

SISTEMAS OPERATIVOS 2.

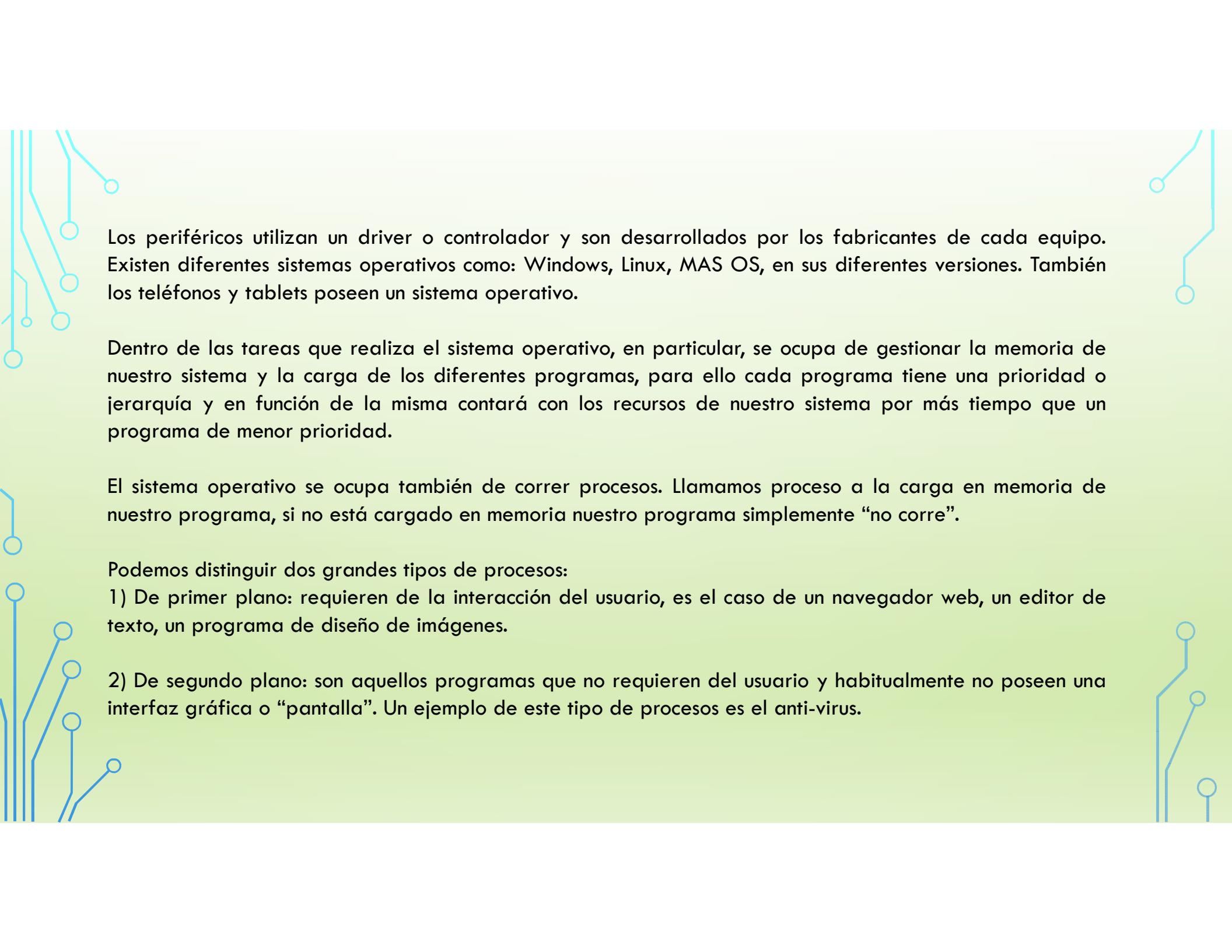
LABORATORIO.

CONCEPTO Y OBJETIVOS DE LOS SISTEMAS OPERATIVOS

Concepto:

Un sistema operativo es un conjunto de programas que permite manejar la memoria, disco, medios de almacenamiento de información y los diferentes periféricos o recursos de nuestra computadora, como son el teclado, el mouse, la impresora, tarjeta de red, entre otros.





Los periféricos utilizan un driver o controlador y son desarrollados por los fabricantes de cada equipo. Existen diferentes sistemas operativos como: Windows, Linux, MAS OS, en sus diferentes versiones. También los teléfonos y tablets poseen un sistema operativo.

Dentro de las tareas que realiza el sistema operativo, en particular, se ocupa de gestionar la memoria de nuestro sistema y la carga de los diferentes programas, para ello cada programa tiene una prioridad o jerarquía y en función de la misma contará con los recursos de nuestro sistema por más tiempo que un programa de menor prioridad.

El sistema operativo se ocupa también de correr procesos. Llamamos proceso a la carga en memoria de nuestro programa, si no está cargado en memoria nuestro programa simplemente “no corre”.

Podemos distinguir dos grandes tipos de procesos:

- 1) De primer plano: requieren de la interacción del usuario, es el caso de un navegador web, un editor de texto, un programa de diseño de imágenes.
- 2) De segundo plano: son aquellos programas que no requieren del usuario y habitualmente no poseen una interfaz gráfica o “pantalla”. Un ejemplo de este tipo de procesos es el anti-virus.

Los principales objetivos de los sistemas operativos son:

Abstraer al usuario de la complejidad del hardware: El sistema operativo hace que la computadora sea más fácil de utilizar.

Eficiencia: Permite que los recursos de la computadora se utilicen de la forma más eficiente posible. Por ejemplo, se deben optimizar los accesos a disco para acelerar las operaciones de entrada y salida.

Permitir la ejecución de programas: Cuando un usuario quiere ejecutar un programa, el sistema operativo realiza todas las tareas necesarias para ello, tales como cargar las instrucciones y datos del programa en memoria, iniciar dispositivos de entrada/salida y preparar otros recursos.

Acceder a los dispositivos entrada/salida: El sistema operativo suministra una interfaz homogénea para los dispositivos de entrada/salida para que el usuario pueda utilizar de forma más sencilla los mismos.

Proporcionar una estructura y conjunto de operaciones para el sistema de archivos.

Controlar el acceso al sistema y los recursos: en el caso de sistemas compartidos, proporcionando protección a los recursos y los datos frente a usuarios no autorizados.

Detección y respuesta ante errores: El sistema operativo debe prever todas las posibles situaciones críticas y resolverlas, si es que se producen.

Capacidad de adaptación: Un sistema operativo debe ser construido de manera que pueda evolucionar a la vez que surgen actualizaciones hardware y software.

Gestionar las comunicaciones en red: El sistema operativo debe permitir al usuario manejar con facilidad todo lo referente a la instalación y uso de las redes de computadoras.

Permitir a los usuarios compartir recursos y datos: Este aspecto está muy relacionado con el anterior y daría al sistema operativo el papel de gestor de los recursos de una red.

Evolución histórica de los sistemas operativos.

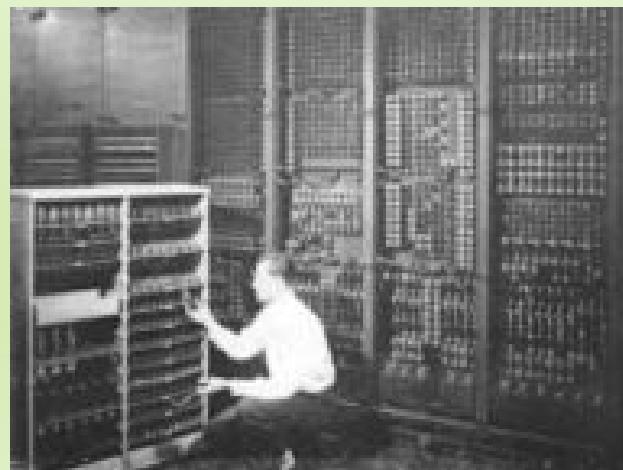
El hardware y el software de las computadoras han evolucionado de forma paralela y conjunta en las últimas décadas. Por lo que la evolución que vamos a ver de los sistemas operativos está estrechamente relacionada con los avances en la arquitectura de las computadoras que se produjo de cada generación.



Primera generación (1945-1955)

Las primeras computadoras estaban construidos con tubos de vacío. En un principio no existían sistemas operativos, se programaba directamente sobre el hardware. Los programas estaban hechos directamente en código máquina y el control de las funciones básicas se realiza mediante paneles enchufables.

Hacia finales de 1950 aparecen las tarjetas perforadas que sustituyen los paneles enchufables. Las tarjetas perforadas supusieron un enorme paso ya que permitían codificar instrucciones de un programa y los datos en una cartulina con puntos que podía interpretar la computadora. La mayoría de los programas usaban rutinas de E/S y un programa cargador (automatizaba la carga de programas ejecutables en la máquina) esto constituía una forma rudimentaria de sistema operativo.



2^a Generación (1955-1965)

Esta generación se caracteriza por la aparición de los transistores que permitieron la construcción de computadoras más pequeños y potentes. La programación se realizaba en lenguaje ensamblador y en FORTRAN sobre tarjetas perforadas. Otro aspecto importante de esta generación es el procesamiento por lotes, en el cual mientras el sistema operativo está ejecutando un proceso, éste último dispone de todos los recursos hasta su finalización. La preparación de los trabajos se realiza a través de un lenguaje de control de trabajos conocido como JCL. El sistema operativo residía en memoria y tenía un programa de control que interpretaba las tarjetas de control, escritas JCL. Dependiendo del contenido de la tarjeta de control el sistema operativo realizaba una acción determinada. Este programa de control es un antecedente de los modernos intérpretes de órdenes.

