Introducción a los sistemas operativos

José Ramón Herrero Zaragoza
Enric Morancho Llena
Dolors Royo Vallés

PID_00169381

UOC
Universitat Oberta de Catalunya

Índice

Introducción

Un sistema computador está formado por los elementos siguientes:

- El hardware, que hace referencia a la máquina física.
- El software, que hace referencia al conjunto de programas que se ejecutan y que permiten sacar provecho de las prestaciones que ofrece el hardware.

Sin software, el computador es prácticamente un objeto sin ninguna utilidad. El software permite almacenar información, procesarla y recuperarla. En general, también permite entablar una serie de actividades muy diversas que justifican la inversión económica en el sistema computador.

Actualmente, un sistema computador consta de uno o más procesadores, memoria, terminales, discos, impresoras y tarjetas de red, entre otros. Los dispositivos que contiene son de diversos tipos (magnético, mecánico, láser o inalámbrico, entre otros), tienen un funcionamiento muy variado y en poco tiempo se pueden convertir en obsoletos, por lo que serán sustituidos por nuevos tipos de dispositivos. Además, en general hay más de un usuario que utiliza a la vez el sistema computador. Así, si un usuario quiere utilizar el sistema computador de manera eficiente y con una cierta protección de su información respecto del resto de usuarios, difícilmente lo podrá conseguir sin ningún tipo de ayuda. Es evidente que tiene que haber alguna manera de conseguir que los usuarios puedan utilizar el sistema computador sin que tengan que ser conscientes de la complejidad del hardware.

El problema se ha resuelto poniendo por encima del hardware una capa de software con el objetivo de gestionar las diferentes partes del sistema computador de forma eficiente y, al mismo tiempo, presentar al usuario una máquina virtual mucho más sencilla de comprender y utilizar. Esta capa de software es el llamado *software de sistema*, cuya parte más importante es el sistema operativo (SO).

Nota

Utilizamos las siglas SO como abreviación de *sistema operativo*.

Objetivos

En los materiales didácticos de este módulo, presentamos los contenidos y las herramientas necesarias para alcanzar los objetivos siguientes:

1. Tener una visión cuanto más clara mejor de lo que es un sistema operativo. A causa de la complejidad del concepto, formulamos su definición a partir de la descripción de las funciones que cumple el sistema operativo dentro del sistema computador y a partir del proceso evolutivo que han experimentado los sistemas operativos a lo largo de la historia, desde los primeros computadores hasta nuestros días.

2. Comprender la visión externa del sistema operativo: desde el punto de vista del usuario que utiliza el sistema operativo para consultar, modificar, crear, destruir y almacenar información y desde el punto de vista del usuario como programador de aplicaciones que se ejecutarán sobre este sistema operativo.

1. Clasificación del software. Definición de sistema operativo

En la figura 1, se muestra un esquema general del software que incluye un sistema computador:

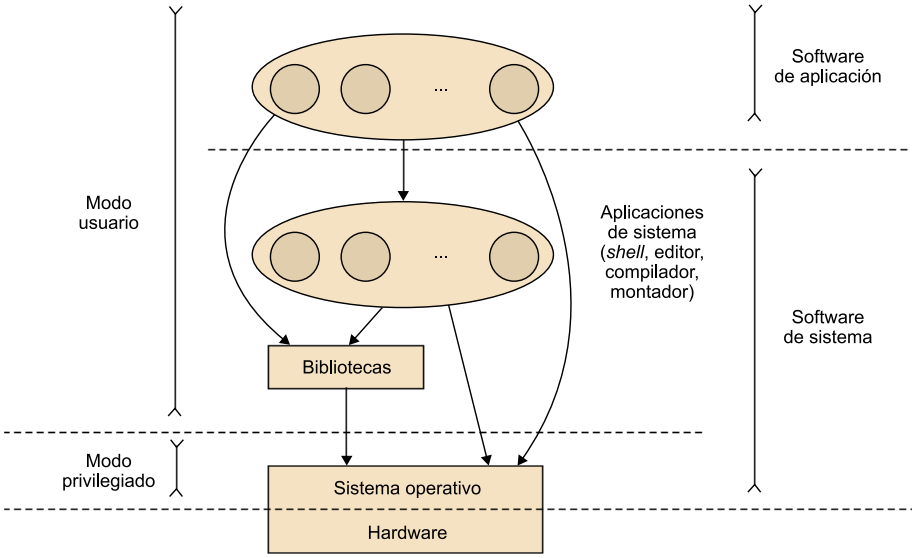


Figura 1. Esquema de un sistema computador en el que se muestra una clasificación de los tipos de softwares.

La capa inferior corresponde al **hardware** del computador. Por encima de ésta, encontramos el **sistema operativo**. Es muy difícil dar una definición exacta y precisa del término *sistema operativo*, que generalmente se define a partir de las funciones que desarrolla. Básicamente, son las dos funciones que presentamos acto seguido:

1) Gestión eficiente de los recursos del sistema:

> El SO controla el acceso eficiente a los recursos del computador: la memoria principal, la unidad central de proceso (CPU, procesador) y los dispositivos. Es decir, el SO se encarga principalmente de tareas de protección y de utilización eficiente del sistema computador.

En general, en un sistema computador tenemos muchos programas que se ejecutan al mismo tiempo, son programas que pueden pertenecer a uno o a varios usuarios, que compiten por los diferentes recursos del sistema (disco o memoria, por ejemplo). El sistema operativo reparte el tiempo de CPU entre los diferentes programas y consigue una ejecución concurrente, protege el acceso a la memoria[1] y coordina el acceso a los dispositivos (ya sean discos, memoria o impresora).

[(1)]El SO protege el acceso a la memoria entre diferentes usuarios y entre usuarios y el mismo SO. Esto se estudiará con detalle en el módulo 3.

