Tema 2

Estructura y funciones de los sistemas operativos

El sistema operativo es un software muy complejo e importante para los sistemas de computación. Veremos como podemos estructurarlo para facilitar su diseño e implementación. Relacionada con esta estructura están las distintas funciones que realiza, con el objetivo de abstraer el hardware y gestionar los recursos del sistema. Por este motivo, el sistema operativo proporciona un conjunto de servicios. En este capítulo veremos cuáles son y las distintas formas de implementarlos.

2.1. Funciones y componentes de los sistemas operativos

Para conseguir los objetivos principales de un sistema operativo, gestionar los recursos y abstraer el hardware, éste realiza una serie de funciones que

se encargan de proporcionar un ambiente para la ejecución de programas. Éstas pueden variar de un sistema a otro, sin embargo, es posible encontrar o diferenciar las siguientes:

Gestión de los procesos y los recursos Un programa cuando se ejecuta se convierte en uno o más **procesos**. El concepto de proceso es clave dentro del sistema operativo, de ahí que ésta sea la función principal del sistema. Los recursos son elementos que necesitan los programas para poder ejecutarse. El sistema define el ambiente de ejecución de los procesos, permitiendo que éstos utilicen los distintos recursos y los compartan. Por lo tanto, se encargará de la creación, eliminación, sincronización, comunicación entre procesos, asignación de recursos, etc. El componente del sistema encargado de realizar esta función es el denominado **administrador de procesos**.

Gestión de la memoria La memoria principal es un recurso necesario para que un programa pueda ejecutarse. El sistema operativo, mediante el **administrador de la memoria**, se encarga de asignar memoria a los procesos de usuario, y de controlar qué zona ocupan.

Gestión del sistema de E/S El **administrador del sistema de E/S** permite al usuario realizar operaciones de E/S sin conocer las particularidades de los diferentes dispositivos. Éste es uno de los objetivos principales del sistema operativo: abstraer el hardware.

Gestión de los ficheros Se encarga de almacenar la información en los dispositivos de almacenamiento, abstrayendo las características de éstos en un concepto lógico como el **fichero**. Esta función se realiza a través del **administrador de ficheros** que se encarga de su creación, borrado, lectura, etc.

Gestión de la red Esta función dependerá del tipo de sistema operativo. En el caso de los sistemas distribuidos, es el propio sistema el que debe encargarse de la comunicación a través de la red de los programas de usuario que se ejecutan en distintos equipos. En sistemas no distribuidos, esta función la realiza un software independiente del sistema.

Protección del sistema Un sistema de computación debe ser seguro, no sólo en cuanto a los usuarios que lo puedan utilizar, sino también a la seguridad en la ejecución de los programas. Así, se deben proteger las zonas de memoria de los distintos procesos de usuario, el acceso a los

diferentes recursos, etc. Esta función viene realizada por el denominado **sistema de protección**.