Procesamiento Fuera de línea (Offline)

Como mejora del procesamiento por lotes surgió el procesamiento fuera de línea (off-line), en el cual las operaciones de carga de datos y salida de resultados de un proceso podían realizarse de forma externa y sin afectar al tiempo que el procesador dedicaba a los procesos. A esto ayudó la aparición de las cintas magnéticas y las impresoras de líneas. Ejemplos de sistemas operativos de la época son FMS (Fortran Monitor System) e IBSYS.

3^a Generación (1965-1980)

La aparición de los circuitos integrados (CI) supuso una mejora consiguiendo un menor tamaño y relación precio/rendimiento respecto de las máquinas de generaciones anteriores. En relación con los sistemas operativos, la característica principal de esta generación fue el desarrollo de la multiprogramación y los sistemas compartidos. En los sistemas multiprogramados se cargan varios programas en memoria simultáneamente y se alterna su ejecución. Esto maximiza la utilización del procesador. Como evolución del sistema operativo aparecen los sistemas de tiempo compartido donde el tiempo del procesador se comparte entre programas de varios usuarios pudiendo ser programas interactivos. Algunos de los sistemas operativos de esta generación son OS/360, CTSS, MULTICS y UNIX.

4^a Generación (1980-hasta hoy)

En esta generación se producen grandes avances en la industria hardware como la creación de los circuitos LSI (integrados a gran escala). También aparecen las computadoras personales, entre finales de la anterior generación y principios de la presente. Ejemplos de sistemas operativos de las primeras computadoras personales son MS-DOS, desarrollado por Microsoft, Inc., para el IBM PC y MacOS de Apple Computer, Inc. Steve Jobs, cofundador de Apple, apostó por la primera interfaz gráfica basada en ventanas, iconos, menús y ratón a partir de una investigación realizada por Xerox. Siguiendo esta filosofía aparecería MS Windows. Durante los 90 apareció Linux a partir del núcleo desarrollado por Linus Torvalds. Los sistemas operativos evolucionan hacia sistemas interactivos con una interfaz cada vez más amigable al usuario. Los sistemas Windows han ido evolucionando, con diferentes versiones tanto para escritorio como para servidor (Windows 3.x, 98, 2000, XP, Vista, 7, Windows Server 2003, 2008, etc), al igual que lo han hecho Linux (con multitud de distribuciones, Ubuntu, Debian, RedHat, Mandrake, etc) y los sistemas Mac (Mac OS 8, OS 9, OS X, Mac OS X 10.6 "Snow Leopard", entre otros).

Un avance importante fue el desarrollo de redes de computadoras a mediados de los años 80 que ejecutan sistemas operativos en red y sistemas operativos distribuidos. En un sistema operativo en red los usuarios tienen conocimiento de la existencia de múltiples computadoras y pueden acceder a máquinas remotas y copiar archivos de una computadora a otra. En un sistema distribuido los usuarios no saben donde se están ejecutando sus programas o dónde están ubicados sus programas, ya que los recursos de procesamiento, memoria y datos están distribuidos entre las computadoras de la red, pero todo esto es transparente al usuario.

Actualmente, existen sistemas operativos integrados, para una gran diversidad de dispositivos electrónicos, tales como, teléfonos móviles, PDAs (Personal Digital Assistant, Asistente Digital Personal o computadora de bolsillo), otros dispositivos de comunicaciones e informática y electrodomésticos. Ejemplos de este tipo de sistemas operativos son PalmOS, WindowsCE, Android OS, etc. Haremos una referencia especial al último, Android OS, se trata de un sistema operativo basado en Linux. Fue diseñado en un principio para dispositivos móviles, tales como teléfonos inteligentes y tablets, pero actualmente se encuentra en desarrollo para su aplicación también en netbooks y PCs.



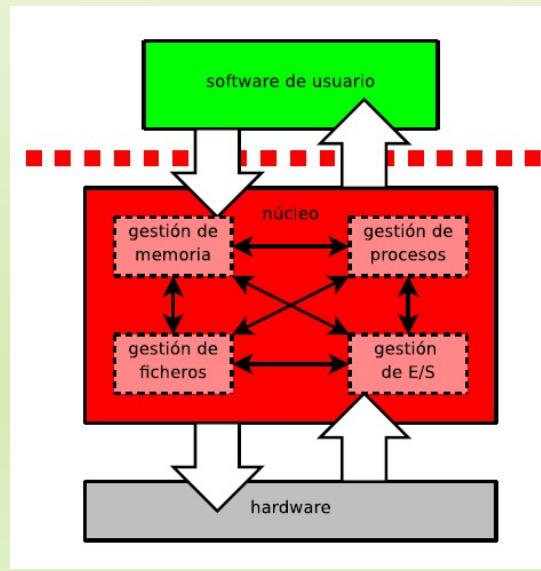
TIPOS DE SISTEMAS OPERATIVOS.

Ahora vamos a clasificar los sistemas operativos en base a su estructura, servicios que suministran y por su forma.

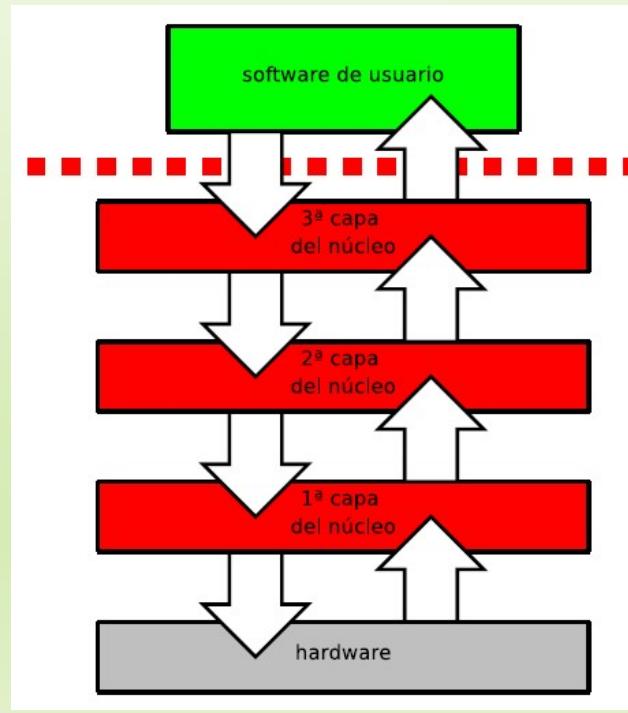
Por estructura	Por sus servicios	Por su forma
Monolíticos	Monousuario	Sistema operativo en red
Jerárquicos	Multiusuario	
Máquina Virtual	Mono tarea	
Microkernel o Cliente-Servidor	Multitarea	Sistema operativo distribuido
Monolíticos	Monoprocesador	
	Multiprocesador	

SISTEMAS OPERATIVOS POR SU ESTRUCTURA

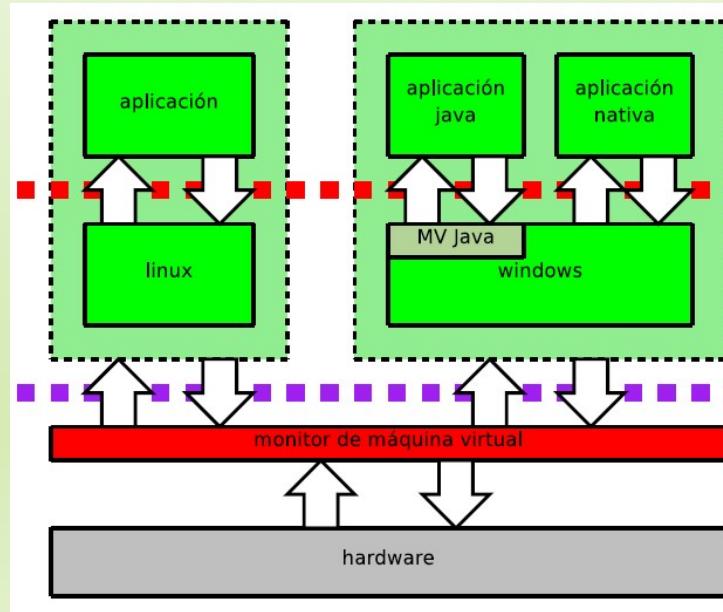
- **Monolíticos:** Es la estructura de los primeros sistemas operativos, consistía en un solo programa desarrollado con rutinas entrelazadas que podían llamarse entre sí. Por lo general, eran sistemas operativos hechos a medida, pero difíciles de mantener



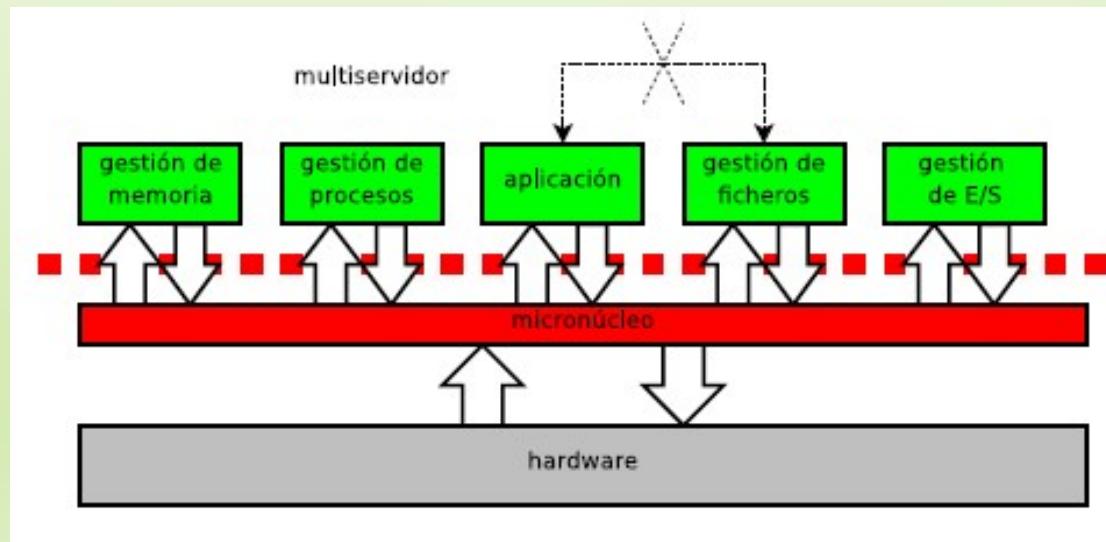
Jerárquicos: Conforme las necesidades de los usuarios aumentaron, los sistemas operativos fueron creciendo en complejidad y funciones. Esto llevó a que se hiciera necesaria una mayor organización del software del sistema operativo, dividiéndose en partes más pequeñas, diferenciadas por funciones y con una interfaz clara para interoperar con los demás elementos. Un ejemplo de este tipo de sistemas operativos fue MULTICS.