2) Presentación a los usuarios de una máquina virtual mucho más sencilla de utilizar:

El sistema operativo proporciona un entorno de trabajo al usuario y a los programas de aplicación que permite utilizar el computador (ejecutar programas) de manera más fácil e intuitiva. Desde este punto de vista, el sistema operativo proporciona al usuario una máquina virtual mucho más fácil de entender y utilizar por el hecho de esconder la complejidad del hardware.

Lectura de un fichero

Por ejemplo, si no existiera el sistema operativo, a la hora de leer un fichero, el usuario necesitaría comprender y ocuparse de los detalles técnicos del disco: cómo ha sido formateado (por ejemplo FAT, NTFS o EXT2), la geometría del disco (como el número de caras, pistas o sectores), el posicionamiento de los cabezales y la lectura, entre otros. El sistema operativo nos permite hacer la lectura de datos de un fichero con una operación mucho más sencilla e intuitiva, que no exige al usuario ningún conocimiento detallado del funcionamiento del disco. Con la operación read(nombre_fichero,pos,reg) se indica que se quiere leer del fichero nombre_fichero del registro reg, que está en la posición pos. Más adelante, veremos que estas operaciones reciben el nombre de llamadas al sistema (*system calls*) y el conjunto de llamadas al sistema ofrecidas por el sistema operativo constituyen la API (*application programming interface*) del sistema operativo.

Por encima del sistema operativo tenemos **el resto de software de sistema**, que consta de bibliotecas de rutinas y de aplicaciones de sistema como el intérprete de comandos (*shell*), compiladores, editores y, en general, programas que facilitan la comunicación entre el sistema operativo y el usuario.

Bibliotecas de rutinas

En algunos textos escritos en catalán o en castellano, se utiliza el término *librería* en lugar del término *biblioteca* a causa de una traducción incorrecta del término inglés original (*library*). Una biblioteca de rutinas es un conjunto de rutinas de uso común y que se encuentra disponible para todos los usuarios.

Un usuario puede tener la sensación de que el intérprete de comandos y las demás aplicaciones del sistema también forman parte del sistema operativo porque estas aplicaciones se instalan junto con el sistema operativo. Además, muchos usuarios no tienen que ser conscientes de la existencia de las llamadas al sistema porque interaccionan con el sistema operativo de forma indirecta utilizando estas aplicaciones. Es importante darse cuenta de la diferencia entre el sistema operativo y el resto de software de sistema:

1) Una manera de darse cuenta de la frontera existente dentro del software de sistema es que un usuario cualquiera de un sistema computador no tendría que estar obligado a utilizar las aplicaciones de sistema disponibles; si quiere, tendría que poder utilizar otras o incluso implementarlas él mismo, sin necesitar ninguna autorización especial. Por ejemplo, el usuario puede seleccionar el intérprete de pedidos, editor, compilador, etc. que quiere utilizar. En cambio, todos los usuarios están obligados a utilizar el software que forma parte del sistema operativo; únicamente usuarios autorizados pueden introducir modificaciones dentro del código del sistema operativo.

2) Otra diferencia entre el sistema operativo y el resto de software es que, al fin y al cabo, todo el software se ejecuta sobre el procesador como instrucciones de lenguaje máquina. Para garantizar que los sistemas operativos puedan hacer su trabajo correctamente, es necesario que el procesador sea consciente de cuándo está ejecutando el código propio del sistema operativo. Eso es debido al hecho de que el código propio del sistema operativo tendría que poder llevar a cabo operaciones privilegiadas, mientras que el resto del software no. Los procesadores disponen de diversos modos de ejecución: un modo de ejecución privilegiado para ejecutar código propio del sistema operativo y un modo no privilegiado (usuario) para ejecutar cualquier otro código (bibliotecas, aplicaciones de sistema, aplicaciones de usuario).

Software y lenguaje máquina

Encontraréis más información sobre el software y su ejecución sobre el procesador como instrucciones de lenguaje máquina en el módulo 2.

3) Sin embargo, es posible que en algún documento encontréis que se hable de *sistema operativo* para referirse a todo el software de sistema presentado en la figura 1. En este caso, se acostumbra a utilizar el término *núcleo* o *kernel* del sistema operativo para referirse a la capa que se encuentra inmediatamente encima del hardware del sistema computador.

Finalmente, sobre el software de sistema está el **software de aplicación**. Una parte de este software la utiliza el usuario sobre todo para resolver problemas específicos. En este nivel encontramos hojas de cálculo, procesadores de textos, navegadores web, lectores de correo electrónico y juegos, entre otros.

2. Visión histórica de los sistemas operativos

Para entender bien qué es un sistema operativo y para qué sirve nos puede ayudar a conocer su evolución. A continuación, explicamos cómo han evolucionado los sistemas operativos desde las versiones más primitivas (monitores residentes y sistemas operativos por lotes), pasando por los sistemas operativos multiprogramados y con tiempo compartido, hasta los sistemas operativos distribuidos y los sistemas operativos en red, los sistemas operativos en tiempos reales y para dispositivos empotrados (*embedded*).

No es fácil separar el desarrollo de la arquitectura de los computadores de la evolución de los sistemas operativos, esencialmente porque los sistemas operativos han evolucionado en función de los cambios tecnológicos para intentar sacar el máximo provecho de las nuevas tecnologías. A su vez, éstas han dado soluciones a muchos problemas que han ido surgiendo en los sistemas operativos.

2.1. Los primeros sistemas computadores (la primera generación)

Al principio, sólo existía el hardware del computador. La tecnología utilizada en la construcción de los primeros computadores eran las válvulas de vacío, que obligaban a construir unos aparatos monstruosamente grandes.

En esta primera época, no había sistema operativo y el usuario interaccionaba directamente con el hardware del computador. Para programar estos aparatos había que configurar manualmente una serie de interruptores y de cuadros de conexión. Evidentemente, la programación de la máquina a este nivel era muy poco productiva, tanto para el usuario como para el hardware. El proceso de introducción de programas, que era largo y complicado, hacía inviable la ejecución de programas medios y grandes en este entorno.

