TEMA 1

Una introducción a los sistemas operativos

1 Introducción

No existe una definición única de sistema operativo. Los sistemas operativos existen porque son una vía razonable para resolver los problemas que crea un sistema informático. El hardware por sí solo no es fácil de utilizar. Es necesario ayudar tanto al programador como al usuario a abstraerse de la complejidad del hardware. La forma de hacerlo es colocando una capa de software, por encima del hardware con el fin de presentar al usuario del sistema y a las aplicaciones una **interfaz de máquina virtual** que facilite la comprensión y utilización del sistema. Esta capa de software es lo que se denomina sistema operativo. El sistema operativo integra un conjunto de funciones responsables de controlar el hardware que son comunes a la mayoría de las aplicaciones, como las funciones de control de los dispositivos y las rutinas de servicio a interrupciones, ocultando al programador los detalles del hardware y ofreciéndole una interfaz cómoda para utilizar el sistema.

Desde otro punto de vista, el sistema operativo debe asegurar el funcionamiento correcto y eficiente del sistema. Un sistema informático actual consta de un elevado número de componentes que es necesario gestionar. A lo largo de la historia de los ordenadores, se viene produciendo una importante evolución que afecta a los diferentes elementos que componen el sistema. Esta evolución se produce tanto en el aspecto tecnológico (desde las válvulas y los relés hasta los los circuitos VLSI) como a nivel de la arquitectura (diferentes técnicas arquitecturales para aumentar la velocidad del procesador, jerarquías de memorias, ...) y en el campo de los lenguajes de programación (bibliotecas, lenguajes, interfaces...). Esta evolución viene forzada por requerimientos de eficiencia y facilidad de uso de los computadores. Sin embargo, hay que tener en cuenta que el aumento de eficiencia de cada uno de los componentes del sistema no asegura un aumento en la eficiencia global del sistema. La **gestión sintonizada de todos los recursos** será la gran responsable del éxito o fracaso. Desde esta perspectiva, el sistema operativo es el responsable de proporcionar una asignación ordenada y controlada de los diferentes recursos (procesador, memoria, dispositivos de E/S...) a cada uno de los programas que compiten por ellos.

Metáfora del conductor y el mecánico

En todo sistema es importante la distinción entre interfaz e implementación. El usuario de un sistema debe conocer su interfaz, pero cómo está implementado es un asunto del diseñador o el personal técnico de mantenimiento. El usuario de un automóvil sólo tiene que conocer la interfaz para que el vehículo le sea de utilidad. Así, debe aprender a manejar el volante, los intermitentes, las luces, el acelerador y el freno. Para facilitarle las cosas, los fabricantes tienden a estandarizar la interfaz: el acelerador es un pedal siempre situado en el mismo sitio; el sentido "derecha" siempre se representa en los mandos como un giro en sentido de las agujas del reloj... Como el sistema no es perfecto, el usuario debe realizar algunas tareas de "gestión": si el coche no es automático, debe elegir la marcha adecuada; cuando el depósito se vacía, debe repostar un tipo u otro de combustible... Sin embargo, estas tareas tienden a estar cada vez más limitadas. Hace un siglo, el usuario del coche solía contar con un chófer-mecánico, pues los automóviles eran muy poco fiables, y debían ponerse en marcha accionando manualmente el motor de arranque desde el exterior. Hoy en día, uno puede ser un buen conductor sin tener conocimientos de mecánica, y muchos conductores ignoran, por ejemplo, que el coche tiene un motor eléctrico para el arranque. Los mecánicos son las personas encargadas del mantenimiento, conocen perfectamente la estructura interna del automóvil y no tienen porqué ser buenos conductores: podrían incluso no saber conducir.

Podemos concretar diciendo que el concepto de sistema operativo está ligado a dos ideas diferentes. Para un usuario/programador, un sistema operativo es el conjunto de funciones que le permiten usar los recursos de la máquina obviando las características del hardware. Esta es la *visión funcional* del sistema operativo, que permite ver al sistema como una *máquina virtual*. Es esta la visión en la que

profundizaremos este curso. Para un diseñador de sistemas, en cambio, un sistema operativo es el software que, instalado sobre la máquina desnuda, permite controlar eficientemente sus recursos. Este punto de vista corresponde a la *implementación* del sistema operativo.

Ambos puntos de vista hacen referencia en gran parte a los mismos conceptos y términos, pero sus enfoques —y sus objetivos— son diferentes. En este curso de introducción a los sistemas operativos estudiaremos las funcionalidades que ofrecen los sistemas operativos en general, así como los fundamentos de cómo el sistema operativo las soporta. Las técnicas y modelos fundamentales del diseño de los sistemas operativos, así como los conceptos y las tareas propias de la administración de sistemas y redes, incluyendo la gestión de la seguridad, se estudian en cursos de la especialidad de Ingeniería de los Computadores.

2 Visión funcional de los sistemas operativos

De los dos enfoques expuestos arriba, este es el menos claramente definido y desarrollado en la literatura. Quizás ello se deba al hecho de que históricamente ha sido el programador de una interfaz quien diseña la funcionalidad de la interfaz y no se siente especialmente inclinado a discutir cuáles son los servicios concretos que debe ofrecer la interfaz, que luego se añaden o modifican en revisiones posteriores de acuerdo a las necesidades. En el sistema operativo, además, esta interfaz no es única, en el sentido de que, además del conjunto de **llamadas al sistema** (primitivas del sistema operativo) que ofrece a las aplicaciones, se puede considerar —de hecho históricamente así ha sido— al **intérprete de comandos** como parte del sistema operativo, e incluso, por evolución, a la interfaz gráfica de usuario. Desde el momento en que estas interfaces presentan los recursos de la máquina al usuario no programador y al desarrollador de aplicaciones, podrían considerarse, en efecto, como parte de la visión funcional del sistema operativo, aunque dado el grado de complejidad que han alcanzado los sistemas operativos, resulta más práctico tratarlas como una disciplina específica.

Interfaces e interfaces

Quizás el vicio profesional más frecuente de los ingenieros informáticos es el de no diferenciar adecuadamente entre las diferentes interfaces del sistema. Basta con echar una mirada a cualquier aplicación o sistema operativo para darse cuenta de ello. Por ejemplo, uno puede encontrar en "Accesorios", junto a una calculadora o un reproductor y un grabador de sonido, herramientas como el "desfragmentador del disco". Sorprendentemente, en "panel de control" uno encuentra una aplicación para leer textos en voz alta. Es como si un fabricante de automóviles hubiera ubicado una llave inglesa en el salpicadero, junto al hueco para las gafas de sol, y el aparato de radio bajo el capó, junto al motor. No es de extrañar que muchos usuarios domésticos aborrezcan los ordenadores. "La informática es muy complicada", dicen. Bueno, la mecánica también lo es, y, sin embargo, ningún conductor se siente inseguro mientras conduce por el hecho de no saber manejar una llave inglesa. En este sentido, podría decirse que los sistemas operativos actuales son como los automóviles de hace un siglo.