Máquina Virtual: El objetivo de los sistemas operativos es el de integrar distintos sistemas operativos dando la sensación de ser varias máquinas diferentes. Presentan una interfaz a cada proceso, mostrando una máquina que parece idéntica a la máquina real subyacente. Estas máquinas no son máquinas extendidas, son una réplica de la máquina real, de manera que en cada una de ellas se pueda ejecutar un sistema operativo diferente, que será el que ofrezca la máquina extendida al usuario. VMware y VM/CMS son ejemplos de este tipo de sistemas operativos.



Microkernel o Cliente-Servidor: El modelo del núcleo de estos sistemas operativos distribuye las diferentes tareas en porciones de código modulares y sencillas. El objetivo es aislar del sistema, su núcleo, las operaciones de entrada/salida, gestión de memoria, del sistema de archivos, etc. Esto incrementa la tolerancia a fallos, la seguridad y la portabilidad entre plataformas de hardware. Algunos ejemplos son MAC OS X o AIX.



SISTEMAS OPERATIVOS POR SUS SERVICIOS

- **Monousuario:** Son aquellos que soportan a un usuario a la vez, sin importar el número de procesos o tareas que el usuario pueda ejecutar en un mismo instante de tiempo. Ejemplos de sistemas operativos de este tipo son MS-DOS, Microsoft Windows 9x y ME, MAC OS, entre otros.
- **Multiusuario:** Son capaces de dar servicio a más de un usuario a la vez, ya sea por medio de varios terminales conectadas a la computadora o por medio de sesiones remotas en una red de comunicaciones. No importa el número de procesadores en la máquina ni el número de procesos que puede ejecutar cada usuario simultáneamente. Algunos ejemplos serán UNIX, GNU/Linux, Microsoft Windows Server o MAC OS X.

Monotarea: Sólo permiten una tarea a la vez por usuario. Se puede dar el caso de un sistema multiusuario y monotarea, en el cual se admiten varios usuarios simultáneamente pero cada uno de ellos puede ejecutar sólo una tarea en un instante dado. Ejemplos de sistemas monotarea son MS-DOS, Microsoft Windows 3.x y 95 (estos últimos sólo simulan la multitarea).

Multitarea: Permite al usuario realizar varias tareas al mismo tiempo. Algunos ejemplos son MAC OS, UNIX, Linux, Microsoft Windows 98, 2000, XP, Vista y 7.

Monoprocesador: Es aquel capaz de manejar sólo un procesador, de manera que si la computadora tuviese más de uno le sería inútil. MS-DOS y MAC OS son ejemplos de este tipo de sistemas operativos.

Multiprocesador: Un sistema operativo multiprocesador se refiere al número de procesadores del sistema, éste es más de uno y el sistema operativo es capaz de utilizarlos todos para distribuir su carga de trabajo. Estos sistemas trabajan de dos formas: simétricamente (los procesos son enviados indistintamente a cualquiera de los procesadores disponibles) y asimétricamente (uno de los procesadores actúa como maestro o servidor y distribuye la carga de procesos a los demás).

SISTEMAS OPERATIVOS POR SU FORMA

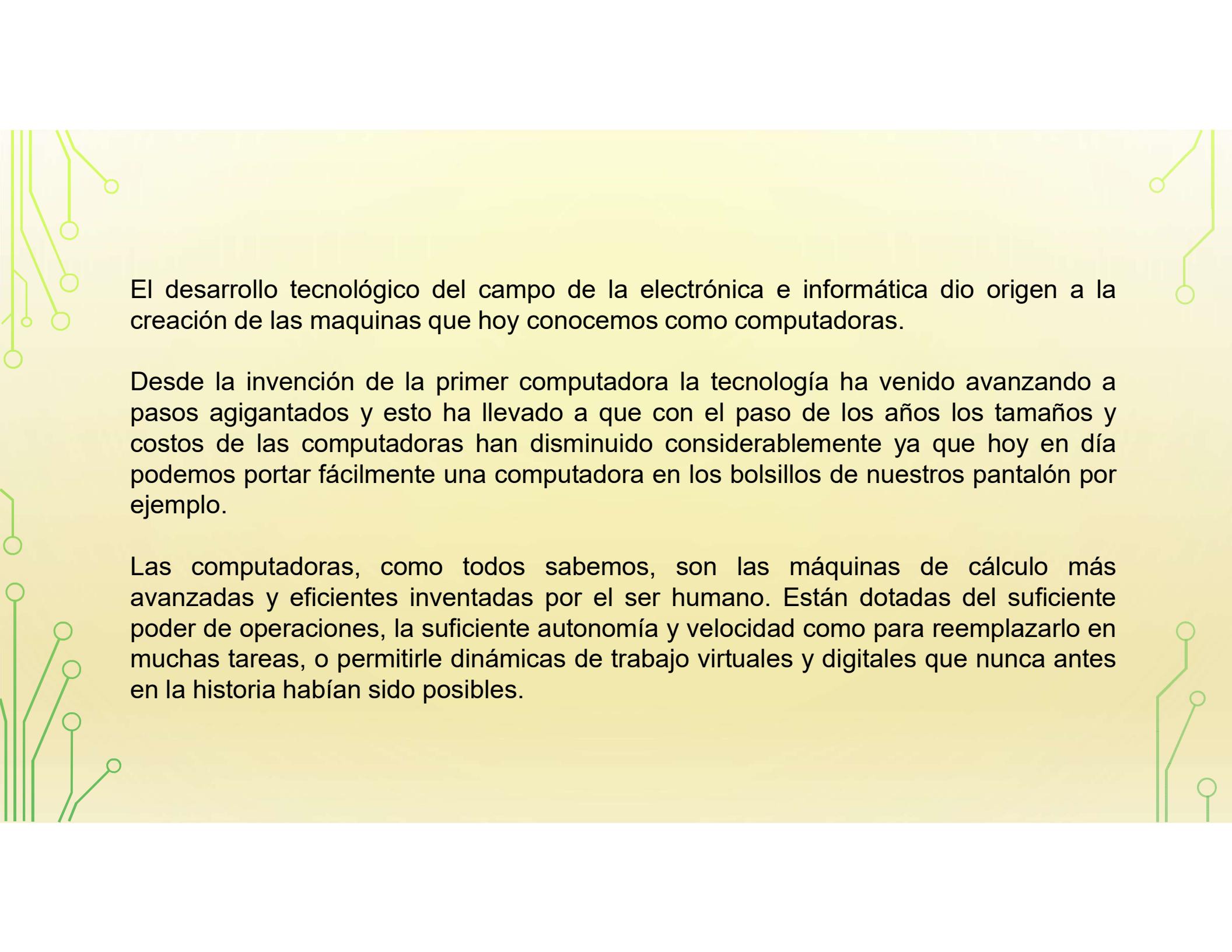
- **Sistemas operativos en red:** Estos sistemas tienen la capacidad de interactuar con los sistemas operativos de otras máquinas a través de la red, con el objeto de intercambiar información, transferir archivos, etc. La clave de estos sistemas es que el usuario debe conocer la ubicación de los recursos en red a los que deseé acceder. Los sistemas operativos modernos más comunes pueden considerarse sistemas en red, por ejemplo: Novell, Windows Server, Linux, etc.
- **Sistemas operativos distribuidos:** Abarcan los servicios de red, las funciones se distribuyen entre diferentes computadoras, logrando integrar recursos (impresoras, unidades de respaldo, memoria, procesos, etc.) en una sola máquina virtual que es a la que el usuario accede de forma transparente. En este caso, el usuario no necesita saber la ubicación de los recursos, sino que los referencia por su nombre y los utiliza como si fueran locales a su lugar de trabajo habitual. MOSIX es un ejemplo de estos sistemas operativos.

SISTEMAS OPERATIVOS 2

LABORATORIO

REVISION DEL HARDWARE DE UNA COMPUTADORA.

