Sistemas Operativos. Estructuras

Tabla de Contenidos

Autor: Lina García Cabrera

1 Introducción

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Sistemas Operativos: Estructuras
Un sistemas operativo proporciona el entorno en el que se ejecutan los programas.

Internamente, los sistemas operativos se pueden estructurar de diferentes modos. Los diseñadores de sistemas operativos deciden cómo organizar los distintos componentes y cómo se interconectan.

En este módulo se describen:

- las diferentes modos de estructurar un sistema operativo,
- se clasifican los sistemas operativos atendiendo a diversos criterios, por último, se explica
- cómo se crean los sistemas operativos y
- cómo consigue una computadora iniciar su sistema operativo.

OBJETIVOS

- Exponer las diversas formas de estructurar un sistema operativo.
- Clasificar a los sistemas operativos aplicando diversos criterios.
- Explicar cómo se instalan, personalizan y arrancan los sistemas operativos.

1 Estructuras de los Sistemas Operativos

1.1 Sistemas monolíticos/Estructura simple

2 Estructuras de los Sistemas Operativos

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Sistemas Operativos: Estructuras
Se ha visto el aspecto externo de los sistemas operativos (es decir, la interfaz con el programador y con el usuario), en este apartado se echará un vistazo al interior del sistema operativo. Se examinarán algunas de las formas posibles de estructurar el código de un sistema operativo, cómo los componentes más comunes de los sistemas operativos se interconectan y se funden en un *kernel*.

2.1 Sistemas monolíticos/Estructura simple

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Sistemas Operativos: Estructuras

Este tipo de organización ha sido la más común. El sistema operativo se escribe como **una colección de procedimientos**, cada uno de los cuales puede llamar a los demás cada vez que así lo requiera. Cuando se usa esta técnica, **cada procedimiento del sistema tiene una interfaz bien definida en términos de parámetros y resultados**, y cada uno de ellos es libre de llamar a cualquier otro, si éste último proporciona un cálculo útil para el primero.

Para construir el programa objeto real del sistema operativo siguiendo este punto de vista, se compilan de forma individual los procedimientos, o los ficheros que contienen los procedimientos, y después se enlazan en un sólo fichero objeto con el enlazador. En términos de ocultación de la información, ésta es prácticamente nula: **cada procedimiento es visible a los demás** (en contraste con una estructura con módulos o paquetes, en la que la mayoría de la información es local a un módulo, y donde sólo los datos señalados de forma expresa pueden ser llamados desde el exterior del módulo).

Los servicios (llamadas al sistema) que proporciona el sistema operativo se solicitan colocando los parámetros en lugares bien definidos, como los registros o la pila, para después ejecutar una instrucción especial de trampa, a veces referida como **llamada al núcleo** o **llamada al supervisor**.

1. Esta instrucción **cambia la máquina del modo usuario al modo núcleo** (también conocido como modo supervisor), y transfiere el control al sistema operativo, lo que se muestra en el evento (1) de la Figura 1.

2. El sistema operativo examina entonces los parámetros de la llamada para determinar cual de ellas se desea realizar, como se muestra en (2) de la Figura 1.
3. A continuación, el sistema operativo analiza una tabla que contiene en la entrada k un puntero al procedimiento que implementa la *k-ésima* llamada al sistema. Esta operación, que se muestra en (3) de la Figura 1, identifica el procedimiento de servicio, al cual se llama.
4. Por último, la llamada al sistema termina y el control vuelve al programa del usuario.