En lo que sigue, consideraremos la *interfaz de llamadas al sistema* como la fundamental del sistema operativo, que define al sistema como *máquina virtual* a este nivel. El conjunto de llamadas al sistema de un sistema operativo describe la interfaz entre las aplicaciones y el sistema y determina la compatibilidad entre máquinas a nivel de código fuente.

El usuario final ve el sistema informático en términos de aplicaciones. Las aplicaciones pueden construirse con un lenguaje de programación y son desarrolladas por los programadores de aplicaciones. Si tuviéramos que desarrollar las aplicaciones responsabilizándonos en cada momento del control del hardware que utilizan, la programación de aplicaciones sería una tarea abrumadora y seguramente no podríamos disfrutar de aplicaciones tan sofisticadas como las de hoy en día. Además, las aplicaciones se sirven también de un conjunto de **utilidades** o servicios que facilitan aún más la labor del programador, como editores, compiladores, depuradores (*debuggers*), etc. Aquí podemos incluir también las **bibliotecas** de funciones que están a disposición de las aplicaciones (funciones matemáticas,

gráficas, etc). Normalmente, estos servicios no forman parte del sistema operativo. La Figura 1 ofrece un resumen de este enfoque.

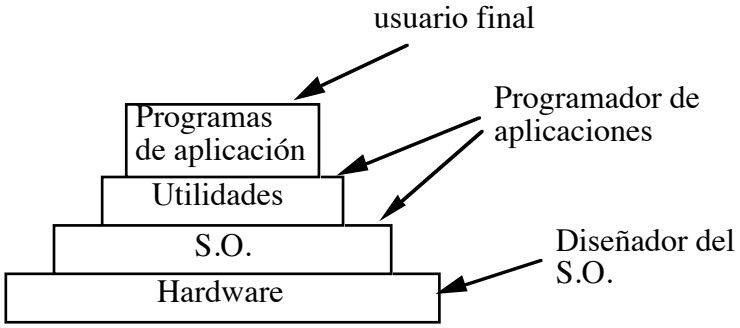

***Figura 1**. Estructura por capas de un sistema informático.*

3 Funciones de un sistema operativo

En general, e independientemente del tipo de interfaz, los sistemas operativos ofrecen habitalmente un conjunto de funciones que podemos resumir en las siguientes:

- **Ejecución de programas**. Para ejecutar un programa se requiere cierto número de tareas. Las instrucciones y los datos se deben cargar en memoria principal, los ficheros y dispositivos de E/S se deben inicializar y preparar otros recursos. El sistema operativo realiza todas estas tareas.

- **Control de los dispositivos de E/S**. Cada dispositivo requiere un conjunto propio y peculiar de instrucciones y señales de control para su funcionamiento. El sistema operativo se encarga de todos estos detalles de manera que el programador pueda ver los accesos a los dispositivos como simples lecturas y escrituras.

- **Acceso a los ficheros**. Históricamente se ha utilizado el concepto de fichero o archivo (del inglés *file*) como la representación permanente de un conjunto de información con un nombre global en el sistema. Los ficheros residen en memoria no volátil, como discos y memorias *flash*. Además de la naturaleza del dispositivo, el sistema operativo ha de gestionar el formato del fichero y la forma de almacenarlo.

- **Control del acceso al sistema**. En el caso de sistemas multiusuario, el sistema operativo dispone de los mecanismos adecuados para controlar el acceso a los recursos del sistema en función de los derechos definidos para cada usuario.

- **Detección y respuesta a errores**. Cuando un sistema informático está en funcionamiento pueden producirse errores. Estos errores pueden ser del hardware (errores de acceso a memoria o de los dispositivos), o del software (*overflow* aritmético, intento de acceder a una posición prohibida de memoria...). En muchos de estos casos el sistema operativo cuenta con elementos hardware para detectar estos errores y comunicarlos al sistema operativo , que debe dar una respuesta que elimine la condición de error con el menor impacto posible sobre las aplicaciones que están en ejecución. La respuesta puede ser desde terminar el programa que produjo el error, hasta reintentar la operación o simplemente informar del error a la aplicación.

- **Contabilidad**. Es habitual que un sistema operativo ofrezca herramientas para tracear operaciones y accesos y recoger datos de utilización de los diferentes recursos. Esta información puede ser útil para anticiparse a la necesidad de mejoras futuras y para ajustar el sistema de manera que mejore su rendimiento. También puede utilizarse a efectos de facturación. Finalmente, ante un problema de seguridad, esta información puede servir para descubrir al atacante.

4 Interfaces del sistema operativo

En un sistema estructurado en capas, una capa L_k ofrece una **interfaz** a la capa superior, la capa L_{k+1}, representada por un conjunto de funciones que determinan la forma en que desde la capa L_{k+1} se accede a la capa L_k. La implementación de la capa L_k es independiente de la interfaz y se dice que es *transparente* a la capa L_{k+1}, en el sentido de que cuando se diseña la capa L_{k+1} no hay que preocuparse de cómo la capa L_k está implementada. Una interfaz debe especificar con precisión las funciones que ofrece y cómo se usan (argumentos, valores de retorno, etc).

En general, un sistema operativo ofrece tres interfaces diferentes:

Interfaz de usuario. Cuando no existían los terminales gráficos de hoy en día, el usuario tenía que comunicarse con el sistema tecleando órdenes que le permitían ejecutar programas, consultar directorios, etc. El sistema operativo le ofrecía para ello una utilidad específica, el intérprete de comandos (*shell* en la terminología de Unix), que le presentaba como interfaz un conjunto de comandos cuya forma de utilización estaba (o debía estar) bien especificada en un manual (por ejemplo el *man* de Unix, en su Sección 1). Hoy en día, las interfaces gráficas de usuario facilitan enormemente la forma de interacción del usuario mediante objetos y conceptos intuitivos (iconos, apuntadores, clicks del ratón, arrastrar y soltar...). Si en el caso de los intérpretes de comandos cada sistema ofrecía el suyo propio (el usuario tenía que aprender a usurlo, habitualmente asistiendo a un curso), las interfaces gráficas de usuario son lo suficientemente comunes e intuitivas como para que su utilización esté al alcance de todo el mundo.