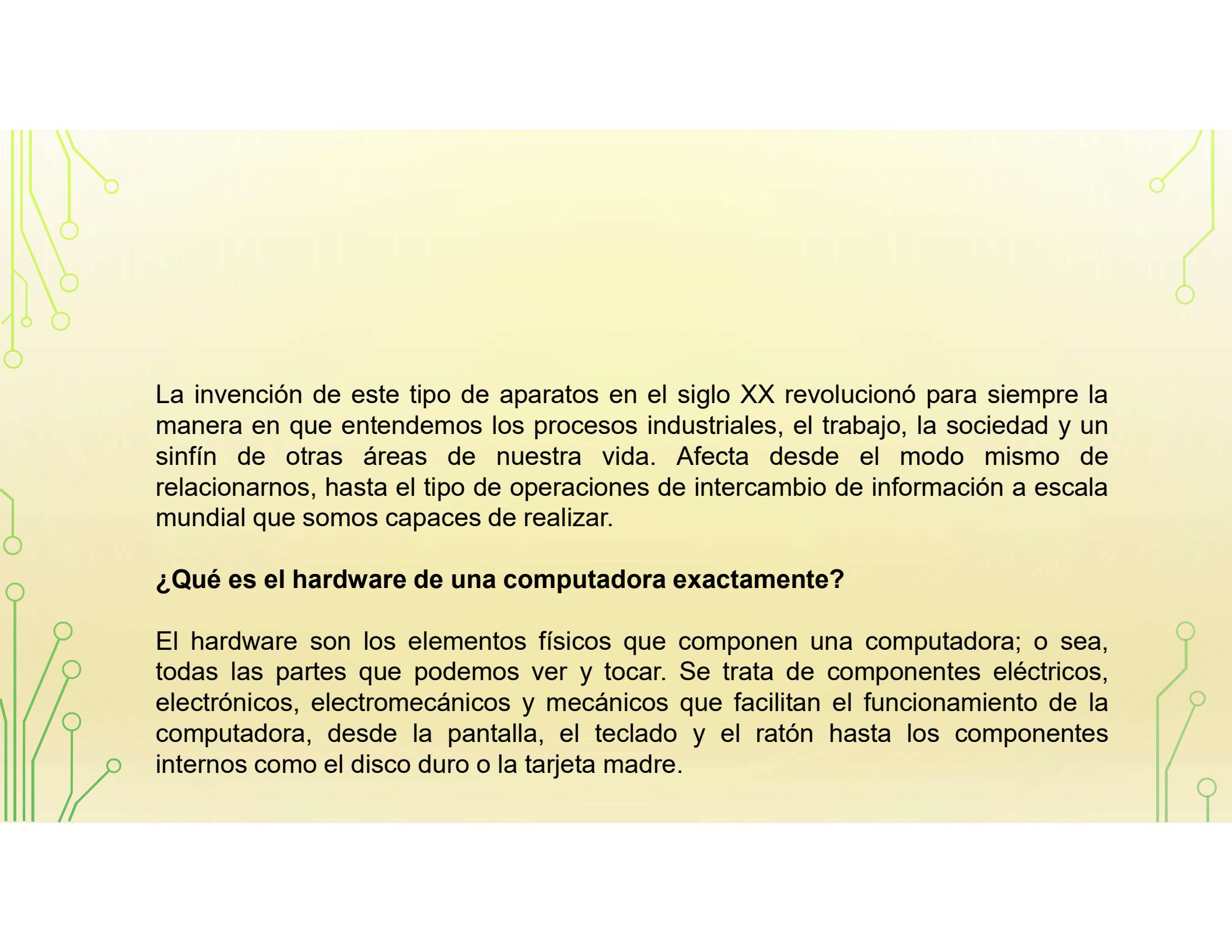




El desarrollo tecnológico del campo de la electrónica e informática dio origen a la creación de las maquinas que hoy conocemos como computadoras.

Desde la invención de la primer computadora la tecnología ha venido avanzando a pasos agigantados y esto ha llevado a que con el paso de los años los tamaños y costos de las computadoras han disminuido considerablemente ya que hoy en día podemos portar fácilmente una computadora en los bolsillos de nuestros pantalón por ejemplo.

Las computadoras, como todos sabemos, son las máquinas de cálculo más avanzadas y eficientes inventadas por el ser humano. Están dotadas del suficiente poder de operaciones, la suficiente autonomía y velocidad como para reemplazarlo en muchas tareas, o permitirle dinámicas de trabajo virtuales y digitales que nunca antes en la historia habían sido posibles.



La invención de este tipo de aparatos en el siglo XX revolucionó para siempre la manera en que entendemos los procesos industriales, el trabajo, la sociedad y un sinfín de otras áreas de nuestra vida. Afecta desde el modo mismo de relacionarnos, hasta el tipo de operaciones de intercambio de información a escala mundial que somos capaces de realizar.

¿Qué es el hardware de una computadora exactamente?

El hardware son los elementos físicos que componen una computadora; o sea, todas las partes que podemos ver y tocar. Se trata de componentes eléctricos, electrónicos, electromecánicos y mecánicos que facilitan el funcionamiento de la computadora, desde la pantalla, el teclado y el ratón hasta los componentes internos como el disco duro o la tarjeta madre.

Hardware y software de una computadora: ¿Cuál es la principal diferencia?

Para que una computadora funcione correctamente necesita tanto del hardware como del software. El software es la parte digital de la computadora, el conjunto de instrucciones, programas y reglas informáticas que el equipo necesita para funcionar.

Básicamente, el software se encarga de realizar las operaciones mientras el hardware garantiza el soporte físico mediante el cual se llevan a cabo esas funciones. Por tanto, el hardware y el software de una computadora se complementan, de manera que **sin uno de ellos** el equipo no podría funcionar.

La principal diferencia entre el hardware y el software de una computadora radica en que el hardware está compuesto por elementos materiales que se pueden ver y tocar mientras que el software no tiene una forma física, sino que responde a elementos digitales intangibles, como es el caso del sistema operativo, los procesadores de texto, los reproductores de vídeo o los programas de edición de imágenes, etc.

¿Cuáles son los principales componentes del hardware de una computadora?

Los componentes hardware de una computadora se pueden dividir en interno y periféricos. El hardware interno es aquel que se encuentra dentro de la computadora, como los circuitos, la unidad central de procesamiento, la memoria RAM, la tarjeta de video y el disco duro. En cambio, el hardware periférico se encuentra fuera de la computadora, pero mantiene una comunicación constante con este, como por ejemplo el ratón, el teclado, el monitor, la impresora o la memoria USB.

También es habitual hacer referencia al hardware básico de una computadora, que se refiere a los elementos imprescindibles para el funcionamiento del equipo, como la tarjeta madre, la memoria RAM o el monitor. En cambio, el hardware complementario de la computadora es aquel que solo añade funciones adicionales, como las impresoras o las memorias USB.

Una clasificación más exhaustiva de las partes del hardware de una computadora es:

Procesamiento. Son los elementos del hardware de una computadora que conforman el núcleo duro del sistema cuya función principal consiste en procesar la información e interpretar y ejecutar las instrucciones, de manera que influyen en la capacidad mecánica del equipo para realizar las operaciones necesarias. Tal es el caso del CPU.

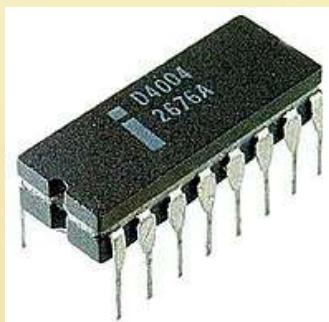
Almacenamiento. Son los componentes hardware de una computadora que permiten guardar y recuperar la información, ya se trate de soportes internos o portátiles. La memoria RAM, por ejemplo, se encarga del almacenamiento temporal de la información mientras que el disco duro almacena los datos de forma permanente.

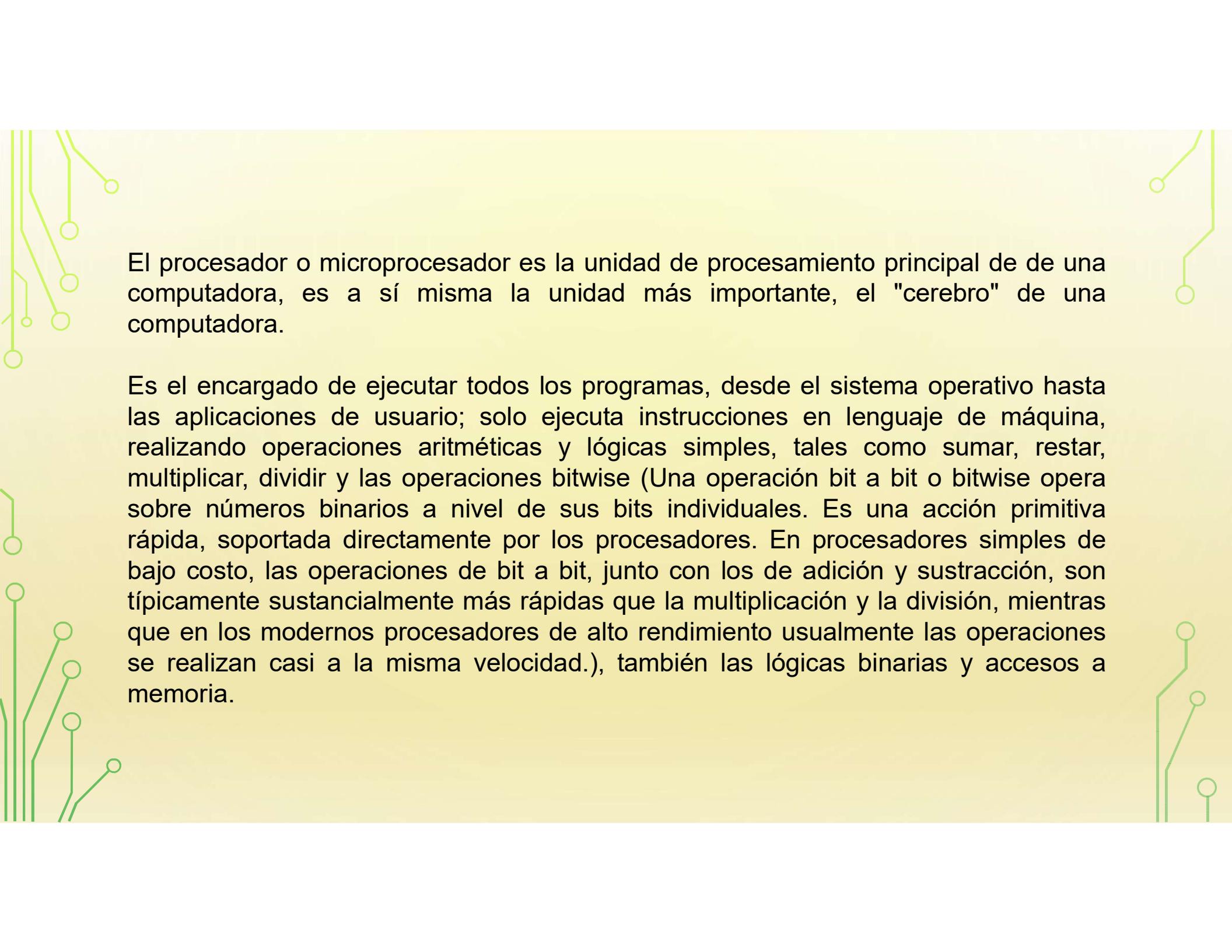
Periféricos de entrada. Estos dispositivos, que pueden estar integrados en la computadora o ser removibles, permiten introducir la información para que el equipo se conecte con el exterior. Es el caso del teclado, el ratón, la webcam, el micrófono o el lector de CD/DVD.