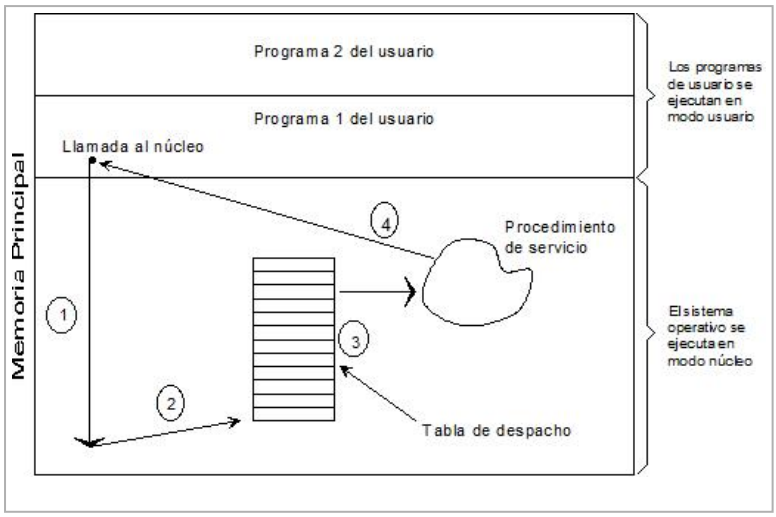


Figura 1. La forma en que debe hacerse una llamada al sistema.
Esta organización sugiere una estructura básica del sistema operativo:

1. Un programa principal que llama al procedimiento del servicio solicitado.
2. Un conjunto de procedimientos de servicio que lleva a cabo las llamadas al sistema.
3. Un conjunto de procedimientos de utilidades que ayudan a los procedimientos de servicio.

En este modelo, para cada llamada al sistema existe un procedimiento de servicio que se encarga de ella. Los procedimientos de utilidad hacen cosas necesarias para varios procedimientos de servicio, como por ejemplo, buscar los datos del programa del usuario. Esta división de los procedimientos en tres capas se muestra en la Figura 2.

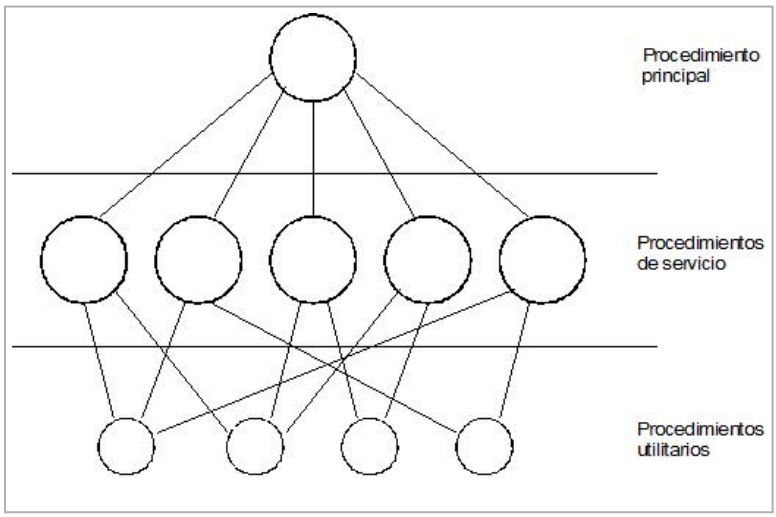
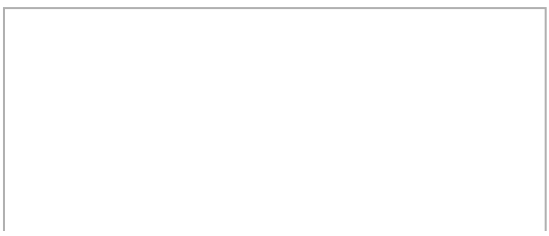


Figura 2. Un modelo de estructura simple para un sistema monolítico.

> En MS-DOS, las interfaces y niveles de funcionalidad no están separados. Los programas de aplicación pueden acceder a las rutinas de E/S para escribir directamente en la pantalla o en las unidades de disco. Esto hace que MS-DOS sea vulnerable a programas erróneos o maliciosos, el sistema operativo falla cuando lo hace el programa de usuario. Como el 8088 de Intel para el que fue escrito no proporciona modo dual ni protección hardware, los diseñadores de MS-DOS no tuvieron más opción que dejar accesible el hardware base.