Interfaz de administración. El administrador de un sistema informático es la persona encargada de instalar el sistema, mantenerlo y gestionar su uso. En un sistema compuesto de varios computadores, esta labor incluye gestionar cuentas de usuario y recursos en red, con especial atención en el cuidado de la privacidad de los usuarios y la seguridad de la información. El administrador del sistema es un profesional que conoce las herramientas y funciones específicas que el sistema le ofrece para ello y que sólo él puede usar, pues requieren privilegios especiales. En general se basa para ello en una extensión del intérprete de comandos (por ejemplo, en Unix, especificada en la Sección 8 el *man*), aunque el uso de estas herramientas no excluye la utilización de la interfaz gráfica. En cambio, un sistema personal no debería exigir, idealmente, esfuerzo alguno de administración por parte del usuario, ya que a este no se le suponen conocimientos específicos para ello, al igual que al conductor de un automóvil no se le requieren conocimientos de mecánica. La realidad es que, al igual que el automovilista debe saber cómo cambiar una rueda, el usuario de un computador tiene que solucionar hoy en día algunos problemas de administración derivados de la inmadurez e imperfección de los sistemas operativos.

Interfaz de programación. Para desarrollar aplicaciones sobre un sistema operativo, el programador utiliza, sea cual sea el lenguaje de programación que use, un conjunto de funciones para acceder a los servicios del sistema operativo, la interfaz de llamadas al sistema. Estas funciones no difieren, en apariencia, de otras funciones de biblioteca que ofrece el lenguaje. Sin embargo, las llamadas a un sistema operativo son específicas de ese sistema y por lo tanto probablemente incompatibles con las de otro sistema operativo, ya que se refieren a objetos y conceptos específicos de ese sistema. En realidad, lo habitual es que el programador no utilice

directamente las llamadas al sistema operativo, sino que se sirva de funciones de biblioteca propias del lenguaje. Por ejemplo, si utiliza C como lenguaje de programación, el programador utilizará la función *printf* para salida de datos, independientemente del sistema operativo que esté utilizando. Sin embargo, *printf* es una función implementada en función de las llamadas al sistema operativo (en concreto para Unix, la llamada al sistema *read*), por lo que el código generado es específico para dicho sistema. Esto, en general, no lo tiene en cuenta el programador de aplicaciones, pero sí el desarrollador de la biblioteca, programador de sistemas, que es el usuario de la interfaz de llamadas al sistema operativo y se basará en la especificación correspondiente (en Unix, la Sección 2 del *man*).

APIs

*Hoy en dia los programadores suelen hablar de API (*Aplication Programming Interface*) para referirse al conjunto de funciones disponibles en una plataforma para el desarrollo de aplicaciones. Una API puede ser el conjunto de llamadas al sistema ampliado con otras funciones de biblioteca, aunque las propias llamadas al sistema suelen estar ocultadas por funciones de biblioteca que facilitan la programación. Puede haber también APIs específicas adaptadas a aplicaciones concretas. En última instancia, una API depende del lenguaje de programación y del sistema operativo para el que esa API está implementada.*

*En el mundo Java, ya que se trata de un lenguaje interpretado, las APIs son independientes del sistma operativo: es la máquina virtual (*JVM*), la que interpreta las funciones de biblioteca para el sistema operativo subyacente.*

5 Evolución de los sistemas operativos

Desde la perspectiva que nos ofrece la ya relativamente larga historia de los sistemas operativos, y teniendo en cuenta sus campos de aplicación, actualmente se puede hablar de diferentes modelos de cómputo, que determinan la funcionalidad de un sistema operativo, y a veces su estructura:

Sistemas por lotes o *batch*. Son los primeros sistemas operativos (década de 1950) propiamente dichos, que permitían procesar en diferido paquetes de tarjetas perforadas basándose en el uso de un elemento software conocido como **monitor**. Los usuarios no interaccionaban directamente con el computador, sino que entregaban los trabajos en un fajo de tarjetas perforadas (un lote) al operador del computador, quien ordenaba secuencialmente los lotes y los colocaba en un dispositivo de entrada (lector de tarjetas). Cada lote tenía insertadas tarjetas de control con órdenes para el monitor. La última tarjeta era una orden de retorno al monitor que le permitía comenzar a cargar automáticamente el siguiente programa.

Multiprogramación. Con el secuenciamiento automático, el procesador estaba a menudo sin trabajo debido la lentitud de los dispositivos de E/S en comparación con la velocidad del procesador, que, cuando encuentra una instrucción de E/S debe esperar a que el dispositivo concluya la operación. Hay que tener en cuenta que el precio de una CPU de esta época era desorbitadamente alto, por lo que se pretendía que trabajara el 100% del tiempo, objetivo imposible de conseguir con los sistemas por lotes. Esto llevó a los ingenieros de la época a idear estrategias de procesamiento más eficientes. Suponiendo que hay memoria suficiente para el sistema operativo y para varios programas de usuario, cuando un trabajo necesite esperar una E/S, el procesador podría cambiar a otro trabajo que no esté esperando a una E/S, permitiendo mantenerlo ocupado. Esta técnica, conocida como **multiprogramación** o **multitarea**, se desarrolló a mediados en la década de 1960 y es la base de los sistemas operativos modernos.

Sistemas de tiempo compartido. En aquella época se comenzaban a idear aplicaciones que exigían un modo de operación en el que el usuario, sentado ante un terminal, interactuaba directamente con el computador. Este modo de trabajo es fundamental, por ejemplo, en el proceso de transacciones o consultas, y se denomina **interactivo**, en contraposición al *batch*. El procesamiento interactivo exige, por supuesto, multiprogramación, pero además debe proporcionar un **tiempo de respuesta** (tiempo que transcurre desde que se hace una transacción hasta que se obtiene la respuesta)

razonablemente corto. Es decir, es usuario que interacciona desde un terminal no puede estar a expensas de que el programa que ocupa el procesador lo abandone *motu proprio* por una entrada/salida. En un extremo, puede ocurrir que sea un programa orientado a cálculo y no requiera entrada/salida en mucho tiempo. Por esta razón, en los sistemas de **tiempo compartido**, introducidos en la segunda mitad de la década de 1960, el sistema operativo ejecuta los programas en ráfagas cortas de tiempo de cómputo o *quantum*, de forma intercalada. De esta manera, si hay *n* usuarios que solicitan servicio a la vez, cada usuario dispondrá en el peor de los casos (cuando ningún programa requiera E/S) de 1/*n* del tiempo del procesador. Dado el tiempo de reacción relativamente lento que tiene el ser humano, para un *quantum* suficientemente pequeño y un *n* no demasiado alto, el usuario no percibe un tiempo de respuesta apreciable y tiene la sensación de que disfruta de un procesador dedicado con una velocidad 1/*n* de la del procesador real. Esta idea se conoce como **procesador compartido**, y refleja el comportamiento ideal de un sistema de tiempo compartido, minimizando el tiempo de respuesta.