Estos primeros computadores se utilizan para efectuar cálculos matemáticos en cuestiones militares, por ejemplo, para calcular trayectorias balísticas y tablas de senos y cosenos.

Hacia el final de esta época empiezan a aparecer dispositivos como las impresoras de cintas de papel y los lectores de tarjetas perforadas. Eso permitió codificar los programas utilizando tarjetas perforadas y almacenar el resultado en cintas de papel o bien en tarjetas perforadas.

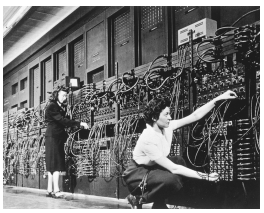


Figura 2
El ENIAC, *Electronic Numerical Integrator and Computer* (computador e integrador numérico electrónico), pesaba aproximadamente 30 toneladas, ocupaba 150 m^2 y consumía una potencia de 150 kW.

2.2. La segunda generación

La aparición del transistor en la década de 1950 permitió construir computadores mucho más pequeños y más fiables, de manera que ya se podían fabricar con la idea de venderlos. Eran máquinas muy caras y sólo estaban al alcance de las grandes corporaciones, instituciones como el ejército, las universidades y el gobierno.

Paralelamente, constatamos los hechos siguientes:

- Aparecen dispositivos de entrada/salida nuevos: los lectores de tarjetas perforadas y las impresoras.

- También se inventan dispositivos de almacenamiento nuevos: las unidades de disco y las cintas magnéticas.

- Se desarrolla el primer software de sistema, que incluye ensambladores, compiladores de lenguajes de programación de alto nivel[2], cargadores, montadores (*linkers*), bibliotecas de funciones matemáticas y rutinas estándar de entrada/salida de los diferentes dispositivos que pueden utilizar los programas[3].

[(2)] Algol, Cobol o Fortran

[(3)] Impresoras, monitores y tarjetas perforadas

Con este nuevo entorno, se mejora considerablemente el proceso de programación y ejecución de programas respecto de la etapa inicial. En esta etapa, el proceso tiene lugar siguiendo los pasos siguientes:

Figura 3
Las tarjetas y las cintas perforadas se utilizaban para programar los computadores y para recibir los resultados de sus cálculos mediante máquinas lectoras-perforadoras de tarjetas.

1) Los programas se escriben en lenguajes ensambladores o de alto nivel; una vez creados se llaman código fuente *(source code*).

2) Los programas se traducen de manera automática mediante los compiladores/traductores (aplicaciones de sistema) a lenguaje máquina:

- Si hay errores de sintaxis, hay que corregirlos y volver a hacer la traducción.
- Si no hay errores sintácticos, el código objeto producido a partir del código fuente por el traductor/compilador se puede cargar en memoria y ejecutar.

3) Otra aplicación de sistema, el cargador, automatiza el proceso de cargar el programa en memoria para ejecutarlo.

4) Después de transferir el control al programa cargado por medios manuales o automáticos, empieza la ejecución del programa.

Una vez cargado en memoria, el programa se puede reejecutar con diferentes datos y los resultados se sacan por la impresora o en cinta de papel. Si se detectan errores en los tiempos de ejecución, es posible determinar dónde se han producido mediante un vertido de la memoria y de los registros del computador o bien utilizando el programa depurador (*debugger*).

En este sistema, las rutinas de entrada/salida, junto con el programa cargador, se pueden considerar una primera versión muy rudimentaria del sistema operativo. Los traductores de lenguaje de alto nivel, los editores y los depuradores son aplicaciones de sistema que se incluyen en los diferentes servicios que ofrece el sistema operativo, pero generalmente se considera que no forman parte del mismo.

A pesar de todas estas mejoras, el modo de operación seguía siendo poco eficaz a causa de las diversas operaciones manuales que había que efectuar y que comportaban una utilización baja de los recursos del sistema. Como el coste económico del sistema computador era elevado, no es extraño que se buscaran maneras de aprovechar el tiempo perdido y aumentar la utilización de los recursos del sistema con el fin de eliminar los tiempos muertos generados por las operaciones manuales.

El primer objetivo fue automatizar al máximo todas las operaciones utilizando hardware. Pero, a pesar de la automatización de estas operaciones y la mejora de los dispositivos[4], la velocidad de las operaciones de entrada/salida de datos seguía siendo demasiado lenta en comparación con la velocidad del procesador y la de la memoria.

(4)Uso de cintas magnéticas en lugar de tarjetas perforadas

Como no era posible reducir el coste temporal de las operaciones de entrada/salida, se intentó al menos efectuarlas el mínimo número de veces posible. La idea era agrupar los trabajos de manera que las operaciones auxiliares tan sólo se ejecutaran una vez para todos los trabajos del grupo. Este sistema se llama **sistema por lotes** (sistema *batch*).

> El sistema por lotes podría funcionar, por ejemplo, de esta forma: agrupando en un lote todos los programas escritos en lenguaje de alto nivel Fortran sólo se tiene que cargar una vez el compilador de Fortran y los trabajos que se hacen dentro del lote se pueden procesar el uno detrás del otro con un secuenciamiento automático. Para que sea posible este secuenciamiento automático, se tiene que poder indicar al sistema operativo al menos cuándo empieza y cuándo finaliza un programa dentro del lote. En este punto, hay que introducir el concepto de lenguaje de control de trabajos (JCL[5]). Con este lenguaje, se podía indicar al sistema operativo el inicio y la finalización de un trabajo, dar órdenes de carga y ejecución y solicitar ciertos recursos del computador (como el tiempo de CPU o la cantidad de memoria).

(5)Las siglas JCL provienen de la abreviación de la expresión inglesa *job control language*.