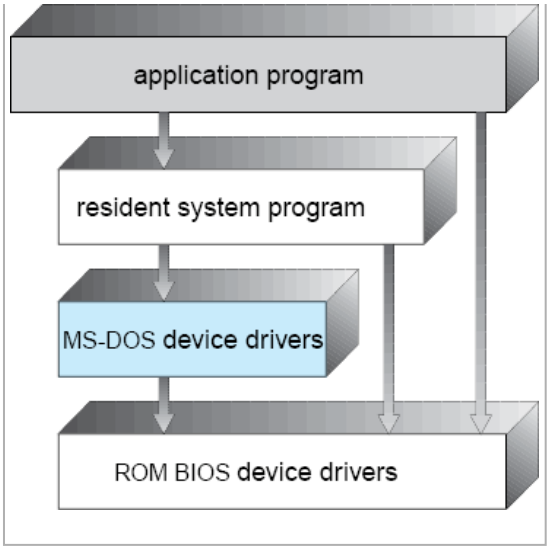


Figura 3. Estructura de niveles de MS-DOS.

> El UNIX original también tiene una estructura simple. Consta de dos partes separadas: el *kernel* y los programas de sistema. El *kernel* se divide en una serie de interfaces y controladores de dispositivos, que se han ido añadiendo y ampliando conforme UNIX ha evolucionado. El UNIX original tiene una estructura simple de niveles. Todo lo que está por debajo de la interfaz de las llamadas al sistema y por encima del hardware físico es el kernel. El kernel proporciona el sistema de ficheros, los mecanismos de planificación de la CPU, la funcionalidad de la gestión de la memoria y otras mediante llamadas al sistema.

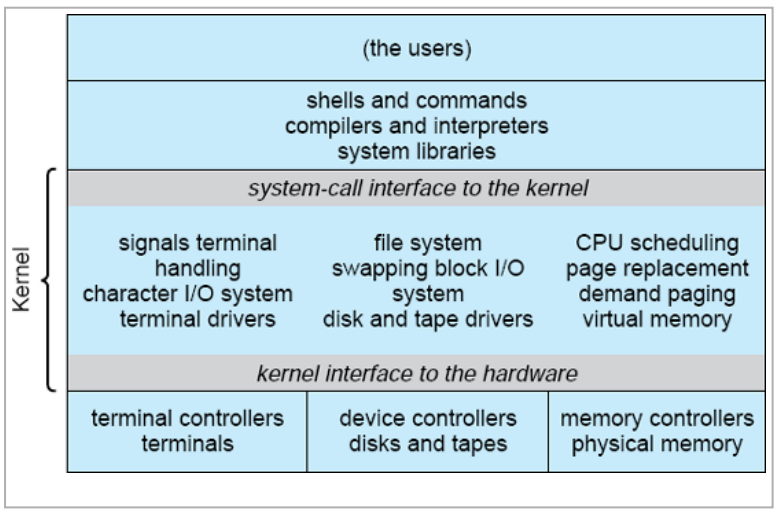


Figura 4. Estructura del sistema UNIX.

2.2 Estructura en niveles

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Sistemas Operativos: Estructuras

Con el soporte *hardware* adecuado, los sistemas operativos pueden dividirse en componentes y tener mayor control de la computadora y de las aplicaciones.

Un sistema se puede hacer modular estructurándolo en una serie de niveles o capas. El **nivel inferior es el hardware** (nivel 0); el **superior es la interfaz de usuario** (nivel N). Una capa de software de nivel i es un conjunto de programas que trabajan directamente con la interfaz de su nivel inferior (i - 1), y presentan a su nivel superior (i + 1) una interfaz que permite utilizar el nivel i -1 de una forma más sencilla. Cada nivel se implementa utilizando sólo las operaciones proporcionadas por los niveles inferiores.