Una cuestión de precio

Es preciso fijarse en la evolución del factor coste en lo que se refiere a la tecnología para comprender el camino seguido por los modelos de gestión del sistema. Antes del desarrollo de las tecnologías de integración de circuitos, un computador costaba millones de dólares, estaba compuesto de decenas o centenares de miles de componentes electrónicos individuales (transistores y, previamente, válvulas), pesaba varias toneladas y ocupaba una gran sala climatizada. Con todo, sus prestaciones en cuanto a capacidad de proceso y almacenamiento eran comparables a las del chip contenido en una tarjeta inteligente de las de hoy en día. Puede entenderse entonces que en la década de 1960 se acometiera el desarrollo de sistemas operativos con multiprogramación y memoria virtual, capaces de sacar el máximo partido a estas máquinas (las configuraciones básicas del IBM/360, el mainframe más popular de esta época, venían con 8 Kbytes de memoria y ejecutaban unos pocos miles de instrucciones por segundo; aún así la CPU era muy rápida comparada con el lector de tarjetas perforadas). Hoy en día, los sistemas operativos siguen incluyendo memoria virtual, pero la mayoría de los ordenadores personales no la necesitarían.

Hoy en día, con sistemas multiprogramados de tiempo compartido, el proceso por lotes sigue teniendo sentido, por ejemplo en supercomputación.

Sistemas de teleproceso. En el esquema de tiempo compartido los terminales se conectaban al procesador por medio de cableado específico que se instalaba por el edificio. Cuando las grandes empresas y entidades (por ejemplo bancos y compañías aéreas) comenzaron a adquirir computadores, se encontraron con la necesidad de transmitir la información entre sus sucursales y la sede del computador. En estos escenarios, la instalación de cableado específico es económicamente inviable. Sin embargo ya existía el cableado telefónico, que se usó para transmitir la información digital mediante la utilización de un modulador-demodulador (*modem*) en cada extremo, conectado a la toma telefónica convencional. A diferencia de la transmisión con cableado específico, la comunicación telefónica es muy proclive a errores, por lo que hubo que elaborar protocolos de comunicación más sofisticados. Estos protocolos eran, en un principio, de tipo propietario (propios del fabricante del ordenador, que era también quien suministraba los terminales, los modems y el software).

Sistemas personales. El abaratamiento del hardware y la irrupción del microprocesador a finales de la década de 1970, hizo posible proporcionar un sistema dedicado para un único usuario a un coste reducido, una característica fundamental de unsistema personal. El sistema operativo de los computadores personales es, en un principio, **monousuario** (carece de mecanismos de protección) y monotarea; es decir, no muy diferente de los primitivos sistemas basados en monitor salvo por el hecho de usarse interactivamente mediante un terminal. Hoy en día el hardware disponible permite sistemas personales multitarea (Mac OS, Windows, Linux) que soportan **interfaces gráficas de usuario** sofisticadas. Otra característica del computador personal es que el usuario es el propio administrador del sistema, por lo que simplificar al máximo las tareas de administración es una necesidad (administración de coste cero).

Sistemas en red. Con la llegada del computador personal, los terminales de los sistemas de teleproceso se convierten en sistemas autónomos que pueden asumir determinadas tareas de cómputo, descargando al sistema central de tiempo compartido. En particular, pueden ejecutar de forma autónoma cualquier protocolo de comunicación. Una vez acordado un protocolo estándar (por ejemplo, TCP/IP), los ordenadores personales pueden comunicarse entre ellos. El concepto de computador central desaparece; ahora hay que hablar de conjunto de computadores que se conectan entre sí utilizando una infraestructura de red. Una máquina de la red que proporciona y gestiona el acceso a un determinado recurso es el **servidor** de ese recurso. Los **clientes** acceden al recurso mediante un esquema **cliente-servidor**. La aparición y amplia difusión de las redes han complicado sobremanera no solo el sistema operativo sino también los servicios que se implementan encima (conocido como *middleware*), dando lugar a **sistemas distribuidos** que se despliegan hoy en día en el ámbito de Internet y que han generado conceptos y esquemas de servicio muy elaborados, como los *servicios web* y el *cloud computing*. Aunque el presente curso se restringe al estudio de sistemas centralizados, no hay que perder de vista que la realidad es más compleja.

Ganadores y perdedores

En los primeros tiempos (años 50 y 60 del siglo XX), el sistema operativo se desarrollaba en lenguaje máquina por el propio fabricante de la arquitectura, que distribuÌa el sistema como un paquete indivisible. El sistema operativo y la arquitectura eran absolutamente interdependientes. Más tarde, tras la experiencia de UNIX y el lenguaje C, los fabricantes de hardware y software se especializaron, lo que permitía, en principio, tanto que un sistema operativo pudiera transportarse fácilmente a diferentes plataformas (el núcleo de UNIX apenas contenÌa 1000 líneas de código máquina, dependiente de la arquitectura), como que, en consecuencia, una arquitectura pudiera soportar diferentes sistemas operativos. Sin embargo, la introducción de los ordenadores personales hizo evidente la necesidad de algún tipo de estandarización en los sistemas operativos, tanto en cuanto a interfaz para las aplicaciones como para la interfaz de usuario. La estandarización llegó por la vía de los hechos a partir de dos factores: la alianza estratégica entre IBM y Microsoft, y la apertura de la plataforma hardware (PC) y software (interfaz MS-DOS) a otros fabricantes. Esto fue en detrimento de Apple, el gran competidor de Microsoft, que en los años 80 partía con una ventaja tecnológica indudable, pero que cerró la plataforma a sus propios productos. A medida que la arquitectura PC fue conquistando mercados, los sistemas de Microsoft lo hicieron con ella. La irrupción de Linux (derivado de Unix, un sistema diseñado para otro tipo de computadores) y la filosofÌa del software libre en los años 90 fue demasiado tardía para responder a la inercia monipolizadora de los sistemas Windows. La historia de los comienzos de los computadores personales se relata en el libro Fire in the Valley: The Making of a Personal Computer*, de Paul Freiberger y Michael Swaine, llevado a la pantalla por Martyn Burke con el título* Los piratas de Silicon Valley*.*

Sistemas móviles. La evolución del hardware no acaba con los computadores personales. Estos son cada vez más pequeños y, en consecuencia, se convierten en móviles gracias a una batería que les permite funcionar sin conexión a la red eléctrica. A su vez, se desarrollan las comunicaciones móviles, de modo que las redes se convierten en inalámbricas. En principio, esta evolución no afecta significativamente al sistema operativo. Sin embargo, con el nuevo siglo y de la mano de la evolución de la teléfonía móvil, aparecen nuevos dispositivos con capacidad de cómputo creciente. Estos, actualmente denominados teléfonos inteligentes o *smart phones*, son capaces de soportar versiones reducidas de los sistemas operativos diseñados para computadores personales (Mac, Windows, Linux), aunque también aparecen sistemas operativos específicos (como Symbiam, o Android de Google) con prestaciones nada desdeñables, incluyendo nuevas formas de interacción (pantallas táctiles, cámaras, información de posicionamiento…) y nuevas aplicaciones (como la navegación). Este sector constituye sin duda el escenario más candente para el desarrollo de la tecnología actual y futura de los sistemas operativos y se extiende a dispositivos de muy diverso tipo (por ejemplo cámaras, tarjetas inteligentes, o dispositivos de control *empotrados* en electrodomésticos o automóviles…) capaces de constituirse espontáneamente en red e interaccionar entre ellos sin intervención humana, dando lugar a comportamientos inteligentes. Para este tipo de sistemas se ha acuñado el término de **sistemas ubícuos**, y se suele hablar de *inteligencia ambiental* para referirse al tipo de aplicaciones que surgen en estos entornos.