Periféricos de salida. Estos dispositivos, que también pueden estar integrados o ser removibles, tienen la función de extraer o recuperar la información de la computadora transfiriendo los datos al exterior, como por ejemplo: La pantalla, impresoras y bocinas son ejemplos de hardware de salida.

Periféricos mixtos. Estos elementos del hardware de una computadora combinan indistintamente la entrada y salida de información del sistema, como es el caso de las tarjetas de red, el módem o las memorias USB.

PROCESADORES





El procesador o microprocesador es la unidad de procesamiento principal de una computadora, es a sí misma la unidad más importante, el "cerebro" de una computadora.

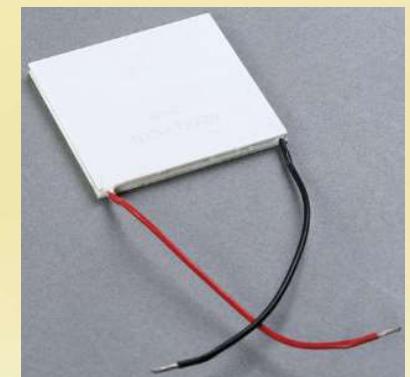
Es el encargado de ejecutar todos los programas, desde el sistema operativo hasta las aplicaciones de usuario; solo ejecuta instrucciones en lenguaje de máquina, realizando operaciones aritméticas y lógicas simples, tales como sumar, restar, multiplicar, dividir y las operaciones bitwise (Una operación bit a bit o bitwise opera sobre números binarios a nivel de sus bits individuales. Es una acción primitiva rápida, soportada directamente por los procesadores. En procesadores simples de bajo costo, las operaciones de bit a bit, junto con los de adición y sustracción, son típicamente sustancialmente más rápidas que la multiplicación y la división, mientras que en los modernos procesadores de alto rendimiento usualmente las operaciones se realizan casi a la misma velocidad.), también las lógicas binarias y accesos a memoria.

Puede contener una o más unidades centrales de procesamiento (CPU) constituidas, esencialmente, por registros, una unidad de control, una unidad aritmético lógica (ALU) y una unidad de cálculo en coma flotante (FPU) (conocida antiguamente como "coprocesador matemático").

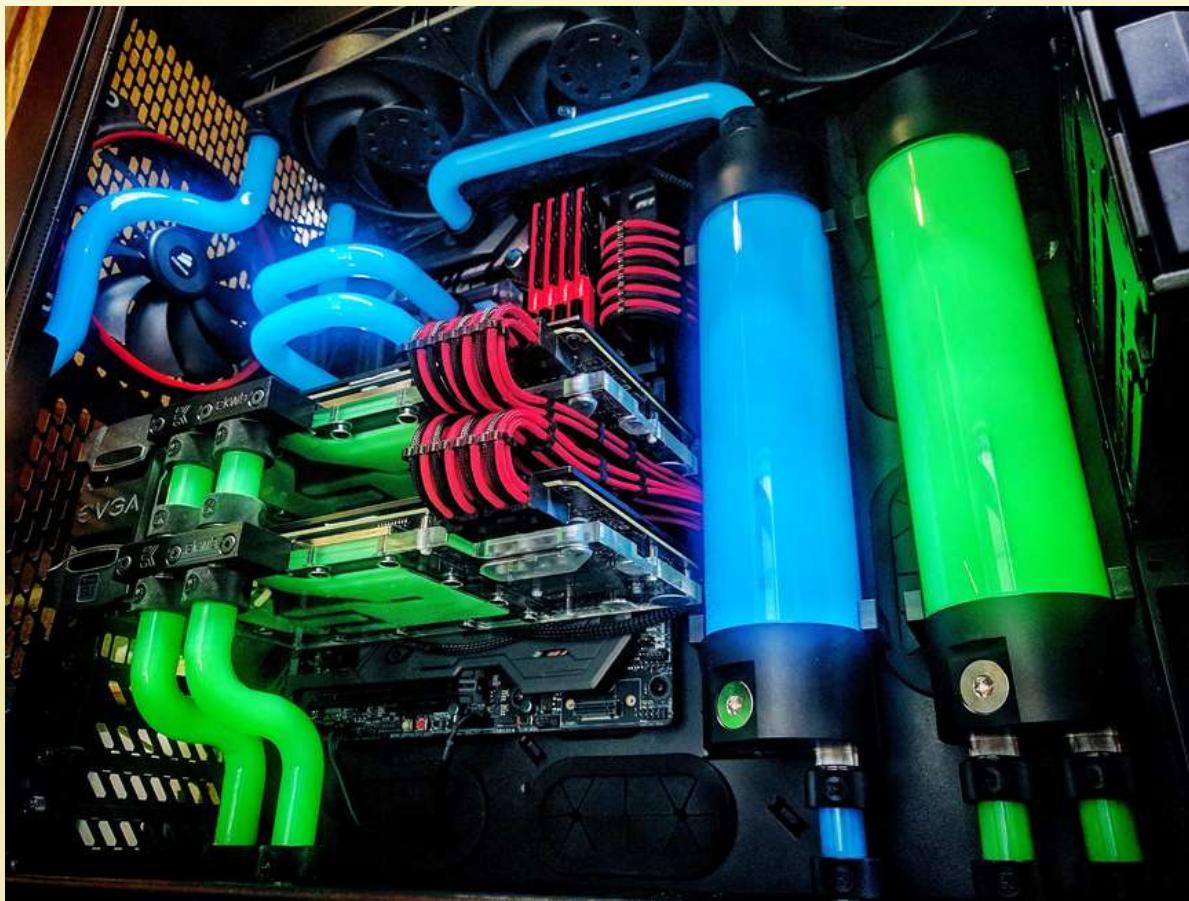


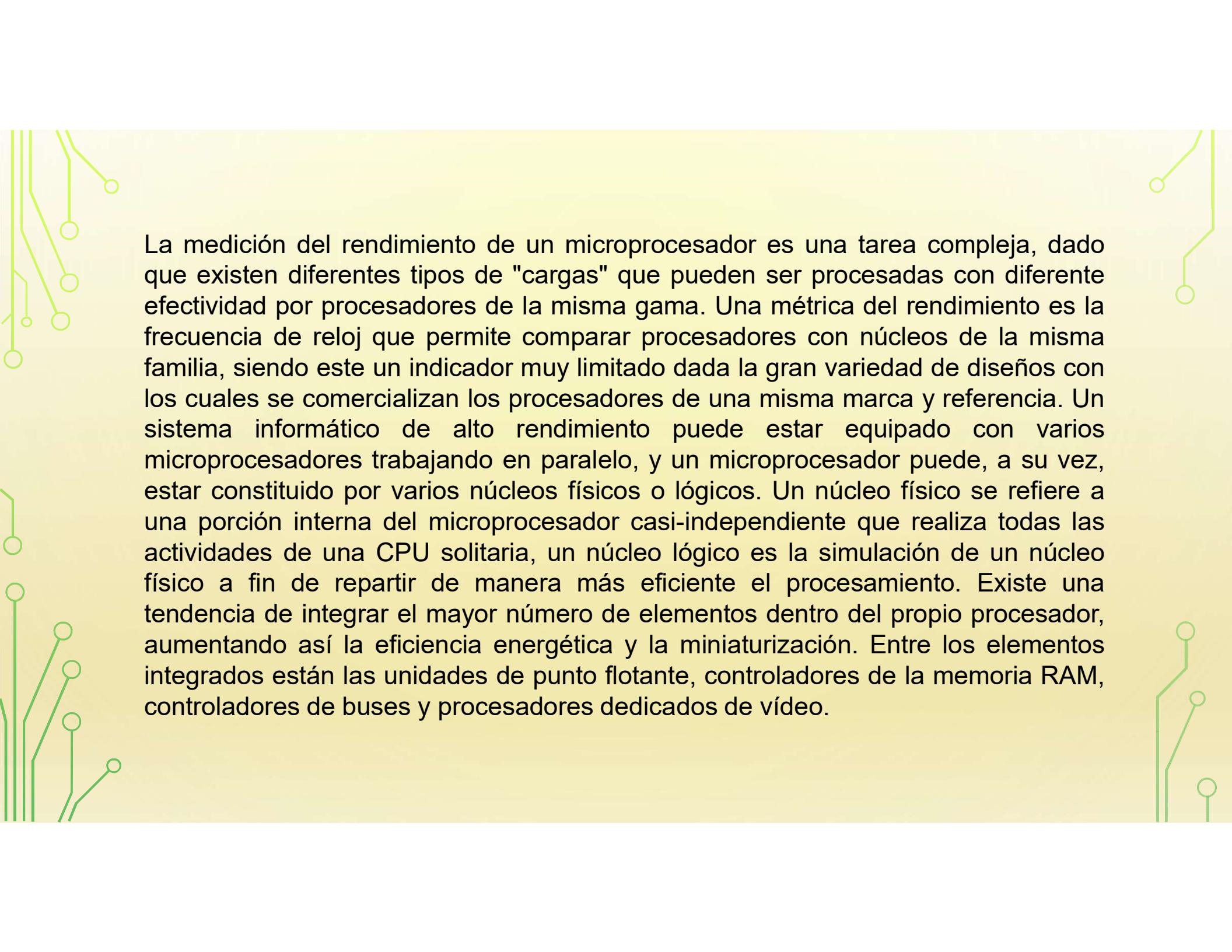
El microprocesador está conectado generalmente mediante un zócalo específico de la placa base de la computadora; normalmente para su correcto y estable funcionamiento, se le incorpora un sistema de refrigeración que consta de un disipador de calor, fabricado de algún material de alta conductividad térmica, como cobre o aluminio, y de uno o más ventiladores que eliminan el exceso del calor absorbido por el disipador. Entre el disipador y la cápsula del microprocesador usualmente se coloca pasta térmica para mejorar la conductividad del calor. Existen otros métodos más eficaces, como la refrigeración líquida o el uso de celda **peltier** para refrigeración extrema, pero estas técnicas se utilizan casi exclusivamente para aplicaciones especiales, tales como en las prácticas de overclocking.