Esta estructuración presenta la dificultad de **cómo definir apropiadamente los diferentes niveles**, no siempre es obvio dónde situar cada funcionalidad del sistema operativo. Otro problema de esta implementación es la eficiencia. **Cada nivel añade una carga de trabajo** a la llamada al sistema que tarda más en ejecutarse en un sistema con niveles.

2.3 Modelo cliente-servidor/Microkernel

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Sistemas Operativos: Estructuras

Una tendencia de los sistemas operativos modernos es la de trasladar el código a capas superiores, y eliminar la mayor parte posible del sistema operativo para mantener un **núcleo mínimo o *microkernel***. El punto de vista usual es el implantar la mayoría de las **funciones del sistema operativo como procesos de usuario**. Para solicitar un servicio, como la lectura de un bloque de cierto fichero, un proceso de usuario (denominado en este caso proceso cliente) envía la solicitud a un proceso servidor, que realiza el trabajo y devuelve la respuesta.

En este modelo, que se muestra en la Figura 5, lo único que hace el **micronúcleo es controlar la comunicación entre los clientes y los servidores mediante paso de mensajes**. Al separar el sistema operativo en partes, cada una de ellas controla una faceta del sistema, como el servicio a ficheros, servicio a procesos, servicio a terminales o servicio a la memoria; **cada parte es pequeña y controlable**. Además, puesto que todos los servidores se ejecutan como procesos en modo usuario, y no en modo núcleo, no tienen acceso directo al hardware. En consecuencia, si hay un error en el servidor de ficheros éste puede fallar, pero esto no afectará en general a toda la máquina.

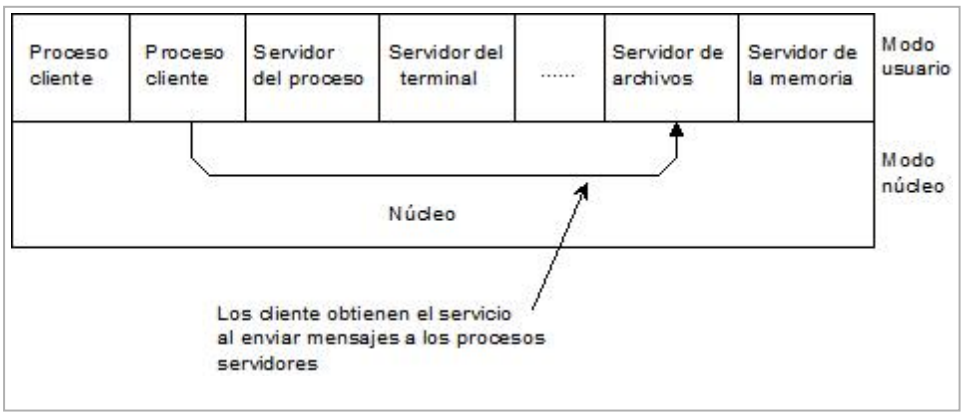


Figura 5. El modelo cliente-servidor.

Otra de las ventajas del modelo cliente-servidor es su capacidad de **adaptación para su uso en sistemas distribuidos** (véase la Figura 6). Si un cliente se comunica con un servidor mediante mensajes, el cliente no necesita saber si el mensaje se gestiona de forma local, en su máquina, o si se envía por medio de una red a un servidor en una máquina remota. En lo que respecta al cliente, lo mismo ocurre en ambos casos: **se envió una solicitud y se recibió una respuesta**.

La idea anterior de un núcleo que sólo controla el transporte de mensajes de clientes a servidores, y viceversa, no es totalmente real. **Algunas funciones del sistema operativo** (como la introducción de órdenes en los registros físicos de los controladores de E/S) son difíciles, si no es que **imposible de realizar, a partir de programas de usuario**.