Lo descrito arriba hace referencia a la línea de evolución principal de los sistemas operativos. Sin embargo, a medida que la tecnología informática fue copando ámbitos de aplicación, se han desarrollado tipos específicos de sistemas operativos. Un ejemplo remarcable es el de los **sistemas**

de tiempo real, comunes desde hace mucho tiempo en la industria (sistemas de control), y más modernamente en otros ámbitos (por ejemplo la descompresión de video en un sistema multimedia). Muchas veces estos tipos de sistemas están **empotrados** en sistemas más complejos (por ejemplo, el sistema de control de la estabilidad en un automóvil). En los sistemas de tiempo real los tiempos de respuesta están limitados por un plazo. Cumplido el plazo, la respuesta carece de validez o incluso el incumplimiento puede resultar catastrófico (piénsese en el control de estabilidad de un automóvil). Por este motivo se han desarrollado sistemas operativos para tiempo real específicos (por ejemplo QNX, FreeRTOS y muchos otros). Muchos sistemas operativos de propósito general también soportan tareas de tiempo real, pero solo son adecuados cuando el incimplimiento del plazo no es crítico (por ejemplo, aplicaciones multimedia).

6 Una clasificación de los sistemas operativos.

A la hora de clasificar los sistemas operativos actuales se pueden tener en cuenta distintos criterios, derivados de los conceptos introducidos más arriba. Una posible clasificación es la siguiente:

Monoprogramados/multiprogramados. También se habla de sistemas monotarea y multitarea. En los sistemas operativos primitivos, tanto los monitores como los primeros sistemas para computadores personales, por ejemplo MS-DOS, la ejecución de un programa debía terminar para que empezara la del siguiente. Estos sistemas se denominan monoprogramados. A partir de 1965 aparacen los primeros sistemas multiprogramados (OS/360, Multics). Hoy en día, la práctica totalidad de sistemas operativos son multiprogramados. En los sistemas multiprogramados, los programas se ejecutan **concurrentemente**, utilizándose el concepto de **proceso** (o tarea) para designar a un programa en ejecutión. Como se dijo más arriba, la multiprogramación estuvo motivada por la necesidad de optimizar el uso del procesador, y por lo tanto los procesos que ejecuta un sistema multiprogramado normalmente representan aplicaciones independientes. Más tarde la multiprogramación se ha utilizado para expresar la concurrencia en una misma aplicación, donde un conjunto de tareas cooperan de manera coordinada. Por ejemplo, en un procesador de textos podemos encontrar una tarea encargada de leer y procesar la entrada de teclado, otra tarea encargada de revisar la ortografía, una tercera tarea encargada de guardar periódicamente las modificaciones… Una clase particular de sistemas operativos multiprogramados son los sistemas multihilo o **multithread**, que permiten expresar la concurrencia en una aplicación de manera más eficiente. La diferencia entre un proceso y un hilo o *thread* (también llamado *subproceso*) es, para nuestros propósitos, de matiz, y no la vamos a abordar por el momento. La multiprogramación implica la multiplexación del procesador entre los procesos, como se explicó más arriba. Evidentemente, un sistema multiprocesador (un ordenador con varios procesadores) potencia la multiprogramación, permitiendo que la ejecución concurrente de los programas sea también **paralela**. Se habla entonces de **multiproceso**, y a los sistemas operativos que controlan estos

> ***El día de la marmota***
>
> *La larga historia de los sistemas operativos ha seguido una trayectoria cíclica. Sorprende conocer que conceptos tan sofisticados y técnicas tan complejas de implementar como la multiprogramación y la memoria virtual cuenten con casi medio siglo de historia y formaron parte de los primeros sistemas de tiempo compartido. Cuando, quince años después, irrumpieron los computadores personales, los primeros sistemas operativos desarrollados paa estos prescincían de estos mecanismos porque su limitado hardware no era capaz de soportarlos. De hecho, a parte del modo de trabajo interactivo, no eran muy diferentes de los primitivos monitores. Sin embargo, a medida que el hardware de los ordenadores personales fue ganando en prestaciones, sus sistemas operativos fueron integrando estas técnicas. Así, si en su momento se distinguía entre mainframes, estaciones de trabajo y ordenadores personales, hoy en día cualquier ordenador es capaz de soportar un sistema operativo complejo. Más recientemente, la miniaturización ha conducido a la aparición de disposivos de pequeño tamaño (los teléfonos móviles de hoy en día,* smart phones*, son el ejemplo más notable) con capacidad de cómputo y almacenamiento creciente. De nuevo, la historia se está repitiendo: si los primeros sistemas operativos para teléfonos móviles eran extraordinariamente simples, ya existen versiones reducidas de los sistemas operativos de propósito general destinadas a los teléfonos móviles y se van integrando prestaciones como la multitarea.*

sistemas se les llama **sistemas operativos multiprocesador**. Aunque existen notables diferencias en la implementación de un sistema operativo multiprocesador con respecto a uno monoprocesador, en lo que respecta a la visión funcional de aplicaciones y usuarios estas apenas transcienden.

Monopuesto/**multipuesto**. Un sistema operativo preparado para tener conectados al mismo tiempo distintos terminales se dice que es multipuesto; en caso contrario es monopuesto. Los sistemas operativos de tiempo compartido, como Unix, son multipuesto. Los sistemas diseñados para ordenadores personales —MS-DOS, Windows 95/98— son, naturalmente, monopuesto. Hay que reseñar el caso de Linux, un sistema Unix para ordenadores personales, que mantiene la filosofía multipuesto de Unix ofreciendo un conjunto de terminales virtuales. Mac OS X, también derivado de Unix, es otro ejemplo. Resulta evidente que un sistema multipuesto sea en algún modo multiprogramado: como veremos, lo normal es que cada terminal (real o virtual) tenga asociado un proceso que gestiona la conexión.

Monousuario/**multiusuario**. Un sistema multiusuario es capaz de proporcionar identificación de usuarios e incluye políticas de gestión de cuentas de usuarios y de protección de accesos que proporcionen privacidad e integridad a los usuarios. En los primitivos sistemas basados en monitor, compartidos por varios usuarios, esta función la llevaba a cabo manualmente el operador del sistema. Los primeros sistemas para computadores personales, como MS-DOS, eran monousuario. Los sistemas de propósito general de hoy en día son multiusuario. Obsérvese que algunos sistemas personales, como los teléfonos móviles, incluyen algún mecanismo de verificación (habitualmente una contraseña), pero carecen de políticas de protección de acceso a los recursos del sistema y de gestión de usuarios; simplemente autentican *al* usuario, siendo a todos los efectos sistemas monousuario.