En los computadores de la segunda generación, la ejecución de los programas era totalmente secuencial. Cuando un programa se detenía para esperar la finalización de una operación de entrada/salida (de una unidad de cinta o de

cualquier otro dispositivo), la unidad central de proceso (CPU) sencillamente se detenía y también se esperaba que la operación de entrada/salida acabara. En el caso de programas científicos, en los que domina el cálculo de operaciones en coma flotante (que se efectúa en la CPU) y las operaciones de entrada/salida son poco frecuentes, este sistema de trabajo ya era bastante eficiente.

A pesar de esta automatización, la CPU seguía estando mucho tiempo parada. El problema principal era la diferencia de velocidades entre la CPU y los dispositivos: las CPU en este periodo eran lentas, pero los lectores de tarjetas todavía lo eran más. Por lo tanto, la CPU estaba mucho tiempo detenida esperando que finalizara una operación de entrada/salida. Para intentar reducir al máximo estas diferentes velocidades de trabajo, apareció el **trabajo fuera de línea** (*off-line*). Con este sistema, se intentaba solapar las operaciones de entrada/salida con la ejecución de programas. Mientras los programas escritos en tarjetas se leían y se pasaban a cinta en una máquina, en otra máquina se podían ejecutar otros programas que ya habían sido cargados anteriormente en otra unidad de cinta.

Posteriormente, se idearon otras técnicas de solapamiento como el **almacenaje en la memoria intermedia** (*buffering*), que consiste en solapar las operaciones de entrada/salida de un programa con las operaciones de cálculo del mismo programa.

La aparición del disco, dispositivo de acceso directo, proporcionó una nueva técnica de trabajo: la **gestión de colas** (*spooling*), que permitía solapar las operaciones de entrada/salida de diferentes trabajos mientras se estaban ejecutando otras operaciones.

2.3. La tercera generación

La tercera generación de computadores surge a mediados de la década de 1960 y tecnológicamente se basa en los circuitos integrados. Estas máquinas nuevas son mucho más pequeñas que las de la segunda generación y también son mucho más rápidas.

Con la aparición de computadores más pequeños y fiables gracias a las nuevas tecnologías y a la posibilidad de incrementar la producción de computadores, los costes de fabricación se abaratan y así nace un sector informático comercial muy importante. Los computadores se ponen al alcance de empresas medianamente grandes y empieza el uso comercial de la informática.

Figura 4
La serie IBM 360. El primer grupo de computadores construidos con circuitos integrados fue el de la serie 360 de IBM.

Otro aspecto muy importante de esta nueva generación es el hecho de que se introducen mecanismos nuevos, como interrupciones, protección o búsqueda de una concurrencia máxima de operación. Paralelamente, se desarrollan el hardware y el software, que dieron lugar a los elementos siguientes:

- periféricos más eficaces y rápidos;

- terminales remotos (con los que se puede acceder a bases de datos y actualizar cuentas corrientes, entre otros);

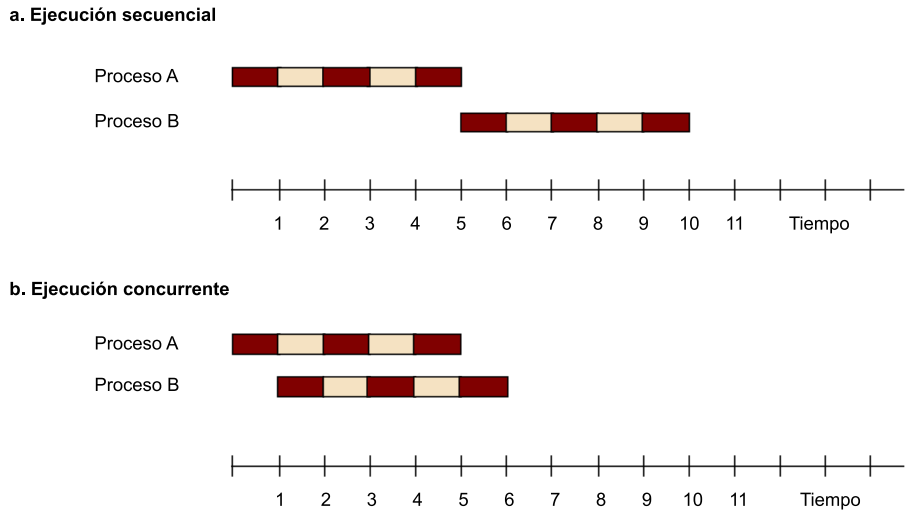
- lenguajes de programación de uso general, pensados para que pudieran ser utilizados por diferentes computadores con características de hardware muy diversas. Se estandarizan los lenguajes de alto nivel ya existentes (Fortran, Algol, Cobol, PL/1) y aparecen otros, como el Basic y el Pascal.

Una de las técnicas más innovadoras de esta generación es la **multiprogramación**, que permite solapar la ejecución de diversos programas.

En el procesamiento de datos comerciales, el tiempo de espera en razón de operaciones de entrada/salida puede representar el 90% del tiempo total de ejecución. Si nuestro computador está ejecutando únicamente un programa, estaremos desperdiciando el procesador todo este tiempo de espera. La solución del problema será la introducción de la multiprogramación, es decir, la capacidad de multiplexar la utilización del procesador entre diversos programas de manera que, aparentemente, el procesador esté ejecutando diversos programas de forma simultánea.

Durante la ejecución de un programa, generalmente podemos distinguir entre fases de cálculo intensivo y fases en las que se efectúan operaciones de entrada/salida de manera intensiva. Eso se ilustra en la figura 5, en la que las fases de cálculo intensivo se indican mediante rectángulos grises y las de entrada/salida, con rectángulos blancos. La ejecución secuencial de dos programas se muestra en el punto *a. Ejecución secuencial*. Para simplificarlo, supondremos que los programas tienen un comportamiento idéntico.