El **efecto Peltier** hace referencia a la creación de una diferencia de temperatura debida a un voltaje eléctrico. Sucede cuando una corriente se hace pasar por dos metales o semiconductores conectados por dos “junturas de Peltier”. La corriente propicia una transferencia de calor de una juntura a la otra: una se enfriá en tanto que otra se calienta.



Refrigeración Liquida



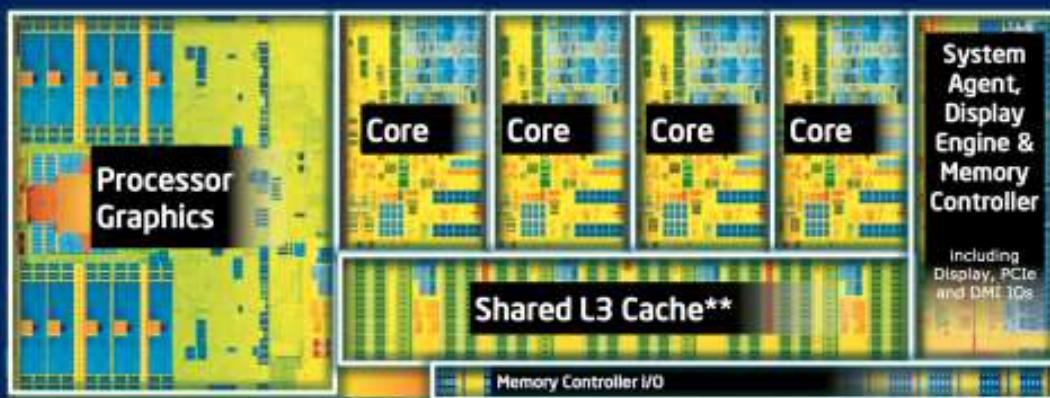


La medición del rendimiento de un microprocesador es una tarea compleja, dado que existen diferentes tipos de "cargas" que pueden ser procesadas con diferente efectividad por procesadores de la misma gama. Una métrica del rendimiento es la frecuencia de reloj que permite comparar procesadores con núcleos de la misma familia, siendo este un indicador muy limitado dada la gran variedad de diseños con los cuales se comercializan los procesadores de una misma marca y referencia. Un sistema informático de alto rendimiento puede estar equipado con varios microprocesadores trabajando en paralelo, y un microprocesador puede, a su vez, estar constituido por varios núcleos físicos o lógicos. Un núcleo físico se refiere a una porción interna del microprocesador casi-independiente que realiza todas las actividades de una CPU solitaria, un núcleo lógico es la simulación de un núcleo físico a fin de repartir de manera más eficiente el procesamiento. Existe una tendencia de integrar el mayor número de elementos dentro del propio procesador, aumentando así la eficiencia energética y la miniaturización. Entre los elementos integrados están las unidades de punto flotante, controladores de la memoria RAM, controladores de buses y procesadores dedicados de vídeo.

ARQUITECTURA VON NEUMANN



4th Generation Intel® Core™ Processor Die Map
22nm Tri-Gate 3-D Transistors

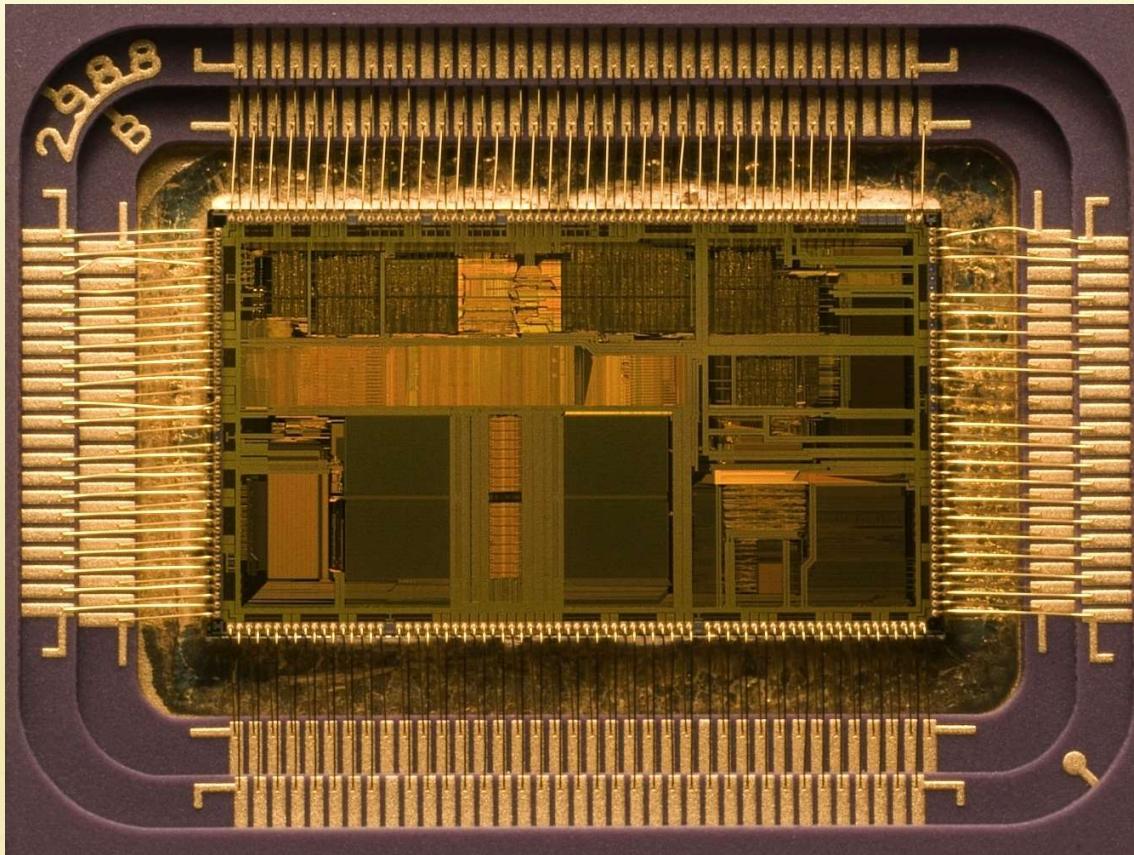


Quad core die shown above

Transistor count: 1.4 Billion

Die size: 177mm²

Empaqueado de un procesador Intel 80486 en un empaque de cerámica



Procesador desarrollado con tecnología de 5 nm y 16.000 millones de transistores, superando todos los récords. Pero hay que tener claro que se trata de un procesador de bajo consumo para portátiles ultra finos y mini PCs



Núcleos (Cores).

Una de las cosas más sonadas en cuanto a CPU's son los núcleos ('cores' en inglés). Los cores son, como un subprocesador en sí mismo. Antes, los procesadores eran de un solo núcleo (single core), por lo que no podían realizar más que una tarea al mismo tiempo.

¿Qué es un hilo?

Un hilo es una unidad básica de utilización de CPU, la cual contiene un id de hilo, su propio program counter, un conjunto de registros, y una pila; que se representa a nivel del sistema operativo con una estructura llamada TCB (thread control block). Los hilos comparten con otros hilos que pertenecen al mismo proceso la sección de código, la sección de datos, entre otras cosas. Si un proceso tiene múltiples hilos, puede realizar más de una tarea a la vez (esto es real cuando se posee más de un CPU).