7 El mercado de los sistemas operativos

Desde una perspectiva más cercana al mundo comercial es preciso referirse a dos grandes grupos de sistemas operativos. En primer lugar, aquellos sistemas operativos que han sido diseñados por un fabricante para una arquitectura concreta con el objetivo de proteger sus productos (tanto software como hardware) de posibles competidores se denominan **propietarios**. El fabricante diseña el sistema operativo específicamente para la arquitectura y proporciona las actualizaciones necesarias. Incluso a veces la especificación de la interfaz de llamadas al sistema no se hace pública o se modifica constantemente, dificultando el desarrollo de aplicaciones por otros fabricantes. Se crea así un mundo cerrado que engloba la arquitectura, el sistema operativo propietario y las aplicaciones, que permite el control del fabricante sobre el mercado de su producto y establece grandes dependencias para los clientes. Algunos ejemplos de sistemas operativos propietarios de gran difusión son (o han sido) los sistemas de IBM, VMS de Digital para VAX, los sistemas Mac de Apple, y los sistemas Windows de Microsoft para plataformas PC[1].

Con la aparición de Unix (hacia 1970) nace una nueva filosofía: al estar escrito casi completamente en un lenguaje de alto nivel (C), el sistema operativo es **transportable** a otras arquitecturas y por lo tanto también lo son las aplicaciones a nivel de código fuente. Además, en el caso de Unix, el código fuente se distribuyó libremente. Esto tuvo efectos contradictorios: por una parte contribuyó a la amplia difusión del sistema; por otra, cada fabricante introdujo sus propias modificaciones no solo en el código sino también en la interfaz de llamadas al sistema, de forma que hay que referirse a diferentes siatemas UNIX, no totalmente compatibles entre sí (System V, BSD, AIX, ULTRIX,

[1] Aún y todo, hay importantes diferencias entre sistemas propietarios. Así, Microsoft tuvo el acierto en los años 1980 de *abrir* la plataforma software (interfaz MS-DOS) a otros desarrolladores.

Solaris, Linux…). Como se aprecia en la Figura 2, el árbol genealógico de Unix es realmente complejo.

El ideal, un mundo de **sistemas abiertos**, con especificaciones públicas, aceptadas y estandarizadas, que permitan la transportabilidad plena de aplicaciones (y usuarios), es un objetivo escasamente logrado. En este sentido se han hecho esfuerzos para definir especificaciones estándar. Por ejemplo, la especificación POSIX es un referente en el mundo Unix. Un desarrollador que siga en las llamadas al sistema de su programa la especificación POSIX sabe que podrá compilarlo y ejecutarlo en cualquier sistema Unix que reconozca el estándar POSIX.

En este sentido, sería útil que los sistemas operativos se diseñaran con la capacidad de soportar diferentes interfaces de llamadas al sistema. Esta fue la filisofía de los micronúcleos, en la década de 1980, que implementaban las interfaces de las llamadas al sistema como servicios *fuera* del sistema operativo propiamente dicho (micronúcleo). Sin embargo, el desarrollo de sistemas operativos basados en micronúcleo ha tenido una repercusión comercial limitada. El más conocido es el micronúcleo Mach 3.0, en el que se basa el sistema Mac OS X de Apple. Sin embargo, lo más habitual hoy en día es soportar aplicaciones de sistemas hetereogéneos mediante *emulación* (virtualización), como aplicaciones sobre el sistema operativo anfitrión. Existen numerosos virtualizadores, como VMware, Virtual PC, o Win4Lin.

Hay que destacar un fenómeno que revolucionó el mercado del software y en particular de los sistemas operativos: la aparición espontánea de una comunidad de programadores que desarrollan **software libre**[1]. Internet constituye el medio necesario para la compartición y el intercambio ágil de ideas y código entre la comunidad. Como consecuencia, y así se ha demostrado ampliamente, se dinamiza la adaptación del software ante problemas particulares y el desarrollo de nuevos productos, y se corrigen errores y afinan versiones con gran agilidad. Organizaciones como GNU[2] otorgan licencia de copia, modificación y redistribución del software libre con la condición de que la nueva distribución incluya el código fuente[3]. Linux es un ejemplo hoy en día asentado de esta filosofía.

En la actualidad los sistemas operativos, más allá de su orientación original, han tenido que adaptarse a multitud de dispositivos, como es el caso de los teléfonos móviles y otros dispositivos de consumo. A ello hay que añadir los **sistemas empotrados**, cada vez más presentes en nuestro entorno (electrodomésticos, automóviles, instalaciones industriales, robots, etc). En algunos casos, los sistemas operativos convencionales se han adaptado a las restricciones de los dispositivos (de tamaño y potencia), como es el caso de Windows Mobile de Microsoft, iPhone OS de Apple o Palm OS; en otros casos se ha optado por desarrollos específicos, como es el caso de Symbian OS o de Android de Google. Los sistemas empotrados, además de restricciones físicas, presentan necesidades de tiempo real, en algunos casos críticas, que conducen a adoptar soluciones específicas, como ya se ha comentado.

8 Ejemplos de sistemas operativos

Vamos a analizar a continuación en más detalle la historia y principales características de los siatemas operativos más relevantes, en sintonía con los conceptos introducidos en los apartados

[1] *Free software*. No confundir con *freeware*. Tampoco debe entenderse como software gratuito.

[2] http://www.gnu.org

[3] Esta licencia se denomina *Copyleft*.

anteriores. Centraremos la atención en aquellas familias de sistemas operativos que han marcado época en la computación y cuyas innovaciones, directa o indirectamente, perduran en la actualidad.

Los grandes sistemas de IBM

IBM fue durante muchos años la empresa de computadores predominante en el mercado del hardware, los sistemas operativos y las aplicaciones. Su primer gran sistema operativo, OS/360, cuyo desarrollo terminó en 1964, era un complejo sistema multiprogramado por lotes que almacenaba las tareas en particiones (de tamaño fijo o variable, dependiendo de la versión). Una versión, TSS/360 (*Time Shared System*, 1967) ofrecía tiempo compartido y multiproceso (con dos CPUs), aunque su enorme complejidad (todos los sistemas de entonces se desarrollaban en ensamblador) provocó que nunca funcionase demasiado bien y que su difusión fuese escasa.

MVS (*Multiple Virtual Storage*, 1974) proporcionaba memoria virtual. Introdujo el concepto de máquina virtual, que permitía que varias copias del sistema operativo se ejecutasen en particiones lógicas independientes, proporcionando un alto grado de seguridad. La arquitectura MVS ha perdurado y forma parte hoy en día del sistema z/OS.