Figura 5



Rectángulos oscuros: fases de cálculo intensivo. Rectángulos claros: fases de entrada/salida.

Tal como se puede deducir de la figura 5, durante la ejecución secuencial de los dos programas, la CPU o bien los dispositivos de entrada/salida están inactivos en algún momento. Una manera de solucionar este problema es asignar alguna otra tarea a la CPU o a los dispositivos de entrada/salida cuando, por algún motivo, estén inactivos. Si es posible ejecutar los programas de manera concurrente, éstos pueden proporcionar trabajo tanto a la CPU como a los dispositivos en cualquier momento durante la ejecución de los dos programas.

En el punto *b. Ejecución concurrente* de la figura 5 se muestra un posible escenario de la ejecución concurrente de los dos programas. El procesador empieza ejecutando el primer proceso, el proceso A, y, cuando éste lleva a cabo una operación de entrada/salida, el procesador, en lugar de quedarse inactivo, pasa a ejecutar el segundo proceso, el proceso B, que se supone que se encuentra en la memoria esperando a ser ejecutado. Cuando este segundo proceso se acaba o bien solicita una operación de entrada/salida, el procesador vuelve a ejecutar el proceso A. El proceso sigue alternando la ejecución de los dos procesos hasta que finalizan los dos.

Tal como se puede ver en el esquema de la figura 5, con la ejecución concurrente, o multiprogramada, se puede incrementar considerablemente el rendimiento del computador: el tiempo de ejecución de los dos programas queda reducido de diez unidades en la ejecución secuencial a seis unidades en la ejecución concurrente y la utilización del sistema en este segundo caso es de un 100%. El ejemplo que hemos presentado, aunque es muy representativo, no es real, ya que generalmente la distribución de las fases de cálculo y programación son más variables, de manera que no siempre se consigue que el procesador y los dispositivos estén activos.

Para aumentar la utilización de los recursos del sistema computador, los sistemas multiprogramados permiten la ejecución concurrente de más de dos programas, de los que se dice que están compitiendo por los recursos del sistema. El número de programas que están activamente en competencia por los recursos del sistema determina el **grado de multiprogramación**.

La aparición de la multiprogramación permite desarrollar sistemas con múltiples usuarios conectados simultáneamente mediante terminales a un sistema computador. En este caso, el sistema multiprogramado puro no es muy eficiente, ya que, tal como hemos explicado anteriormente, la CPU deja de ejecutar un programa y pasa a ejecutar otro que esté pendiente de ejecución siempre que el programa que se está ejecutando establece una operación de entrada/salida o bien cuando finaliza. A causa de este funcionamiento, el tiempo de respuesta que tiene un usuario que está conectado en modo interactivo puede ser considerablemente largo.

Para solucionar este problema, se propone un nuevo mecanismo llamado **tiempo compartido** (*time sharing*). Con este mecanismo, los diferentes programas que compiten por los recursos del sistema, entre los cuales está la CPU, tienen asignada la CPU durante un periodo de tiempo limitado, llamado *quantum*[6]. Así, cuando un programa deja de ejecutarse, libera la CPU por alguno de los tres motivos siguientes:

(6)En el sistema operativo Linux, la duración del cuanto es un parámetro de configuración que se puede escoger entre los valores 1 ms, 4 ms y 10 ms.

- porque el programa finaliza,
- porque el programa inicia una operación de entrada/salida,
- porque ha pasado un cuanto de tiempo en ejecución.

El cuanto de tiempo es lo bastante pequeño para que los programas de los diferentes usuarios que están conectados al sistema progresen de forma concurrente y cada usuario tenga la impresión de que es el único que utiliza el sistema.

En esta generación, podemos enmarcar el nacimiento del sistema operativo UNIX, precursor del sistema operativo Linux. Ken Thompson, un científico de los laboratorios Bell, decidió implementar una versión simplificada de un ambicioso –pero sin éxito– proyecto de sistema operativo llamado MULTICS. Las buenas críticas de la primera versión del UNIX le animaron a escribir una nueva versión para una de las máquinas más populares de la época (PDP-11) y, por primera vez en la historia de los sistemas operativos, utilizando un lenguaje de alto nivel (el lenguaje C, creado para la ocasión). Como AT&T (propietaria de los laboratorios Bell) no estaba autorizada a hacer negocio en el sector informático, hacia 1974 empezó a distribuir el código fuente del UNIX a las universidades que lo pedían. A partir de ese momento, muchas universidades empezaron a utilizar, estudiar y mejorar el UNIX. Hacia 1984, AT&T recibió la autorización para operar en el sector informático y empezó a comercializar el UNIX System III y, posteriormente, el UNIX System V. Ahora bien, la Universidad de Berkeley también distribuyó su versión del UNIX (BSD), que entre otras mejoras incorporaba el protocolo de comunicaciones TCP/IP, que ha resultado clave en el desarrollo de la red Internet. Además, muchas empresas comercializaron sus versiones del UNIX (como HP-UX, AIX, Solaris, Sun-OS o Ultrix). Esta gran familia de sistemas operativos no era totalmente compatible, por lo que se iniciaron diversos esfuerzos para estandarizar el sistema operativo UNIX. El esfuerzo más destacado fue POSIX (*portable operating system interface*).

2.4. De la cuarta generación hasta ahora

Con la aparición de la integración a gran escala (LSI/VLSI), se inicia la era de los computadores personales (1974), a los que se aplican todos los mecanismos desarrollados hasta ahora (tanto de hardware como de software). Seguramente, la aportación más importante de esta nueva generación son las redes de computadores (Internet e intranet).

A partir de este momento y hasta nuestros días, los computadores se utilizan en cualquier ámbito: los usuarios en su casa (computador personal) o en la calle (PDA, teléfonos móviles), electrodomésticos, las pequeñas y grandes empresas (software de gestión), las corporaciones, las universidades y los centros de investigación (supercomputadores y multicomputadores). En general, hay una explosión de recursos: nuevos dispositivos cada vez más avanzados (como impresoras láser o comunicaciones inalámbricas) y CPU cada vez más rápidas, sistemas con más memoria y discos con más capacidad.

