que volver al modo usuario. Sin embargo, el cambio de modo usuario a supervisor y viceversa, no se contempla como un cambio de contexto.

Desde un punto de vista formal, el contexto de un proceso es la unión de su contexto del nivel usuario, su contexto de registro y su contexto de nivel de sistema.

El contexto del nivel usuario se compone de los segmentos de texto, datos y pila del proceso, así como de las zonas de memoria compartida que se encuentran en la zona de direcciones virtuales del proceso. Las partes del espacio de direcciones virtuales que periódicamente no residen en memoria principal debido al intercambio o a la paginación, también son parte del contexto del nivel de usuario.

El contexto de registros se compone de las siguientes partes:

- El contador de programa que contiene la dirección de la siguiente instrucción que debe ejecutar el CPU. Esta dirección es una dirección virtual, tanto en modo usuario como en modo supervisor.
- El registro de estado del procesador que especifica el estado del hardware de la computadora. El registro de estado normalmente contiene campos para indicar que el resultado de la última operación ha sido cero, positiva, negativa, si ha habido desbordamiento, acarreo, etc. Las operaciones que causan la modificación del registro de estado son realizadas en un determinado proceso, por lo tanto, este registro contiene el estado del hardware en relación con ese proceso. Otros campos importantes son los que indican el nivel de ejecución actual del proceso (en relación con las interrupciones) y el modo de ejecución (supervisor o usuario). El campo que indica el modo de ejecución determina si un proceso puede ejecutar instrucciones privilegiadas y si un proceso puede acceder al espacio de direcciones del kernel.
- El apuntador de la pila contiene la dirección de la siguiente entrada en la pila de usuario o supervisor. La arquitectura de la computadora dictará si el apuntador de la pila señala a la siguiente entrada libre o a la última entrada usada. De igual forma, es la arquitectura la que impone si la pila crece hacia direcciones altas o bajas de memoria.
- Los registros de propósito general contienen datos generados por el proceso durante su ejecución.

El contexto de nivel de sistema de un proceso tiene una parte estática (los tres primeros campos de la siguiente lista) y una parte dinámica (los dos últimos campos). Todo proceso tiene una única parte estática del contexto del nivel de usuario, pero puede tener un número variable de partes dinámicas. La parte dinámica es vista como una pila de capas de contexto que el kernel puede introducir y sacar según los eventos que se produzcan.

Las partes del contexto del nivel del sistema son las siguientes:

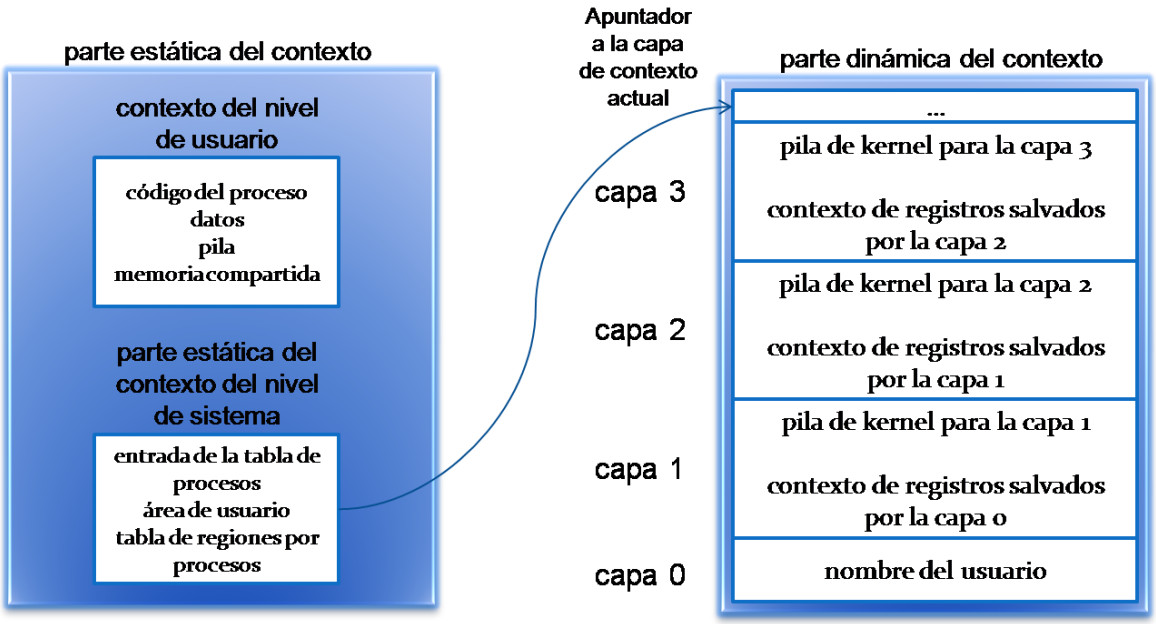
- La entrada en la tabla de procesos. Define el estado del proceso y contiene la información de control que es siempre accesible al kernel.
- El área del usuario. Contiene información de control del proceso que necesita ser accedida sólo en el contexto del proceso.
- Entradas de la tabla de regiones por proceso (`pregion`), tabla de regiones y tabla de páginas, que definen el mapa de transformaciones entre las direcciones del espacio virtual y las direcciones físicas. Si varios procesos comparten regiones comunes, estas regiones también son consideradas parte del contexto de cada proceso, ya que cada proceso las manipula independientemente. Es trabajo del módulo de gestión de memoria indicar qué partes del espacio de direcciones virtuales de un proceso no están cargadas en memoria.
- La pila de supervisor, que contiene los marcos de pila de las funciones ejecutadas en modo supervisor. Si bien todos los procesos ejecutan el mismo código del kernel, hay una copia privada de la pila del kernel para cada uno de ellos que da cuenta de las llamadas que cada proceso hace a las funciones del kernel. Por ejemplo, un proceso puede ejecutar la llamada `creat` y ponerse a dormir hasta que se le asigne un nodo-i, y otro proceso puede invocar la llamada `read` y dormir hasta que se efectúe la transferencia entre el disco y la memoria. Ambos procesos están ejecutando funciones del kernel, pero tienen pilas

separadas que contienen la secuencia de llamadas a funciones que ha realizado cada uno. El kernel debe ser capaz de recuperar el contenido de la pila del modo supervisor y la posición del apuntador de la pila para reanudar la ejecución de un proceso en modo supervisor. La pila del kernel está vacía cuando el proceso se está ejecutando en modo usuario.

- La parte dinámica del contexto del nivel de sistema se compone de una serie de capas que se almacenan al modo de pila. Cada capa contiene la información necesaria para poder recuperar la capa anterior, incluyendo el contexto de registro de la capa anterior.

El kernel introduce una capa de contexto cuando se produce una interrupción, una llamada al sistema o un cambio de contexto. Las capas de contexto son extraídas de la pila cuando el kernel vuelve del tratamiento de una interrupción, el proceso vuelve al modo usuario después de ejecutar una llamada al sistema o cuando se produce un cambio de contexto. La capa introducida es la del último proceso que se estaba ejecutando y la extraída es la del proceso que se parará a ejecutar. Cada una de las entradas de la tabla de procesos contiene información suficiente para poder recuperar las capas de contexto.

La siguiente figura muestra los componentes que forman el contexto de un proceso. En el lado izquierdo tenemos la parte estática del contexto, y en el lado derecho, la parte dinámica.



Un proceso se ejecuta en su contexto, o más exactamente, en su capa de contexto actual. El número de capas de contexto está limitado por el número de niveles de interrupción que soporte la computadora. Por ejemplo, si una computadora soporta 5 niveles de interrupción (interrupciones de software, de terminales, de disco, de varios periféricos y del reloj), un proceso tendrá al menos 7 capas de contexto: una por cada nivel de interrupción, una para las llamadas al sistema y otra para el nivel de usuario. Estas 7 capas son suficientes para contener todas las capas posibles, incluso si las interrupciones se dan en la secuencia más desfavorable, ya que las

interrupciones de un nivel determinado son bloqueadas (el CPU no las atiende) mientras que el kernel está atendiendo una interrupción de nivel igual o superior.