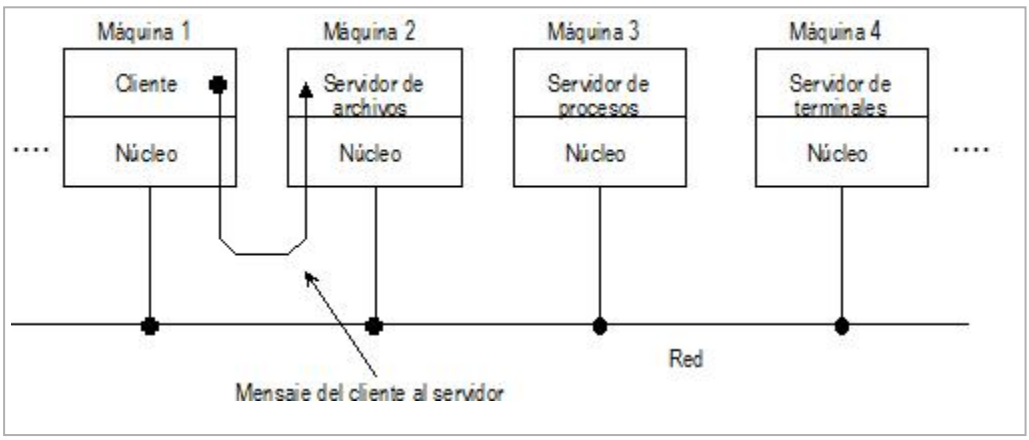


Figura 6. El modelo cliente-servidor en un sistema distribuido.
Existen dos formas de afrontar este problema.

1. Una es hacer que algunos **procesos de servidores críticos** (por ejemplo, los gestores de los dispositivos de E/S) se ejecuten en realidad en modo núcleo, con acceso total al hardware, pero

de forma que se comuniquen con los demás procesos mediante el mecanismo normal de mensajes.
2. La otra forma es construir una **cantidad mínima de mecanismos dentro del núcleo**, pero manteniendo las decisiones de política relativos a los usuarios dentro del espacio de los usuarios.

> Por ejemplo, el núcleo podría reconocer que cierto mensaje enviado a una dirección especial indica que se tome el contenido de ese mensaje y se cargue en los registros del controlador de algún disco, para iniciar la lectura del disco. En este ejemplo, el núcleo ni siquiera inspeccionaría los bytes del mensaje para ver si son válidos o tienen algún sentido; sólo los copiaría ciegamente en los registros del controlador del disco. Es evidente que debe utilizarse cierto esquema para limitar tales mensajes sólo a los procesos autorizados.

La separación entre mecanismos y política es un concepto importante, aparece una y otra vez en diversos contextos de los sistema operativos.

Mecanismos y Políticas
Los **mecanismos determinan cómo hacer algo**; las **políticas determinan qué hacer**. En el diseño de un sistema operativo es importante implementar mecanismos genéricos insensibles a los cambios de política. Un cambio de política requerirá solo la redefinición de algunos parámetros del sistema.

> Por ejemplo, si tenemos un mecanismo para dar prioridad a los procesos independiente podemos utilizarse para beneficiar a los programas que hacen uso intensivo de E/S o, por el contrario, dar prioridad a los hacen uso intensivo de CPU.

2.4 Módulos

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Sistemas Operativos: Estructuras

La mejor metodología para diseñar sistemas operativos es la que usa las técnicas de programación orientada a objetos para crear un **kernel modular**. El ***kernel dispone de un conjunto de componentes fundamentales y enlaza dinámicamente los servicios adicionales, durante el arranque o en tiempo de ejecución***. Esta estrategia es la que usan las implementaciones modernas de UNIX, como Solaris, Linux y Mac OS X.