Surgen los **sistemas operativos en red y los distribuidos**, que permiten utilizar recursos de máquinas remotas (como discos, CPU o software) con diferentes niveles de abstracción.

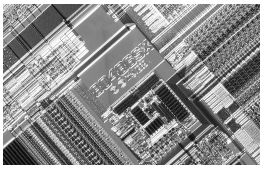


Figura 6. Interior del procesador 80486 de Intel.

También se introduce el concepto de **sistema operativo en tiempo real** referido a los sistemas que tienen que procesar datos, que generalmente se obtienen del exterior (sensores), en un tiempo limitado y dentro de periodos de tiempo determinados. Es importante ver que estos sistemas operativos se centran en conseguir un tiempo de respuesta efectivo, mientras que la utilización adecuada de recursos (gestión eficiente de la memoria) y la perspectiva del usuario quedan en un segundo plano.

Aplicaciones

Los sistemas operativos en tiempo real se utilizan en entornos industriales, en equipos de conmutación telefónica, en control de vuelo en aviones y en simulaciones en tiempo real.

En esta generación, podemos situar la aparición de los **sistemas operativos encajados** (*embedded*). Estos sistemas operativos se utilizan en computadores que controlan aparatos como televisores, reproductores de MP3, reproductores de DVD, lavadoras o teléfonos móviles sencillos. Estos dispositivos están diseñados para un uso muy concreto y el usuario difícilmente podrá instalar nuevo software. El hecho de que estos sistemas no tengan un propósito general permite simplificar el diseño de sus sistemas operativos.

También podemos situar la aparición de los **sistemas operativos para computadores de bolsillo** (*handheld computers*). Aquí, podemos incluir una serie de sistemas computadores prácticamente de propósito general, como PDA, teléfonos móviles o consolas. Estos sistemas operativos cada vez son más sofisticados porque cada vez incorporan más prestaciones, permiten que el usuario instale nuevo software y tienen que controlar periféricos como pantallas gráficas, pantallas táctiles, cámaras de fotos, GPS o comunicaciones inalámbricas. Además, a diferencia de otros entornos, tienen que gestionar la energía de forma eficaz para aumentar el tiempo de autonomía de los equipos.

Hay que señalar que, según qué uso se haga del sistema computador, interesa que el sistema operativo proporcione herramientas específicas a los usuarios que lo tienen que utilizar. Así, pues, se tienen que conseguir los objetivos siguientes:

- Los sistemas en red o los distribuidos tienen que desarrollar gestores de comunicación eficientes.

- Los sistemas multiprogramados/multiusuario se tienen que centrar en conseguir tiempos de respuesta aceptables.
- Los sistemas en tiempo real interesa que sean cuanto más sencillos mejor para poder procesar datos del exterior tan rápido como sea posible.

En esta generación, podemos enmarcar el nacimiento de dos de los sistemas operativos más utilizados en el ámbito doméstico: la versión 1.0 del sistema operativo MS-DOS y el Linux.

La **versión 1.0 del sistema operativo MS-DOS** fue introducida en 1981 por una modesta compañía de software llamada Microsoft. El MS-DOS era monousuario y tenía bastantes limitaciones y problemas de estabilidad debido a la falta de dos modos de ejecución (usuario y privilegiado) del hardware en el que se ejecutaba (el procesador 8088 de Intel). El MS-DOS fue evolucionando para adaptarse a los nuevos dispositivos de almacenamiento e incorporar nuevas prestaciones. Hacia 1990 apareció el Windows 3.0, una interfaz gráfica para el MS-DOS, pero el sistema operativo seguía teniendo bastantes problemas de estabilidad (la fatídica pantalla azul). Hacia 1993 apareció la línea Windows NT, seguida del Windows XP (2001), Windows Vista (2006) y Windows 7 (2009). Estos últimos sistemas ya eran multiproceso y multiusuario y consiguieron alcanzar grados muy altos de estabilidad porque aprovechaban las características avanzadas de los procesadores Intel. Para poder desarrollar aplicaciones portátiles entre las últimas versiones del Windows, Microsoft ha definido una interfaz estándar llamada Win32.

El **Linux** nació en 1991 cuando un estudiante de informática finés, Linus Torvalds, decidió implementar una versión del UNIX que siguiera el estándar POSIX para procesadores 80386. Linus utilizó Internet para buscar una comunidad de voluntarios que le ayudara a mantener y mejorar el sistema operativo. El Linux se ofrece con una licencia de software libre, de manera que el código fuente está disponible para todo el mundo. Ahora bien, para hacer usable el sistema operativo hay que disponer de software de sistema (como intérpretes de comandos, compiladores y editores). Las distribuciones del Linux (Debian, Red Hat, Fedora, Knoppix, Ubuntu, entre otras) van acompañadas por las aplicaciones desarrolladas por la comunidad GNU. Por lo tanto, algunos hablan de distribuciones GNU/Linux para referirse a todo el software de sistema utilizado en las instalaciones del Linux. Este sistema está en evolución constante, se adapta a nuevas arquitecturas y cada vez es más amigable para usuarios no expertos y para usuarios acostumbrados a los sistemas Windows.

3. Los servicios que ofrece el sistema operativo

El sistema operativo ofrece una gran variedad de servicios que iremos ampliando en los siguientes módulos.

Si nos fijamos en su función, los servicios se pueden clasificar en gestión de procesos, señalización de procesos, gestión de dispositivos de entrada/salida, gestión directa de los recursos del sistema, gestión del sistema de archivos, protecciones y funciones de tiempo. A continuación, describimos algunas de estas clases:

1) Gestión de procesos: el concepto de proceso es uno de los más importantes de la asignatura, engloba todos los recursos necesarios para ejecutar un programa. En este apartado, englobamos los servicios que hacen referencia a la creación y eliminación de procesos, pero también podemos encontrar otros servicios, como la suspensión y la reanimación de procesos. También podemos modificar atributos de ejecución, como es la prioridad o el tiempo de cuota de CPU asignado.