SISTEMAS OPERATIVOS 2

LABORATORIO

MEMORIA

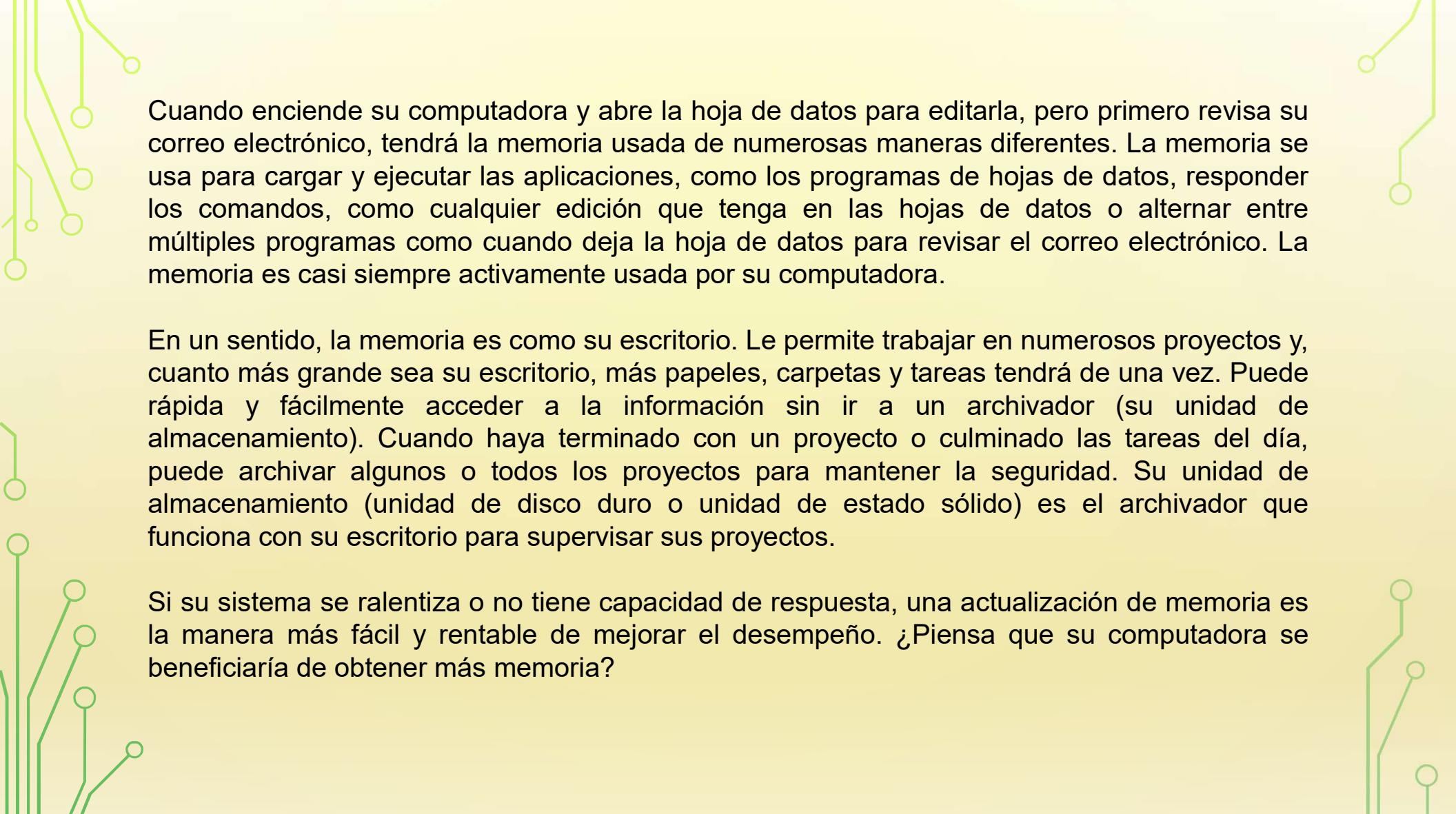


¿Qué es la memoria de la computadora (RAM)?

La memoria de computadora o la memoria de acceso aleatorio (RAM) es su almacenamiento de datos a corto plazo del sistema. Almacena la información que usa de forma activa su computadora para que pueda acceder a ella de manera rápida. Cuanto más programas ejecute su sistema, más memoria necesitará.

¿Para qué se usa la RAM?

La memoria RAM le permite que su computadora desempeñe muchas tareas diarias como cargar aplicaciones, buscar en Internet, editar una hoja de datos o experimentar un juego más reciente. La memoria también le permite intercambiar rápidamente entre estas tareas. Recuerda dónde está en una tarea cuando intercambia con otra tarea. Como regla, cuanto más memoria tenga, mejor.



Cuando enciende su computadora y abre la hoja de datos para editarla, pero primero revisa su correo electrónico, tendrá la memoria usada de numerosas maneras diferentes. La memoria se usa para cargar y ejecutar las aplicaciones, como los programas de hojas de datos, responder los comandos, como cualquier edición que tenga en las hojas de datos o alternar entre múltiples programas como cuando deja la hoja de datos para revisar el correo electrónico. La memoria es casi siempre activamente usada por su computadora.

En un sentido, la memoria es como su escritorio. Le permite trabajar en numerosos proyectos y, cuanto más grande sea su escritorio, más papeles, carpetas y tareas tendrá de una vez. Puede rápida y fácilmente acceder a la información sin ir a un archivador (su unidad de almacenamiento). Cuando haya terminado con un proyecto o culminado las tareas del día, puede archivar algunos o todos los proyectos para mantener la seguridad. Su unidad de almacenamiento (unidad de disco duro o unidad de estado sólido) es el archivador que funciona con su escritorio para supervisar sus proyectos.

Si su sistema se ralentiza o no tiene capacidad de respuesta, una actualización de memoria es la manera más fácil y rentable de mejorar el desempeño. ¿Piensa que su computadora se beneficiaría de obtener más memoria?

Jerarquía de memoria

Los componentes fundamentales de las computadoras de propósito general son la CPU, el espacio de almacenamiento y los dispositivos de entrada/salida. La habilidad para almacenar las instrucciones que forman un programa de computadora y la información que manipulan las instrucciones es lo que hace versátiles a las computadoras diseñadas según la arquitectura de programas almacenados.

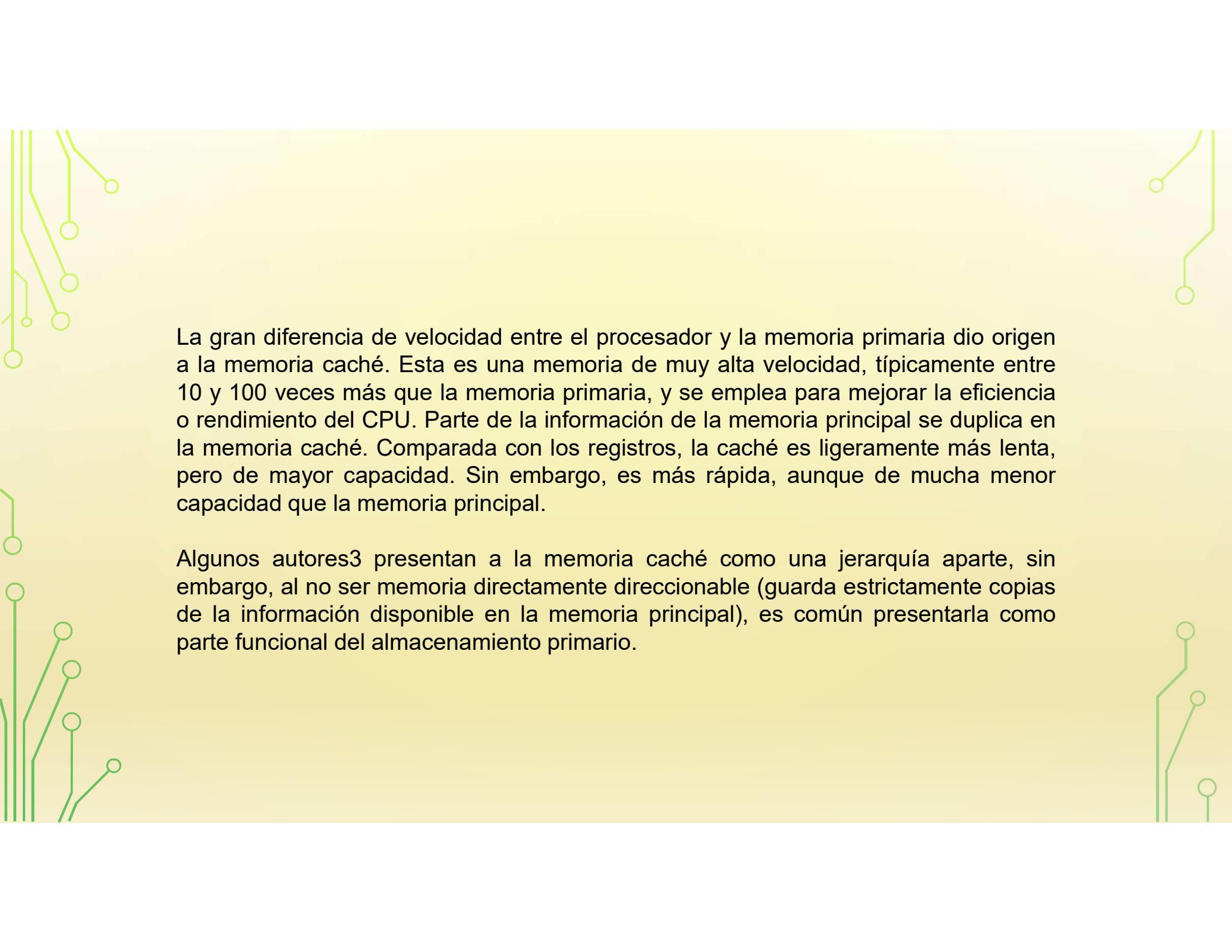
Una computadora digital representa toda la información usando el sistema binario. Texto, números, imágenes, sonido y casi cualquier otra forma de información puede ser transformada en una sucesión de bits, o dígitos binarios, cada uno de los cuales tiene un valor de 1 o 0. La unidad de almacenamiento más común es el byte, igual a 8 bits. Una determinada información puede ser manipulada por cualquier computadora cuyo espacio de almacenamiento sea suficientemente grande como para que quepa el dato correspondiente o la representación binaria de la información. Por ejemplo, una computadora con un espacio de almacenamiento de ocho millones de bits, o un megabyte, puede ser usada para editar una novela pequeña.