El resultado es similar al de un sistema de niveles, cada sección del *kernel* tiene interfaces bien definidas y protegidas, pero es más flexible porque cualquier módulo puede llamar a cualquier otro módulo. Además, también es similar a la utilización de un microkernel, ya que el módulo principal sólo dispone de las funciones esenciales y de los conocimientos de cómo cargar y comunicarse con otros módulos. Sin embargo, es más eficiente que un microkernel porque los módulos no necesitan invocar un mecanismo de paso de mensajes.

2.5 Máquina Virtual

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Sistemas Operativos: Estructuras

Una estructura de **máquina virtual** suministra un conjunto de máquinas abstractas cada una de la cuales actúa casi como una **copia idéntica del hardware subyacente**. Esta estructura funciona simulando separadamente (multiplexando) cada máquina abstracta sobre la máquina base real. Así, para cada usuario del sistema, este aparece como si el tuviera su propia copia del hardware subyacente, de forma que la protección y seguridad se obtiene de una forma directa.

Un problema de este enfoque es que **el rendimiento del sistema operativo tiende a degradarse**, dado que la simulación es costosa. La principal ventaja de esta estructura es que **permite implementar varios sistemas operativos sobre cada máquina virtual**. Sin embargo, los sistemas operativos de cada máquina virtual son disjuntos, esto prohíbe la riqueza de interacciones, compartimiento, y comunicación que se necesita en los sistemas actuales.

Actualmente las máquinas virtuales se están poniendo de nuevo de moda para solucionar problemas de compatibilidad entre sistemas. Vmware, VirtualBox y la máquina virtual Java operan por encima de un sistema operativo, se ejecutan como una aplicación más sobre un sistema operativo host.

2.6 Estructuras orientadas a objetos.

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Sistemas Operativos: Estructuras
En estructuras de este tipo los servicios del sistema operativo se implementan como una colección de objetos que se definen como segmentos protegidos por capacidades. Cada objeto tiene un tipo que designa sus propiedades: procesos, directorios, archivos, etc. Un objeto tiene un conjunto de operaciones mediante las cuales se puede acceder y modificar su segmento interno. Antes de que un usuario solicite un servicio de un objeto, este debe adquirir sus capacidades, incluidos los derechos de las operaciones permitidas. El kernel tiene la responsabilidad de proteger las capacidades frente a accesos erróneos o malintencionados. Ejemplos de estos sistemas son HYDRA, StarOS, Medusa, iMAX, Eden, Amoeba, y Clouds.

3 Clasificación de Sistemas Operativos

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Sistemas Operativos: Estructuras
Se van a clasificar los sistema operativos atendiendo a diferentes criterios. Un sistema operativo presentará ciertas características que dependerán de la máquina virtual que se quiera implementar.

3.1 Respecto al modo de trabajo del usuario

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Sistemas Operativos: Estructuras
Se pueden clasificar los sistema operativos, partiendo de este punto de vista,

- en **on line** (o **interactivos**) y
- **off line** (o **batch** o por **lotes**).

Como ejemplo de los primeros ya hemos comentado **los sistemas de tiempo compartido, los sistemas interactivos** son útiles, entre otros, en entornos de desarrollo de programas, de procesamiento de textos y de ejecución de programas interactivos.

Un ejemplo de los segundos son los **sistemas por lotes**. Los sistemas ***batch*** se caracterizan porque una vez introducida una tarea en el ordenador, el usuario no mantiene contacto alguno con ella hasta que finaliza su ejecución.

3.2 Respecto al número de usuarios

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Sistemas Operativos: Estructuras
Si se tiene en cuenta el número de usuarios se puede hablar de sistemas

- **monousuario**: se puede acceder al ordenador mediante un único terminal
- **multiusuario** (o **multiacceso**): varios terminales de acceso simultáneo. Un sistema mutiusuario necesariamente debe ser de tiempo compartido.

Ejemplos de sistemas operativos monousuario son MS DOS y CP/M. Ejemplos de sistemas multiusuario son UNIX, AS/400, MVS.