2) Señalización entre procesos: en un sistema multiproceso, el nexo natural de unión entre todos los procesos es el sistema operativo. Éste ofrece servicios para comunicar y sincronizar procesos.

3) Gestión de dispositivos de entrada/salida: los servicios relacionados con el sistema de entrada/salida tienen como funciones principales crear, abrir y cerrar canales de entrada/salida, así como escribirlos y leerlos.

4) Gestión directa de los recursos del sistema: este grupo de servicios es muy específico de cada sistema operativo, pero un ejemplo común es la gestión de la memoria. Algunas de las operaciones que se efectúan con la memoria del sistema son asignar, liberar, proteger y compartir.

5) Gestión del sistema de archivos: junto con el bloque de servicios de entrada/salida, también es un grupo de servicios muy utilizado. La gestión del sistema de ficheros incluye funcionalidades como crear y eliminar ficheros y directorios o cambiar de directorio.

Para poder solicitar estos servicios hay que utilizar las **llamadas al sistema** que ofrece el sistema operativo. Ahora bien, aunque las llamadas al sistema resultan un mecanismo muy cómodo para los usuarios programadores, no es así para los usuarios finales del sistema operativo. Por lo tanto, mientras que los usuarios programadores solicitarán estos servicios directamente utilizando

llamamientos al sistema, los usuarios finales solicitarán estos servicios de forma indirecta, utilizando el intérprete de órdenes o de otras aplicaciones de sistema.

1) Las **llamadas al sistema** permiten realizar peticiones directas al sistema operativo. Las llamadas al sistema son la visión que tiene un programador de los servicios que puede ofrecer el sistema operativo. Los lenguajes de alto nivel en última instancia hacen una llamada al sistema.

2) A un nivel más próximo al usuario, el software de sistema ofrece un programa, el llamado **intérprete de comandos**, gracias al cual el usuario puede dialogar con el sistema operativo sin necesidad de escribir ningún programa.

3.1. Las llamadas al sistema

Las llamadas al sistema (*system calls*) ofrecen las funciones básicas para poder utilizar todos los recursos del sistema de manera correcta y controlada.

Ved también

En la figura 1 del apartado 1, las llamadas al sistema corresponderían a los diversos puntos de entrada a la caja del sistema operativo.

Desde el punto de vista del programador, una llamada al sistema no es más que una rutina con una serie de parámetros de entrada, una serie de parámetros de salida y una semántica asociada. Cada sistema operativo ofrece un repertorio de llamadas al sistema (como referencia, la versión 2.6.32 del sistema operativo Linux ofrece 337 llamadas al sistema). Este repertorio de llamadas al sistema constituye la API (*application programming interface*) del sistema operativo.

Los diseñadores de cada sistema operativo han decidido qué repertorio de llamadas al sistema ofrecen y qué semántica asocian. Comparando el repertorio de llamadas al sistema de diferentes sistemas operativos, podemos encontrar semejanzas y diferencias significativas.

Tanto el UNIX como el Win32 ofrecen una llamada al sistema para leer datos de un canal de E/S. En el caso del UNIX, la interfaz es `ssize_t read(int fd, void *buf, size_t count)`; en el caso del Win32, la interfaz es `BOOL ReadFile(HANDLE hFile, LPVOID lpBuffer, DWORD nNumberOfBytesToRead, LPDWORD lpNumberOfBytesRead, LPOVERLAPPED lpOverlapped)`.

En cambio, la llamada al sistema UNIX que permite crear un proceso tiene la interfaz `pid_t fork(void)`; mientras que el equivalente Win32 tiene la interfaz `BOOL CreateProcess(LPCTSTR lpApplicationName, LPTSTR lpCommandLine, LPSECURITY_ATTRIBUTES lpProcessAttributes, LPSECURITY_ATTRIBUTES lpThreadAttributes, BOOL bInheritHandles, DWORD dwCreationFlags, LPVOID lpEnvironment, LPCTSTR lpCurrentDirectory, LPSTARTUPINFO lpStartupInfo, LPPROCESS_INFORMATION lpProcessInformation)`.

3.2. El intérprete de comandos

El intérprete de comandos (o *shell*) es un programa encargado de interpretar y comunicar al SO lo que quiere hacer el usuario del sistema. Este programa reconoce un conjunto limitado de órdenes que, básicamente, permiten al usuario ejecutar programas y acceder, modificar, crear y proteger la información. Se implementa mediante llamadas al sistema.

Ved también

En la figura 1 del apartado 1, el intérprete de comandos correspondería a una de las aplicaciones que encontramos dentro del software de sistema.

Mientras los terminales no disponían de prestaciones gráficas, los intérpretes de comandos trabajaban en modo texto; actualmente, también hay intérpretes de comandos que trabajan en modo gráfico.

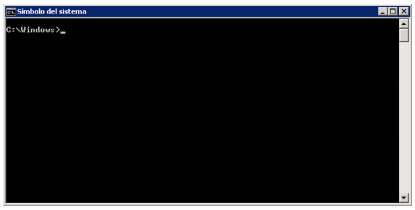


Figura 7. Intérprete de órdenes en modo texto

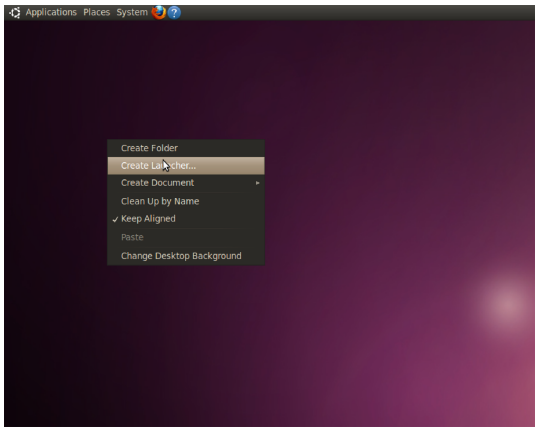


Figura 8. Intérprete de órdenes en modo gráfico

Resumen

Los sistemas operativos han evolucionado durante las últimas décadas, orientados hacia dos objetivos principales:

1) Gestionar de manera eficiente los recursos del sistema computador.