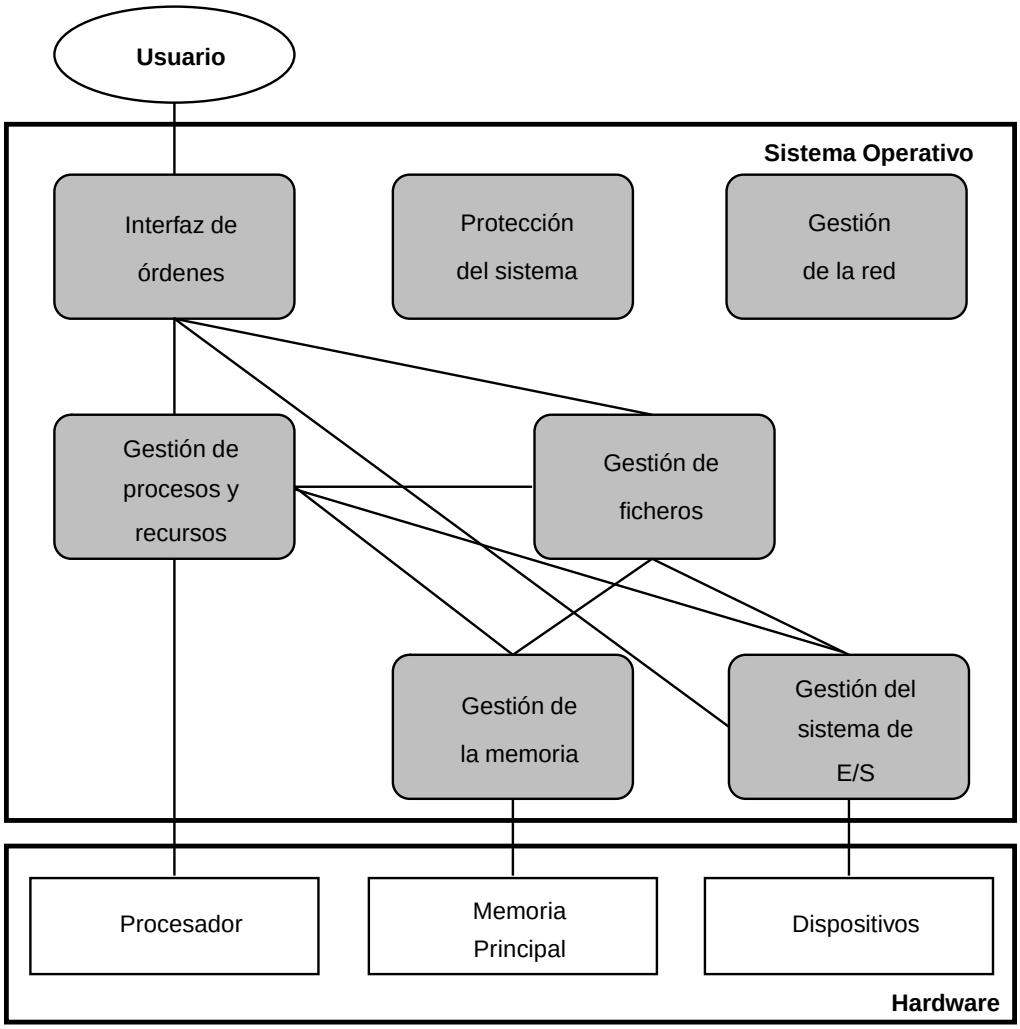

Figura 2.1: Funciones del sistema operativo

Interfaz de órdenes Esta función no tiene por qué formar parte del sistema operativo. Por ejemplo, los sistemas de tiempo real duros suelen carecer de ella. Sin embargo, en sistemas interactivos es una de las más importantes, debido a que es la interfaz entre el usuario y el sistema. En algunos sistemas operativos, como UNIX, esta función no forma parte del núcleo y la realiza un software especial que se ejecuta en el momento en que el usuario entra al sistema. Éstos son los distintos *shell*

como el bash, Bourne, C, etc, permitiendo al usuario escoger una u otra interfaz. En otros sistemas, como MS-DOS, esta función está integrada dentro del propio sistema operativo en el denominado **intérprete de órdenes**, sin posibilidad de cambiarla.

La figura 2.1 muestra las relaciones entre las distintas funciones y las correspondientes abstracciones de las que se encargan. Podemos ver como el hardware queda oculto por el sistema operativo, siendo la interfaz de órdenes la relación entre el sistema y el usuario. Por claridad, se han omitido las relaciones de la función de protección del sistema, debido a que afecta a todos los módulos. Igualmente la gestión de la red sólo tiene sentido en sistemas distribuidos, donde se encarga de la comunicación con otros equipos.

Algunas de las funciones que realiza el sistema operativo pueden ser solicitadas por los programadores, recibiendo el nombre de **servicios**. Éstos nos permiten trabajar con el hardware y ciertas estructuras software del sistema de una forma transparente, facilitando de este modo la labor de programación.

2.2. Solicitud de servicios

Los servicios proporcionados por el sistema operativo pueden solicitarse mediante dos técnicas, las **llamadas al sistema** y el **paso de mensajes**. La utilización de una u otra dependerá de la estructura que presente el sistema como se verá en el apartado 2.3. Ambos mecanismos proporcionan una interfaz entre los procesos y el sistema operativo.

2.2.1. Llamadas al sistema

La mayoría de los sistemas operativos proporcionan unas instrucciones especiales que se denominan llamadas al sistema. Algunos sistemas sólo permiten hacer estas peticiones desde el lenguaje ensamblador, mientras que otros, como UNIX, permiten realizarlas desde un lenguaje de alto nivel como C. En el programa 2.1 tenemos un ejemplo de cómo se realizan llamadas al sistema en lenguaje C en un sistema UNIX. En él se utilizan las llamadas read y write. La llamada read lee de la entrada estándar y va almacenando los datos leídos en el vector buf. Posteriormente, la llamada write los escribe en la salida estándar. Desde el punto de vista del usuario, no hay ninguna diferencia entre hacer una llamada al sistema o a una función de biblioteca.

Programa 2.1 Llamada al sistema

```
#include <unistd.h>
#define TAM 255

main()
{
  char buf[TAM];
  int count;

  while ((count = read(STDIN_FILENO, buf, TAM)) > 0)
    write(STDOUT_FILENO, buf, count);
}
```

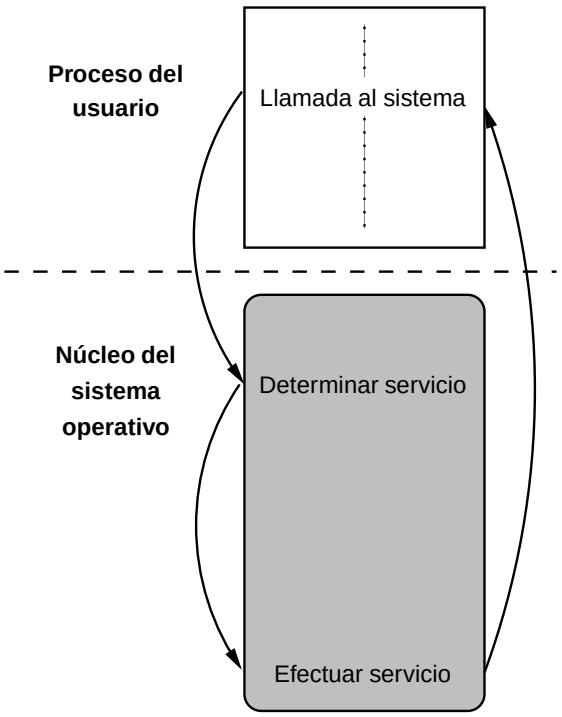

Figura 2.2: Funcionamiento de una llamada al sistema

En la figura 2.2 podemos ver su funcionamiento. Cuando un proceso hace una llamada al sistema provoca una interrupción al sistema operativo, se

cambia de modo usuario a modo supervisor, el sistema determina el servicio solicitado, y llama a la función que se encarga de realizarlo. Cuando finaliza, cambia a modo usuario y devuelve el control al proceso de usuario que realizó la llamada.

2.2.2. Paso de mensajes

Otro mecanismo para solicitar los servicios del sistema operativo es el paso de mensajes. El procedimiento se puede ver en la figura 2.3. En este caso, el proceso de usuario construye un mensaje que describe el servicio que solicita, éste se envía al sistema operativo mediante la función send. Éste se encarga de comprobarlo, cambiar a modo supervisor, entregarlo a un proceso del sistema que realiza el servicio solicitado, y cambiar a modo usuario.