VMS de Digital

En torno a 1970 la introducción de los circuitos integrados había permitido abaratar sensiblemente el coste de los computadores y ampliar su ámbito de utilización. Surgió entonces el concepto de minicomputador para designar una gama de computadores de precio asequible (del orden de las decenas de miles de euros) y un tamaño reducido (como un armario pequeño). En aquélla época Digital Equipment Corporation triunfaba con la famila de minicomputadores PDP. El PDP-11, de 16 bits, fue la culminación de la saga. Funcionaba con el sistema operativo RSX-11, pensado para soportar aplicaciones de tiempo real.

La limitación inherente a la arquitectura de 16 bits llevó a Digital a introducir en 1977 la arquietctura VAX-11 (*Virtual Address eXtension*), de 32 bits y el sistema operativo VMS (*Virtual Memory System*). Una de las características de VMS es su capacidad de adaptación al variado nivel de soporte hardware de las diferentes implementaciones de la arquitectura VAX, sobre todo en cuanto a memoria virtual. Otra de sus características es que el sistema de ficheros gestiona versiones de los ficheros, identificadas con un sufijo que denota la versión y forma parte del nombre del fichero. Cuenta con una sofisticada política de planificación de procesos basada en prioridades dinámicas. Muchas de las ideas presentes en VMS se adoptaron en el desarrollo de Window NT. En 1991 se renombró como OpenVMS y se destinó a la arquitectura Alpha, sucesora de la VAX.

La familia UNIX

En 1970 se inició en los laboratorios Bell de AT&T el desarrollo de un sistema Unix, que tendría un gran impacto y desarrollo posterior. Sus antecesores fueron los sistemas CTSS y Multics. Este último, aunque no tuvo éxito comercial, marcó las pautas de los sistemas operativos futuros. Unix, cuya primera versión se desarrolló en lenguaje ensamblador sobre un PDP-7, se reescribió en 1972 enteramente en C (lenguaje desarrollado en los laboratorios Bell específicamente para el proyecto Unix), siendo el primer sistema operativo escrito en un lenguaje de alto nivel. En 1974 ya había una descripción pública del mismo.

AT&T distribuyó libremente Unix, con lo que gran cantidad de universidades y empresas lo utilizaron para sus ordenadores y desarrollos. Debido a ello tuvo gran cantidad de ramificaciones (**Ultrix** de Digital, **Xenix** de Microsoft, **AIX** de IBM, HP-UX de HP...), aunque básicamente se

distinguen dos familias: ***System V*** de AT&T y ***BSD*** de la Universidad de Berkeley cuya versión más popular comercializó Sun. Aunque esta última es más potente en lo referente al soporte de redes, ha habido una unificación alrededor de *SystemV Release 4 (SVR4)*, que en la versión de Sun se denominó **Solaris**.

También existen versiones de Unix para PCs siendo las más populares **SCO** o *Santa Cruz*, dentro de los comerciales y Linux y FreeBSD entre los de libre distribución. **Linux** es un proyecto iniciado por Linus Trovalds en la universidad de Helsinky a principios de la década de 1990 y que propone software de sistema operativo libre en la misma línea de GNU (licencia pública general) y *Free Software Fundation* en el campo de las aplicaciones. Linux está teniendo un enorme éxito no solo en pequeños servidores, sino también en equipos grandes. Su introducción en el mercado de los sistemas personales es cada vez mayor, gracias a importantes avances en tres áreas: facilidad de instalación, entornos gráficos amigables, y un creciente número de aplicaciones de ofimática de calidad.

La Figura 2 muestra, de forma simplificada, el árbol de familia de Linux.

Unix es multiprogramado, multiusuario y multipuesto y soporta distintas interfaces tanto alfanuméricas (shell, C-shell, K-shell...) como gráficas (Openwin, Motif, KDE, Gnome, ...). Las versiones más modernas soportan multiproceso.