3.3 Respecto al propósito

2º Grado en Ingeniería en Informática

teoría de Sistemas Operativos
Bloque Sistemas Operativos: Estructuras
Atendiendo al uso que quieran dar los usuarios al ordenador, los sistema operativos se pueden dividir en

- **sistemas de propósito general** que se caracterizan por tener un gran número de usuarios trabajando sobre un amplio abanico de aplicaciones. Se suelen dividir en dos grandes grupos:
  - los de lotes y
  - los de tiempo compartido.
- y **sistemas de propósito específico** que suelen ser embebidos, disponen de una interfaz muy limitada o ni siquiera la incluyen, prefieren invertir su tiempo en monitorizar y gestionar dispositivos *hardware* (motores de automóvil, brazos robóticos, controlar la calefacción, la luz, los sistemas de alarma).

Sin embargo, existen sistemas que compaginan el tiempo compartido con procesos por lotes, incluso posibilidades de tiempo real.

> Los sistemas embebidos casi siempre ejecutan **sistemas operativos de tiempo real**, estos sistemas se usan en entornos donde se deben aceptar y procesar en tiempo breve un gran número de sucesos, en su mayoría externos al ordenador. Una serie de sensores proporcionan estos datos a la computadora, los analiza y, posiblemente, ajusta los controles para modificar los datos de entrada de los sensores. Ejemplos de tales aplicaciones incluyen control industrial, equipamiento telefónico conmutado, control de vuelo y simulaciones en tiempo real.

> La mayoría de los sistemas operativos están diseñados para gestionar datos convencionales (texto, programas, documentos). Sin embargo, ahora existen una tendencia a incorporar datos multimedia (audio, vídeo) que requieren una restricciones de tiempo.
> Aplicaciones mutimedia tales como archivos audio, películas DVD, videoconferencias, secuencias de vídeo, webcasts requieren que los sistemas operativos sean capaces de darles soporte.

Los sistemas de mano (PDA, Pocket-PC, Ipod, teléfonos móviles) usan sistemas operativos embebidos de propósito especial. Debido a su tamaño, tienen poca memoria, procesadores más lentos y pantallas pequeñas.

3.4 Al existir varios procesadores

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Sistemas Operativos: Estructuras

Los ordenadores con más de una CPU se clasifican en

- **multiprocesadores**. En un ***multiprocesador los procesadores comparten memoria y reloj (son síncronos)***.
- **sistemas distribuidos (o multicomputadores)**. En un ***sistema distribuido tenemos varios procesadores con su propia memoria, además, no están sincronizados***.

Los sistemas operativos que controlan un multiprocesador son distintos a los empleados en los sistemas distribuidos.

Actualmente toman fuerza nuevas tendencias como el Multiprocesamiento Simétrico, donde todas las CPU's que forman el sistema comparten una única copia del código y los datos del kernel, frente a Multiprocesamiento Asimétrico, donde cada CPU tiene su propio kernel y datos locales.

En una red de ordenadores o multicomputadora se tiene una configuración de varios ordenadores conectados físicamente. Los ordenadores de una red pueden tener **sistemas operativos de red o sistemas operativos distribuidos**.

En un **sistema operativo de red** los usuarios son conscientes de la existencia de varios ordenadores, y pueden conectarse con máquinas remotas para, por ejemplo, copiar ficheros. Cada máquina ejecuta su propio sistema operativo local y tiene su propio usuario (o grupo de usuarios). Los sistemas operativos de red no difieren de los sistemas operativos tradicionales de un sólo procesador. Necesitan un controlador de red, algunas rutinas de E/S para utilizar dicho controlador, y programas que permitan la conexión y el acceso a ordenadores remotos, pero esas características adicionales no modifican la estructura esencial del sistema operativo.

En un **sistema distribuido** los ficheros que utiliza un usuario (así como el procesador en el que se ejecutan sus programas) pueden estar situados en cualquier ordenador de la red. Además, esto es

transparente al usuario. Los sistemas operativos distribuidos requieren algo más que añadir un poco de código a un sistema operativo de un único procesador, ya que los sistemas distribuidos difieren en aspectos críticos de los sistemas centralizados.