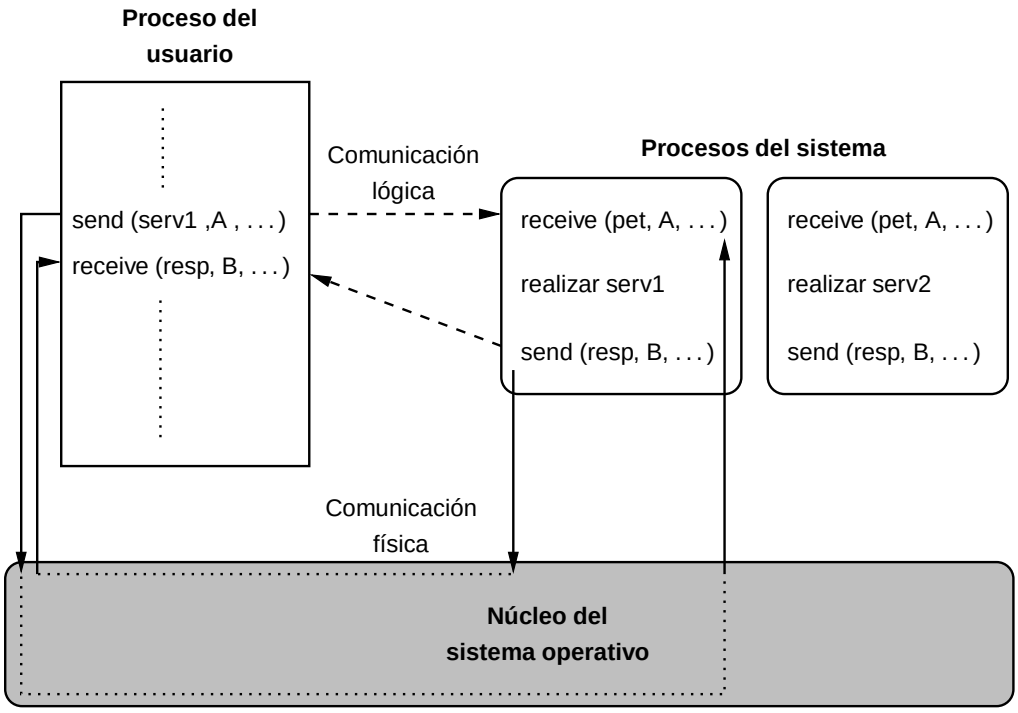

Figura 2.3: Funcionamiento del paso de mensajes

Mientras, el proceso de usuario espera el resultado del servicio solicitado mediante la función receive. Cuando el proceso del sistema termina de realizar

el servicio, devuelve a través del sistema operativo, mediante la función send, la respuesta al proceso de usuario que lo solicitó.

Podemos ver que se establece una comunicación lógica entre el proceso del usuario y el proceso del sistema que realiza el servicio. Sin embargo, el mecanismo de comunicación entre ambos va a través del núcleo que se encarga de realizar el intercambio de mensajes.

2.3. Estructura de un sistema operativo

Un sistema operativo es un software bastante complejo y grande, cuyo objetivo principal es el de abstraer el hardware haciendo más fácil su utilización. Para ello, el sistema operativo se suele implementar como una colección de módulos que realizan las funciones vistas en el apartado 2.1.

No existe un mecanismo que permita diseñar e implementar de forma óptima un sistema operativo. La labor del diseñador del sistema consiste precisamente en determinar cómo se van a diseñar los módulos que implementan las funciones del sistema, con objeto de satisfacer los requisitos mínimos de corrección, rendimiento, mantenibilidad, flexibilidad y transportabilidad.

El núcleo es la parte del sistema operativo que se ejecuta en modo supervisor, y se puede decir que es el verdadero sistema operativo. Existen dos formas básicas de estructuración de un núcleo. Una consiste en que incluya todas las funciones y servicios del sistema operativo; es lo que se denomina **núcleo complejo**. La segunda consiste en que el núcleo sea lo más pequeño posible, y la mayor parte de los servicios sean proporcionados por los denominados procesos del sistema. Hablamos del **núcleo mínimo**.

Como puede verse en la clasificación de la figura 2.4, en ambos casos existen distintas posibilidades a la hora de estructurar los componentes que lo forman. Así, para los sistemas que presentan un núcleo complejo podemos distinguir a los que no tienen una estructura interna, como es el caso de la estructura monolítica, y los que sí la tienen, en este segundo caso la estructura puede ser modular o por capas.

No existen reglas que determinen cuando es mejor utilizar un núcleo complejo o mínimo. Sin embargo, es posible encontrar ciertas diferencias entre los dos modelos. Así, cuando se diseña un sistema con un núcleo complejo, podemos verlo como un conjunto de funciones, de tal modo, que los procesos de usuario se ejecutan en modo usuario, y cuando solicitan un servicio lo ha-

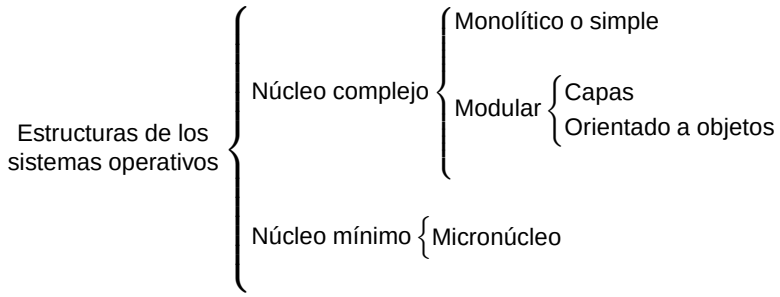

Figura 2.4: Clasificación de los sistemas operativos según su estructura

cen a través de las llamadas al sistema, pasando en ese momento a ejecutar código del sistema operativo en modo núcleo.