2) Proporcionar un entorno que facilite el desarrollo y la ejecución de programas. Inicialmente, se trabajaba con lenguaje máquina, pero la aparición de lenguajes de alto nivel, compiladores y cargadores han facilitado la tarea de programación.

Por otra parte, la evolución del hardware (dispositivos cada vez más rápidos) determina en cada momento los métodos de trabajo y las técnicas que se desarrollan.

Inicialmente existían los sistemas por lotes (*batch*) y las operaciones fuera de línea (*off-line*) que permitían solapar operaciones de entrada/salida con operaciones de cálculo. Con los sistemas operativos multiprogramados, diversos programas se cargan en la memoria y la CPU hace una ejecución concurrente para intentar conseguir estar siempre ocupada. Finalmente, tenemos el tiempo compartido, que hace que diversos usuarios puedan utilizar un computador interactivamente al mismo tiempo.

Otros sistemas operativos incluyen los conceptos de red, distribución de recursos y tiempo real.

Finalmente, hemos introducido los servicios que suele ofrecer un sistema operativo. Estos servicios se pueden solicitar utilizando llamadas al sistema. Para facilitar el trabajo a los usuarios finales del sistema operativo, dentro del software de sistema encontramos una aplicación, el intérprete de órdenes, que esconderá al usuario final la complejidad inherente a las llamadas al sistema.

Actividades

1. ¿Cuáles son las dos funciones principales de un sistema operativo?

2. ¿Qué es la multiprogramación? ¿Cuál es su ventaja principal?

3. ¿El concepto de multiprogramación tiene sentido en sistemas operativos en tiempo real? ¿Por qué? ¿Y el concepto de tiempo compartido? ¿Por qué?

4. Buscad en Internet más información sobre las diversas versiones del MS-DOS/Windows.

5. Buscad en Internet más información sobre la historia del UNIX y del Linux. Intentad obtener un diagrama en el que aparezcan los sistemas operativos de la familia UNIX.

6. Buscad en Internet el mensaje que Linus Torvalds envió el 25 de agosto de 1991 anunciando que estaba trabajando en un sistema operativo (Linux). Leed las reacciones que provocó.

7. Buscad en Internet información sobre los sistemas operativos utilizados por los computadores Apple.

Glosario

almacenaje en la memoria intermedia *m* Almacenaje que permite solapar el procesamiento de un trabajo con su propia entrada/salida.
en *buffering*

buffering Ved **almacenaje en la memoria intermedia**.

gestión de colas *f* Gestión que permite solapar la entrada/salida de un trabajo con la entrada/salida de otro.
en *spooling*

sistema monousuario *m* Software que permite utilizar y gestionar el ordenador. Comprende el sistema operativo (núcleo o kernel), que se ejecuta en modo de ejecución privilegiado, y las bibliotecas, los editores, el intérprete de órdenes, los compiladores, etc., que se ejecutan en modo de ejecución no privilegiado (usuario).

sistema multiprogramado *m* Sistema que permite que diversos procesos compartan el procesador. Cuando un proceso deja de ejecutarse (al finalizar o bien al establecer una petición de entrada/salida), el procesador pasa a ejecutar otro proceso.

sistema multiusuario *m* Sistema que distingue entre diferentes usuarios (grupos de usuarios) y permite controlar el acceso a la información por parte de los diferentes grupos de usuarios que se puedan definir.

sistema operativo distribuido *m* Sistema que está conectado en red y permite compartir recursos. Un usuario no tiene que saber necesariamente dónde se ejecutan sus procesos o qué recurso concreto de la red está utilizando. El sistema gestiona todos los recursos de manera transparente para el usuario.

sistema operativo en red *m* Sistema que está conectado por una red de interconexión. El usuario sabe en todo momento en qué máquina de la red se ejecutan sus procesos.

sistema operativo en tiempo real *m* Sistema operativo que tiene que ser capaz de reaccionar en un tiempo máximo determinado ante acontecimientos externos.

sistema por lotes *m* Manera de utilizar un computador en el que la ejecución de los trabajos no es interactiva. La ejecución de los trabajos suele ser secuencial.

software de sistema *m* Software que permite utilizar y gestionar el computador. Comprende el sistema operativo (núcleo o *kernel*, que se ejecuta en modo de ejecución privilegiado, y las bibliotecas, los editores, el intérprete de órdenes y los compiladores, entre otros que se ejecutan en modo de ejecución no privilegiado (usuario).

spooling Ved **gestión de colas**.

tiempo compartido *m* Sistema en que diversos procesos pueden compartir el procesador. En este caso, cada proceso tiene asignado el procesador por un tiempo máximo, un *quantum*. El procesador se asigna a otro proceso cuando el proceso actualmente en ejecución llega al final de su ejecución, realiza una operación de entrada/salida o agota su *quantum*.

trabajo fuera de línea *m* Trabajo que se efectúa en un dispositivo fuera de línea. Un dispositivo está fuera de línea si está desconectado del computador y, por lo tanto, no puede llevar a cabo ningún tipo de transferencia directa.

Bibliografía

Silberschatz, A.; Galvin; Gagne, G. (2008). *Operating Systems Concepts* (8.ª ed.). John Wiley & Sons.

Tanembaum, A. (2009). *Modern Operating Systems*. Prentice Hall.