4 Cómo se crea un Sistema Operativo

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Sistemas Operativos: Estructuras

Normalmente los sistemas operativos se diseñan para ejecutarse en cualquier tipo de máquina y en diversas instalaciones, con una amplia variedad de periféricos. El sistema debe configurarse o generarse para cada computadora en concreto, a esto se le denomina **generación del sistema** (*system generation*, SYSGEN).

El sistema operativo se distribuye en CD-ROM o DVD-ROM o como imagen ISO, un fichero en formato CD-ROM o DVD-ROM. Para generar el sistema, **el programa SYSGEN** lee un fichero determinado o pregunta al operador del sistema **la información de configuración específica del *hardware* o prueba el *hardware* para averiguar qué componentes están instalados**.

Se debe determinar:

- qué CPU se va a usar,
- con qué cantidad de memoria se cuenta,
- qué dispositivos están instalados,
- qué opciones del sistema operativo se desean (número de búferes, tamaño, algoritmo de planificación, el máximo número de procesos, etc.).

Conseguida esta información, el administrador de sistemas puede usarla para **modificar una copia de código fuente del sistema operativo y compilar el sistema operativo completo**.

Otra forma menos personalizada, es utilizar la descripción del sistema para **crear una serie de tablas y seleccionar una biblioteca precompilada**. Estos módulos se montan para formar el sistema operativo final.

También se puede construir **un sistema controlado por completo por tablas**. Todo el código forma parte del sistema y la selección se produce en tiempo de ejecución.

5 Cómo iniciar un Sistema Operativo

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Sistemas Operativos: Estructuras
Una vez generado el sistema operativo hay que usarlo. **¿Cómo sabe el hardware donde está el kernel y cómo cargarlo?**

Al proceso de iniciar una computadora mediante la carga del kernel se llama Arranque del Sistema. Una pequeña parte del código (**programa de arranque**, *bootstrap program* o **cargador de arranque**, *bootstrap loader*), se encarga de localizar el kernel, lo carga en memoria principal e inicia su ejecución. Los PC usan un proceso en dos pasos, un sencillo cargador de arranque extrae del disco un programa de arranque más complejo, que carga el kernel.

Cuando se enciende una computadora, la CPU recibe un evento de **reinicializaición** (*reset event*). Entonces el registro instrucción se carga con una dirección de memoria predefinida (en esta posición está el programa de arranque) y comienza la ejecución. El programa de arranque está situado en una memoria de sólo lectura (ROM o EEPROM *erable programmable read-only memory*).

El **programa de arranque** realiza varias tareas:

1. ejecuta una serie de diagnósticos para determinar el estado de la máquina y
2. luego inicializa registros de la CPU, de los controladores de los dispositivos y la memoria principal y
3. finaliza con el inicio del sistema operativo.

Los dispositivos de mano suelen cargar todo el sistema operativo en memoria ROM dado que suele tener menor tamaño y controla un *hardware* sencillo.

Los sistemas operativos de propósito general (Windows, Mac OS X y UNIX) el cargador se almacena en firmware y el sistema operativo en disco. El programa de arranque que ejecuta los diagnósticos tiene un pequeño fragmento de código que puede leer un sólo bloque situado en un posición fija (el bloque 0) del disco, lo carga y ejecuta el código que hay en dicho **bloque de arranque** (*boot block*). Este programa puede cargar el sistema operativo completo en memoria e iniciar su ejecución.

El programa de arranque del disco, y el propio sistema operativo, se puede cambiar escribiendo nuevas versiones en el disco. Un disco que tiene una **partición de arranque** (se llama un **disco de arranque** o **disco del sistema**).

> GRUB es un ejemplo de un programa de arranque de código abierto para sistemas Linux.