Por otro lado, un sistema que presenta un conjunto mínimo de servicios en su núcleo, requiere que el resto de los servicios sean proporcionados mediante procesos del sistema. En este caso, los procesos de usuario solicitan los servicios a los procesos del sistema mediante el paso de mensajes, siendo el núcleo el encargado del intercambio de mensajes.

Las diferencias entre los dos modelos tienen implicaciones en el rendimiento del sistema. Los sistemas operativos basados en llamadas al sistema son más eficientes que los basados en paso de mensajes, ya que éstos requieren crear un mensaje, copiarlo al espacio de direcciones del proceso del sistema y pasar a ejecutar este proceso. La respuesta implica hacer los mismos pasos a la inversa. Mientras que en el caso de una llamada al sistema, el proceso de usuario al hacer la llamada provoca un cambio de modo pasando a ejecutar código del núcleo; al finalizar la ejecución de éste se produce otro cambio de modo.

Por otro lado, los sistemas basados en paso de mensajes son más flexibles porque permiten que el propio usuario pueda añadir y modificar servicios del sistema, ya que esto no implica modificar el código del núcleo. Además es más fácil diseñarlos para que puedan tomar el control de la máquina en situaciones especiales, tales como la ejecución temporal de tareas. Por este motivo, en los sistemas operativos basados en funciones suele existir un conjunto de procesos del sistema que se encargan de manejar estas situaciones especiales; podemos citar como ejemplo los sistemas UNIX que poseen los llamados **demonios** (*daemons*).

2.3.1. Estructura simple o monolítica

Los sistemas que presentan esta estructura se componen de un único módulo lógico que realiza todas las funciones. Aunque se aplican algunas técnicas de programación modular, el problema es distinguir los distintos componentes del sistema (administrador de procesos, de la memoria, etc.), ya que todos están relacionados. De esta forma, si queremos aislar el administrador de procesos es difícil debido a que suele estar relacionado con el administrador de memoria; éste a su vez necesita conocer el estado de las operaciones de E/S, etc. La característica principal de estos sistemas operativos es que los servicios se solicitan mediante llamadas al sistema.

Este tipo de sistema operativo está escrito como un conjunto de procedimientos. Cada uno tiene una interfaz bien definida en términos de parámetros y resultados, y puede llamar a cualquier otro que le proporcione algo que necesite. No existe ninguna ocultación de información ya que todos los procedimientos son visibles a los demás. Para construir el programa objeto del sistema operativo, en primer lugar se compilan los procedimientos individuales, enlazándolos posteriormente en un solo fichero objeto.

Estos sistemas operativos tienen la ventaja de estar muy adaptados al hardware, por lo que consiguen un alto rendimiento. Sin embargo, sacrifican otros aspectos del diseño como la facilidad de mantenimiento, portabilidad, etc.

Hoy en día, numerosos sistemas comerciales presentan esta estructura, debido a que empezaron siendo sistemas pequeños, simples y limitados. Sin embargo, gracias a su éxito fueron creciendo y superando los objetivos inicialmente establecidos. Ejemplos de estos sistemas son UNIX y MS-DOS.

2.3.1.1. El sistema operativo UNIX

Uno de los ejemplos más representativos de la estructura monolítica es el sistema operativo UNIX. En ellos es posible distinguir dos partes bien diferenciadas: el núcleo y el resto de programas del sistema. En la figura 2.5 podemos ver su estructura.

El núcleo es la parte más importante del sistema UNIX. Se sitúa entre el hardware y la interfaz de llamadas al sistema. En él se encuentran las funciones de gestión de memoria, de E/S, planificación de la CPU, así como un conjunto de manejadores de dispositivos, que con el paso del tiempo ha ido aumentando. Por tanto, en un solo nivel existe una gran cantidad de

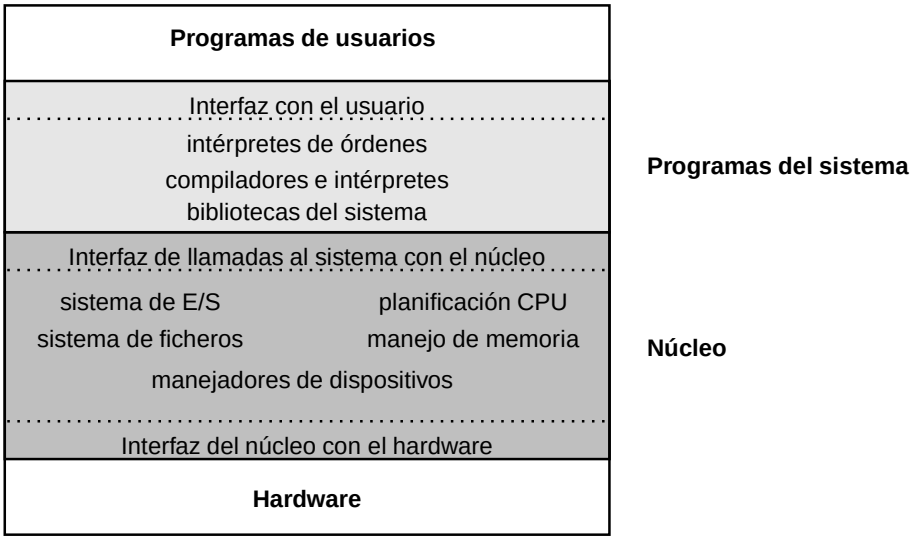

Figura 2.5: Estructura del sistema UNIX

funciones que imposibilita organizarlas. Como consecuencia de la evolución sufrida por UNIX, el núcleo ha pasado de ocupar unos 40 KiB a tener cerca de 1 MiB, aunque en algunas máquinas ocupa varios MiB.