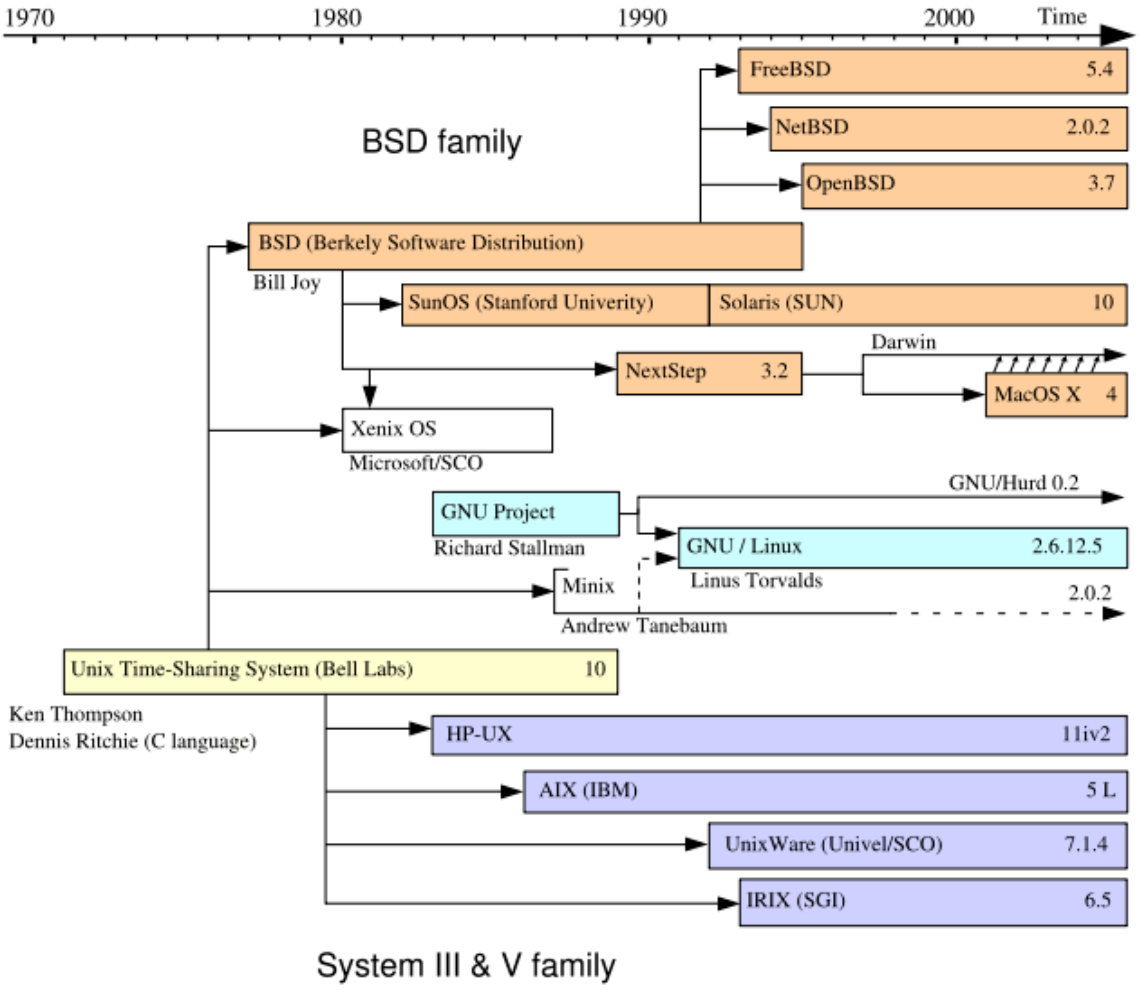

Figura 2*. La familia Unix (Wikipedia).*

Microsoft: de MS-DOS a Windows NT

Cuando en 1980 IBM decide adentrarse en el mundo de la informática personal, encarga a Microsoft el desarrollo de un sistema operativo para su nuevo PC. De esta manera, en agosto de 1981 IBM saca su primer PC con **MS-DOS** como sistema operativo. **MS-DOS 1.0** era compatible con CP/M, el sistema operativo que utilizaban la mayoría de los microprocesadores existentes hasta entonces, aunque también tenia mejoras significativas sobre éste. Mantenía mayor información sobre cada fichero, un mejor algoritmo de asignación de espacio en disco y era más eficiente. Sin embargo, sólo podía contener un único directorio de ficheros y dar soporte a un máximo de 64 ficheros. Ocupaba solamente 8 Kbytes.

Cuando apareció el PC XT (1983), que incluía un disco duro, Microsoft desarrolló la segunda versión de **MS-DOS**, con soporte para disco duro y directorios jerárquicos. También se le añadieron algunas características tipo Unix, como el redireccionamiento de la E/S.

En 1984, con el PC/AT, se incorporaba el procesador Intel 80286, provisto de direccionamiento ampliado y recursos de protección de memoria. Microsoft introdujo la versión 3.0 de MS-DOS, que no sacaba partido del nuevo soporte. Se hicieron varias actualizaciones notables de esta versión. La versión 3.1 incluía soporte para redes. A partir de aquí sucesivas versiones de MS-DOS van apareciendo sin grandes cambios estructurales.

Hay que destacar dos hechos que explican el éxito de MS-DOS: (a) la aparición, con el beneplácito de IBM, de PCs clónicos a bajo precio a los que Microsoft proveía de software —con lo que Microsoft consiguió colocar a MS-DOS como sistema operativo propietario— y (b) el mantenimiento de la compatibilidad con las versiones anteriores. Esto último provocó, por el contrario, que MS-DOS fuera un sistema menos desarrollado que otros de la competencia.

Después de que IBM optara por su propio sistema operativo OS/2, Microsoft lanza en 1990 **Windows 3.0**, copiando la idea de la interfaz gráfica comercializada anteriormente por Apple. Windows no es sino una interfaz sobre MS-DOS y no proporciona multitarea real. Aún así tuvo un gran éxito y su uso se extendió rápidamente.

Windows 95/98. En 1995 Microsoft ya había sacado Windows NT, un nuevo sistema operativo diseñado desde cero, para el mercado de servidores, pero el hardware de los computadores personales de la época era muy limitado para soportarlo. Por otre parte, Windows 3.11 era ridículamente primitivo en comparación con otros sistemas menos extendidos, como el Mac de Apple, que desde hacía tiempo ofrecía multitarea, protección de memoria y direccionamiento de 32 bits. En vista de ello, Microsoft se decide por un rediseño de Windows 3.11 para ofrecer esas características sin perder la compatibilidad con las aplicaciones de 16 bits de Window 3.x y MS-DOS. Los sistemas Windows 98 y Windows ME (Millenium Edition) son una continuación de Windows 95.

Windows NT/2000/XP/Vista/7. En 1988, Microsoft contrató a ingenieros de Digital, con experiencia en el desarrollo de VMS, para un nuevo proyecto de sistema operativo denominado Windows NT (*New Technology*). El objetivo es desarrollar un sistema operativo que integre los nuevos conceptos de diseño: arquitectura cliente/servidor basada en micronúcleo y soporte para multiprocesador, si bien la estructura de micronucleo se fue diiluyendo a través de las sucesivas versiones. Las primeras versiones —NT 3.1, de 1993, hasta NT 5.0, comercializada como Windows 2000— están orientadas a estaciones de trabajo y servidores. En 2001 se lanza la versión 5.1, comercializada como Windows XP, que incluye por primera vez una versión específica para uso doméstico, poniendo fin a Windows 95/98 y, con ello, a la línea de compatibilidad de las aplicaciones de 16 bits. Incluye versiones para procesadores de 64 bits. NT 6.0 (Windows Vista), lanzado en 2007, supone una fuerte revisión de la arquitectura, incluyendo una nueva interfaz

gráfica y prolijos mecanismos de protección, además de numerosos servicios. Todo ello resulta en una gran avidez de recursos que daja obsoleta una buena parte del parque de computadores personales. Le sucede en 2009 NT 6.1 (Windows 7), que afina la implementación para mejorar el rendimiento y actualiza las formas de interacción con el usuario.

Mac OS

En 1979 Xerox PARC cedió a Apple los derechos de utilización de su interfaz gráfica, que incluía elementos como iconos y ratón. Apple incluyó esta interfaz en el computador personal Lisa (1980), precursor del Macintosh (1984) y el sistema operativo Mac OS. Aparte de su avanzada interfaz gráfica, Mac OS ofrecia multiprogramación cooperativa (una forma de tiempo compartido en el que cada tarea es la responsable de ceder el procesador a otra tarea). En sus primeros años, el éxito de Macintosh fue enorme, pero su precio relativamente elevado y su estrategia de sistema cerrado motivaron que Microsoft, gracias sobre todo a su alianza con IBM, impusiera su MS-DOS, pese al retraso en introducir una interfaz gráfica decente.

Mac OS evolucionó hasta la versión 9 (1999). En el año 2000 Apple comercializa el nuevo Mac OS X, derivado de NeXTSTEP, un sistema operativo basado en el micronúcleo Mach 3.0. Mac OS X incorpora código de unix BSD y ofrece su interfaz de llamadas al sistema. Posteriormente Apple adoptó Intel como plataforma hardware en sustitución de las anteriores de Motorola.

Apple ha adaptado Mac OS X para sus dispositivos móviles, comercializado bajo la denominación iOS. La posición preponderante de Apple en este mercado le garantiza una buena difusión.

Bibliografía

A.S. Tanenbaum: *Modern Operating Systems (3rd edition)*. Prentice-Hall, 2008.

W. Stallings: *Sistemas Operativos. (5ª Edición). Pearson* Prentice-Hall, 2005.

Wikipedia: http://en.wikipedia.org