El sistema UNIX a pesar de poseer una estructura monolítica, suele tener una serie de procesos del sistema. Éstos son los demonios, que se encargan de tomar el control de la máquina en situaciones especiales y permiten tener más seguridad en el sistema.

2.3.1.2. El sistema operativo MS-DOS

Otro claro ejemplo de este tipo de estructura es MS-DOS. Este sistema fue diseñado e implementado originalmente por unas pocas personas que no suponían que llegaría a ser tan popular. Fue escrito para proporcionar la máxima funcionalidad en el menor espacio, por lo que no fue dividido en módulos cuidadosamente. La figura 2.6 muestra su estructura.

Aunque MS-DOS tiene cierta estructura, sus interfaces y módulos funcionales no están bien separados. Por ejemplo, los programas de aplicación son capaces de acceder a las rutinas básicas de E/S para escribir directamente en

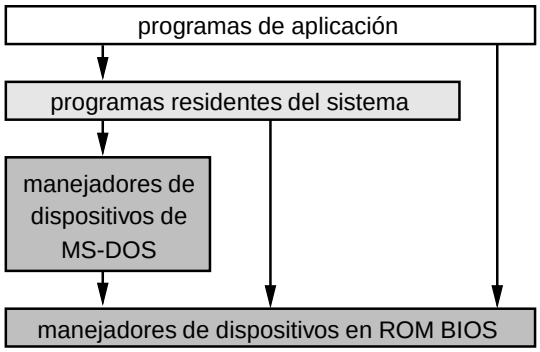

Figura 2.6: Estructura de MS-DOS

la pantalla y en las unidades de disco. Tal libertad hace al sistema MS-DOS vulnerable a los programas erróneos, causando la caída completa del sistema o el borrado del disco cuando un programa de usuario falla. Por supuesto, MS-DOS está también limitado por el hardware para el que fue diseñado, dado que el microprocesador Intel 8088 no proporciona modo dual, por lo que los diseñadores de MS-DOS tuvieron que dejar el hardware base accesible.

2.3.2. Estructura modular

Los sistemas con una estructura monolítica presentan ciertos inconvenientes, como la dificultad de mantenerlos y actualizarlos. Sin embargo, un objetivo de diseño de cualquier software debe ser la facilidad de mantenimiento. Por este motivo, se debe buscar una estructura interna que nos permita cumplir con ese objetivo.

Para facilitar tanto el diseño como la implementación de los sistemas operativos, se puede utilizar una estructura modular. Consiste en dividirlo en unidades lógicas definiendo bien las interfaces entre ellas. Posteriormente, esos módulos pueden ser implementados bien como procesos o bien como funciones del sistema operativo.

Esta estructura aparece como consecuencia de los métodos establecidos por la Ingeniería del software. Ésta establece ciertas técnicas para construir el software, basadas en la abstracción de datos y la encapsulación de funciones, permitiendo ocultar las implementaciones, de tal modo que puedan adaptarse a nuevos cambios sin tener que modificar las interfaces.

Un sistema que presente estructura modular es más fácil de mantener y actualizar. Sin embargo, la modularización conlleva un coste en pérdida de rendimiento del sistema respecto a la estructura monolítica. Por este motivo, es necesario llegar siempre a un compromiso entre modularización y rendimiento.

La modularización de un sistema se puede llevar a cabo de muchas formas. Una es empleando la metodología orientada a objetos, que ha permitido el desarrollo de este tipo de sistemas operativos. Éstos se caracterizan porque todos sus elementos se representan como objetos, permitiendo tener una serie de clases representativas, tales como dispositivos hardware, procesos, ficheros, etc. Un ejemplo de un sistema operativo orientado a objetos es Choices[1]. Surge de una investigación experimental, donde se emplea tanto un diseño como un lenguaje orientado a objetos.

Otro enfoque para realizar la modularización es la estructura en capas, que a continuación vamos a considerar.

2.3.2.1. Estructura en estratos o por capas

Una forma de estructurar el sistema operativo es mediante la **estratificación**, que consiste en romper el sistema en estratos, niveles o capas, cada uno de ellos construido sobre los anteriores. Una capa de un sistema operativo es una implementación de un objeto abstracto que permite la encapsulación de datos y las operaciones que pueden manipular esos datos. En un sistema operativo, la capa más baja (capa 0) es el hardware, y la más alta (capa N) es la interfaz con el usuario.

En la figura 2.7 aparece una capa típica de un sistema operativo, que consta de algunas estructuras de datos y un conjunto de funciones que pueden ser llamadas por capas de nivel superior. Esta capa, a su vez, puede utilizar operaciones de las de nivel inferior, sin importarle cómo están implementadas.

La principal ventaja de los sistemas por capas es la encapsulación. Las capas se seleccionan de tal forma que cada una sólo utiliza las funciones (operaciones) y servicios de las de nivel inferior. Esto hace mucho más fácil la depuración y verificación del sistema. El primer nivel puede ser depurado sin tener en cuenta el resto del sistema, puesto que, por definición, sólo usa el hardware para implementar sus funciones. Una vez que el primer nivel ha sido depurado, se pasa a depurar el siguiente independientemente de los

[1] Sistema operativo construido en la Universidad de Illinois

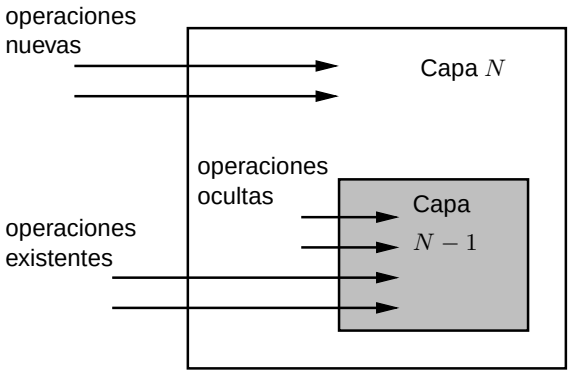

Figura 2.7: Capa de un sistema operativo

demás, y así sucesivamente. Si se encuentra un error durante la depuración de un nivel particular, sabemos que el error debe estar en él, puesto que los inferiores han sido ya depurados. Por tanto, el diseño y la implementación de un sistema se simplifican cuando se rompe en niveles.

La mayor dificultad que presenta este método es la definición de los niveles. Es necesario hacer una planificación cuidadosa, ya que cada capa sólo puede usar aquellas funciones que están por debajo de ella. Esta limitación ha hecho que en los últimos años se haya abandonado un poco este tipo de estructura. En la actualidad se tiende a utilizar menos capas con más funcionalidad, proporcionando la mayoría de las ventajas del código modular y evitándose los problemas de definición de capas y su interacción.

El primer sistema operativo que se diseñó siguiendo este método fue el sistema THE en la *Technische Hogeschool Eindhoven*, que estaba formado por seis capas, como se muestra en la figura 2.8.

Capa 5: Programas de usuario
Capa 4: *Buffering* para dispositivos de E/S
Capa 3: Manejador del dispositivo de consola
Capa 2: Manejo de la memoria
Capa 1: Planificación de la CPU
Capa 0: Hardware

Figura 2.8: Estructura en capas del sistema THE

Otro ejemplo de sistema en capas es OS/2, descendiente directo de MS-DOS, que fue creado para vencer las limitaciones de éste. El sistema OS/2

añade la multitarea, así como otras características nuevas. Debido a esta complejidad añadida y al hardware más potente para el que fue diseñado, OS/2 fue implementado de una forma más estratificada que MS-DOS. En la figura 2.9 puede verse claramente que no está permitido el acceso directo de los usuarios al hardware, ya que éste proporciona el modo de operación dual, dándole al sistema operativo mayor control sobre el hardware.

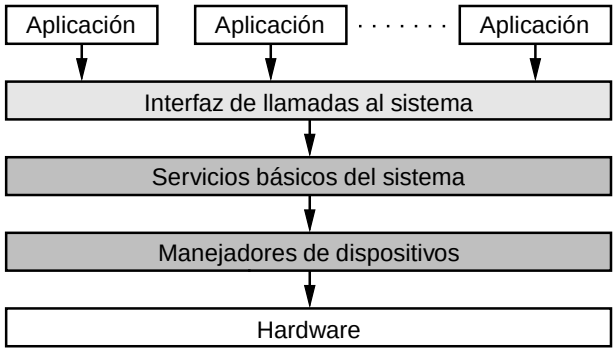

Figura 2.9: Estructura en capas de OS/2

2.3.3. Micronúcleo

Una tendencia en los sistemas operativos modernos es llevar más allá la idea de trasladar gran parte del código del sistema operativo a niveles superiores, dejando un núcleo mínimo. El método usual consiste en implementar la mayor parte de las funciones del sistema operativo en procesos del sistema. Para pedir un servicio, como por ejemplo la lectura de un bloque de un fichero, un proceso de usuario envía una petición al proceso que realiza el trabajo y éste le envía la respuesta. De esta forma, la petición de los servicios al sistema no se realiza mediante una llamada al sistema, sino mediante el paso de mensajes.

En este modelo, que se muestra en la figura 2.10, todo lo que el núcleo hace es manejar la comunicación entre los procesos de usuario y del sistema. Al dividir el sistema operativo en partes, cada una de las cuales se encarga del manejo de una faceta de éste, como el servicio de ficheros, procesos, terminales, etc., cada parte del sistema se hace más pequeña y fácil de manejar.

La imagen que aparece en la figura 2.10 de un núcleo que maneja sólo el transporte de mensajes entre procesos de usuario y procesos del sistema no

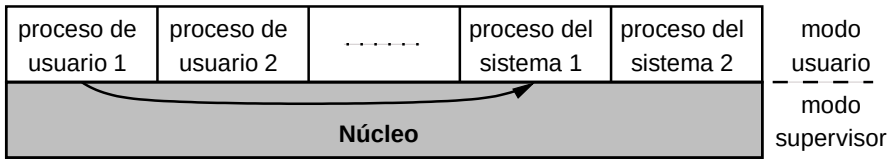

Figura 2.10: Estructura de micronúcleo

es completamente real. Algunas funciones del sistema operativo (tales como la carga de órdenes en los registros de los dispositivos de E/S) son difíciles, si no imposible, de hacer desde programas en el espacio del usuario. Una forma de tratar este problema es tener algunos procesos críticos del sistema (por ejemplo, los manejadores de los dispositivos de E/S) que se ejecuten en modo supervisor, con acceso completo a todo el hardware, pero que se comunican con otros procesos usando el mecanismo normal de mensajes.

Este tipo de estructura también se suele conocer como **cliente-servidor**; a los procesos de usuario que piden un servicio se les denomina clientes, mientras que a los procesos que realizan el servicio servidores.

2.3.4. Máquinas virtuales

El concepto de máquina virtual surge con el sistema VM/370 de IBM en 1972. La idea principal de la máquina virtual es la de permitir ejecutar varios sistemas operativos simultáneamente sobre el mismo hardware. Para ello, separa las dos funciones básicas que realiza un sistema de tiempo compartido: multiprogramación y abstracción del hardware.

Un sistema operativo se sitúa entre el hardware y el resto del software del sistema, de forma que posee el control total de los recursos, actuando como intermediario para las distintas peticiones de los procesos que se están ejecutando. Para facilitar la labor, el sistema forma una capa de abstracción del hardware, de forma que proporciona un conjunto de llamadas al sistema que ocultan los detalles reales.

El software del sistema ubicado encima del sistema operativo, así como los programas de usuario usan tanto las llamadas al sistema como instrucciones hardware sin llegar a diferenciarlas, dando la ilusión de tener contacto directo con éste. Esta idea es utilizada por los sistemas de máquina virtual. En éstos el corazón del sistema, conocido como **monitor de máquina virtual**, se

ejecuta sobre el hardware y proporciona varias máquinas virtuales al siguiente nivel de software, como se muestra en la figura 2.11. Para ello crea una interfaz de llamadas al sistema que simulan el hardware subyacente. Estas máquinas virtuales no son máquinas extendidas, con ficheros y otras características agradables, sino que son copias exactas del hardware desnudo, incluyendo los modos núcleo/usuario, dispositivos de E/S, interrupciones, y todo aquello que tiene la máquina real.

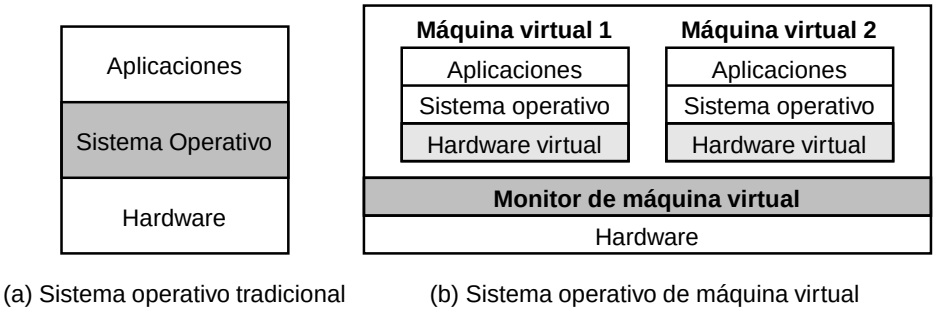

Figura 2.11: Sistema operativo tradicional frente a máquina virtual

Debido a que cada máquina virtual es idéntica al hardware verdadero, cada una puede estar ejecutando cualquier sistema operativo que se pudiera ejecutar directamente sobre éste. Así, diferentes máquinas virtuales pueden ejecutar diferentes sistemas operativos.

El concepto de máquina virtual vuelve a estar de moda en los sistemas operativos por distintos motivos. Así, para poder mantener la compatibilidad con antiguos programas de MS-DOS en nuevos equipos, se crea una máquina Intel virtual donde poder ejecutarlos, de forma que sus instrucciones son posteriormente traducidas al procesador nativo. Esto lo emplean fabricantes como Sun.

Recientemente ha surgido VMWare que permite llevar esta tecnología a los ordenadores personales, de forma que un usuario pueda estar ejecutando sistemas como Windows NT y LINUX de forma simultánea.

La idea de máquina virtual ha sido llevada también a los lenguajes de programación. Así, Java se implementa con un compilador que genera un código de salida que se ejecuta en la Máquina Virtual Java (JVM[2]). De este modo, los programas tienen una mayor transportabilidad, ya que para

[2] *Java Virtual Machine*

poder ejecutarlo sólo necesitan disponer en el sistema informático de la JVM, independientemente del hardware original sobre el que se ejecute.

2.4. El sistema operativo LINUX

LINUX es una versión de UNIX para procesadores Intel 386 y 486, que se distribuye de forma gratuita. Fue desarrollado principalmente por Linus Torvalds de la Universidad de Helsinki, y posteriormente se ha ido desarrollando con la ayuda de muchos programadores a través de Internet.

Existen muchas distribuciones de LINUX, cada una con sus peculiaridades y sus propios mecanismos de instalación. El único elemento común a todas las distribuciones es el núcleo del sistema operativo, que se desarrolla de forma coordinada y con actualizaciones sistemáticas. LINUX, al igual que su predecesor, presenta un núcleo monolítico. No obstante, presenta cierta estructura modular.

2.4.1. Componentes del núcleo

En la figura 2.12 se muestra un diagrama de bloques del núcleo, en el que aparecen todos los módulos que lo componen y las relaciones que se establecen entre ellos.

Esta figura muestra tres niveles: usuario, núcleo y hardware. El núcleo proporciona un conjunto de servicios al resto de aplicaciones o procesos de usuario que pueden ser solicitados a través de las llamadas al sistema. Éstas junto con las funciones de biblioteca constituyen la interfaz entre los programas de usuario y el núcleo. En LINUX se definen un total de 163 llamadas al sistema.

Las partes más importantes del núcleo son el administrador de la memoria y el de procesos. El primero es el encargado de asignar la memoria y áreas de intercambio a los procesos. De este modo, es el responsable de ubicar un proceso en memoria principal o en memoria secundaria, según las necesidades del sistema.

Por otro lado, el administrador de procesos es el encargado de la creación de éstos, permitiendo la multiprogramación, mediante la conmutación del procesador entre los procesos activos.

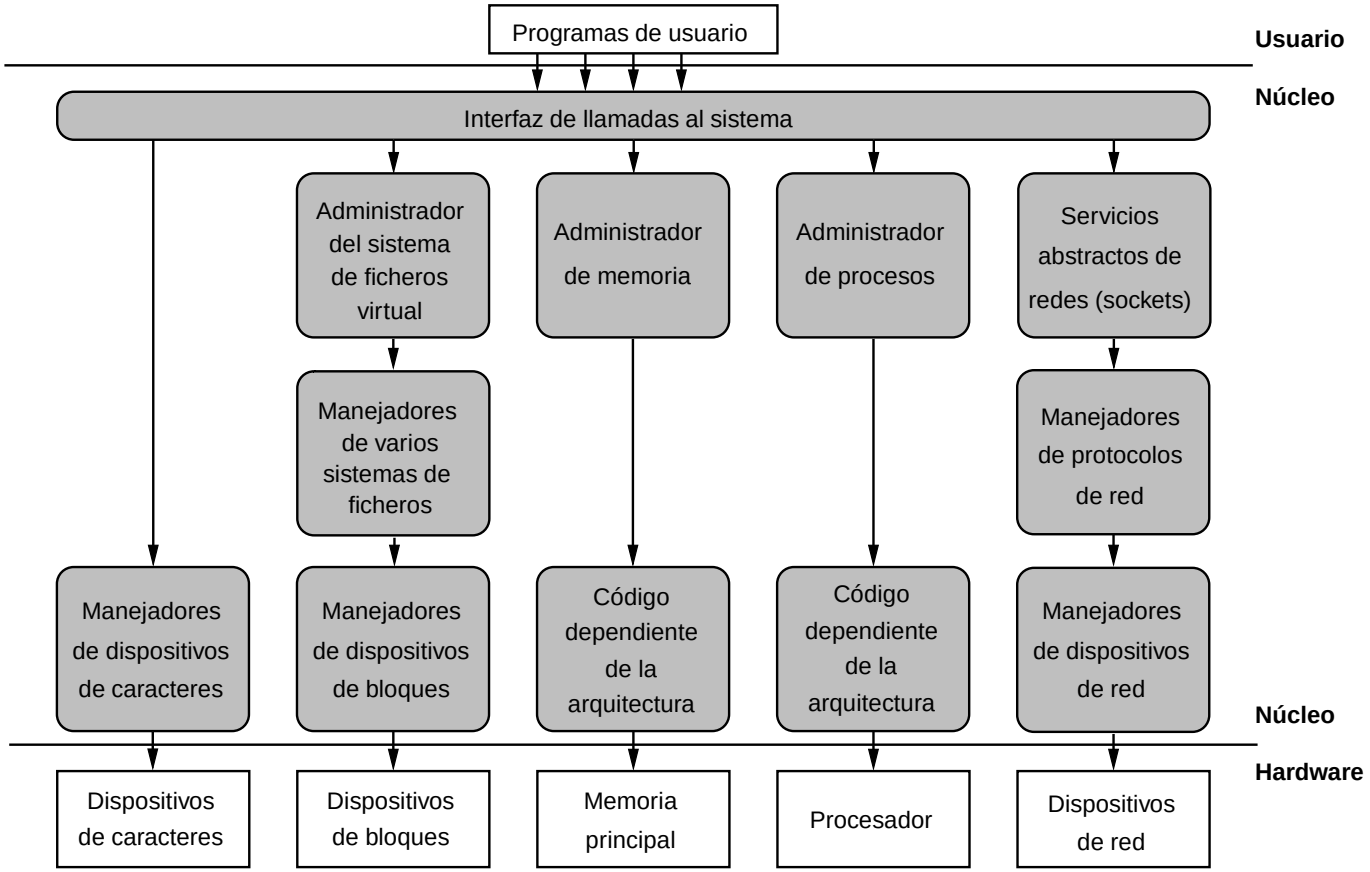

Figura 2.12: Diagrama del núcleo de un sistema LINUX

LINUX proporciona varios sistemas de ficheros, por lo que el núcleo contiene un manejador específico para cada uno de ellos. La similitud entre las operaciones que éstos realizan permiten agruparlas en el denominado **sistema de ficheros virtual** (VFS[3]). Cuando un proceso intenta realizar una operación sobre un fichero, la petición va a través del VFS que la desvía al manejador específico del sistema de ficheros.

En el nivel más bajo, el núcleo contiene un manejador específico para cada tipo de dispositivo soportado. Sin embargo, las similitudes entre ellos hace posible establecer clases generales de manejadores. Esto permite que los miembros de una clase posean la misma interfaz con el núcleo, diferenciándose en la manera de implementarla.

Otros servicios proporcionados por el núcleo tienen propiedades similares. Así, los distintos protocolos de redes han sido abstraídos en una única interfaz programable: la biblioteca *BSD socket*.

LINUX utiliza una serie de procesos del sistema, denominados demonios. Éstos se encargan de tomar el control de la máquina en situaciones especiales. Entre ellos se encuentran el demonio de impresión, de paginación, etc.

El sistema operativo LINUX permite configurar el núcleo adaptándolo a las características del sistema. Se puede generar un **núcleo dinámico**, en el que ciertas partes del sistema se cargarán en memoria cuando se necesiten y, en caso contrario, se descargarán. Éstos son los llamados **módulos** que permiten tener un núcleo más flexible y de menor tamaño.

2.5. Resumen

Un sistema operativo realiza numerosas funciones que definen un ambiente de ejecución de los programas. Éstas se encargarán de gestionar todos los recursos del sistema, y realizar una abstracción del hardware, permaneciendo oculto a los usuarios.

A la hora de diseñar un sistema operativo, es muy importante determinar su estructura interna, ya que ésta influye en la forma de organizar las distintas funciones del sistema, así como el mecanismo de solicitar los servicios.

Éstos pueden solicitarse mediante dos mecanismos: paso de mensajes y llamadas al sistema. Los primeros son más lentos pero más flexibles a la hora

[3] *Virtual File System*

de diseñar nuevos servicios. Las llamadas al sistema son funciones propias del núcleo del sistema operativo, por lo que introducir un nuevo servicio requiere disponer del código fuente y un gran conocimiento de cómo se encuentra diseñado.