Se han inventado varias formas de almacenamiento basadas en diversos fenómenos naturales. No existe ningún medio de almacenamiento de uso práctico universal y todas las formas de almacenamiento tienen sus desventajas. Por tanto, un sistema informático contiene varios tipos de almacenamiento, cada uno con su propósito individual.

Almacenamiento primario (Memoria principal)

La memoria primaria, está directamente conectada al CPU de la computadora. Debe estar presente para que el CPU efectúe cualquier función. El almacenamiento primario consta de la memoria primaria del sistema; contiene los programas en ejecución y los datos con que operan. Se puede transferir información muy rápidamente (típicamente en menos de 100 ciclos de reloj) entre un registro del microprocesador y localizaciones del almacenamiento principal. En las computadoras modernas se usan memorias de acceso aleatorio basadas en electrónica del estado sólido, que está directamente conectada a la CPU a través de buses de direcciones, datos y control.

El almacenamiento lleva por principal requisito que cualquiera de sus localidades debe ser directamente direccionable, esto es, todo dato contenido en memoria debe poder encontrarse basándose en su dirección. Es por esto que los registros del procesador no pueden considerarse almacenamiento primario. Las referencias a estos se efectúan por nombre, de forma directa, y no por dirección. Los registros representan el estado actual del cómputo y los datos utilizados inmediatamente, pero no pueden almacenar un programa (solo apuntar al lugar de ejecución actual).



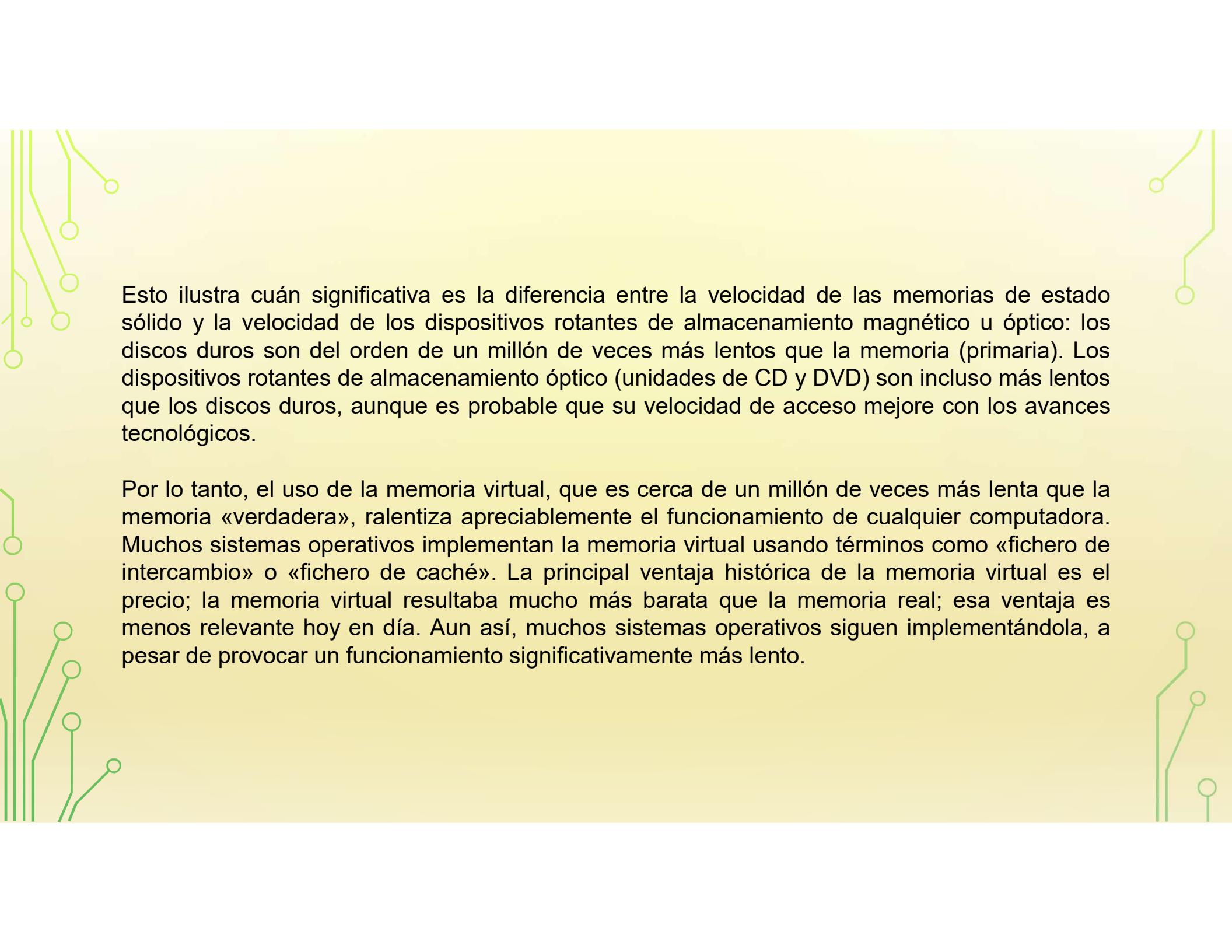
La gran diferencia de velocidad entre el procesador y la memoria primaria dio origen a la memoria caché. Esta es una memoria de muy alta velocidad, típicamente entre 10 y 100 veces más que la memoria primaria, y se emplea para mejorar la eficiencia o rendimiento del CPU. Parte de la información de la memoria principal se duplica en la memoria caché. Comparada con los registros, la caché es ligeramente más lenta, pero de mayor capacidad. Sin embargo, es más rápida, aunque de mucha menor capacidad que la memoria principal.

Algunos autores³ presentan a la memoria caché como una jerarquía aparte, sin embargo, al no ser memoria directamente direccionable (guarda estrictamente copias de la información disponible en la memoria principal), es común presentarla como parte funcional del almacenamiento primario.

Almacenamiento secundario (Memoria secundaria)

La memoria secundaria requiere que la computadora use sus canales de entrada/salida para acceder a la información y se utiliza para almacenamiento a largo plazo de información persistente. Sin embargo, la mayoría de los sistemas operativos usan los dispositivos de almacenamiento secundario como área de intercambio para incrementar artificialmente la cantidad aparente de memoria principal en la computadora (a esta utilización del almacenamiento secundario se le denomina memoria virtual). La memoria secundaria también se llama de «almacenamiento masivo». Un disco duro es un ejemplo de almacenamiento secundario.

Habitualmente, la memoria secundaria o de almacenamiento masivo tiene mayor capacidad que la memoria primaria, pero es mucho más lenta. En las computadoras modernas, los discos duros suelen usarse como dispositivos de almacenamiento masivo. El tiempo necesario para acceder a un byte de información dado almacenado en un disco duro de platos magnéticos es de unas milésimas de segundo (milisegundos). En cambio, el tiempo para acceder al mismo tipo de información en una memoria de acceso aleatorio (RAM) se mide en mil-millonésimas de segundo (nanosegundos).



Esto ilustra cuán significativa es la diferencia entre la velocidad de las memorias de estado sólido y la velocidad de los dispositivos rotantes de almacenamiento magnético u óptico: los discos duros son del orden de un millón de veces más lentos que la memoria (primaria). Los dispositivos rotantes de almacenamiento óptico (unidades de CD y DVD) son incluso más lentos que los discos duros, aunque es probable que su velocidad de acceso mejore con los avances tecnológicos.

Por lo tanto, el uso de la memoria virtual, que es cerca de un millón de veces más lenta que la memoria «verdadera», ralentiza apreciablemente el funcionamiento de cualquier computadora. Muchos sistemas operativos implementan la memoria virtual usando términos como «fichero de intercambio» o «fichero de caché». La principal ventaja histórica de la memoria virtual es el precio; la memoria virtual resultaba mucho más barata que la memoria real; esa ventaja es menos relevante hoy en día. Aun así, muchos sistemas operativos siguen implementándola, a pesar de provocar un funcionamiento significativamente más lento.

Almacenamiento terciario.

La memoria terciaria es un sistema en el que un robot industrial —brazo robótico—, montará —conectará— un medio de almacenamiento masivo fuera de línea (véase el siguiente punto) según lo solicite el sistema operativo de la computadora. La memoria terciaria se usa en el área del almacenamiento industrial, la computación científica, en grandes sistemas informáticos y en redes empresariales. Este tipo de memoria es algo que los usuarios de computadoras personales nunca ven de primera mano.

Almacenamiento fuera de línea.

El almacenamiento fuera de línea (off-line) es un sistema donde el medio de almacenamiento puede ser extraído fácilmente del dispositivo de almacenamiento. Estos medios de almacenamiento suelen usarse para transporte y archivo de datos. En computadoras modernas son de uso habitual para este propósito los discos duros, discos ópticos y las memorias flash, incluyendo las unidades USB. También hay discos duros USB que se pueden conectar rápidamente. Los dispositivos de almacenamiento fuera de línea usados en el pasado son cintas magnéticas en muchos tamaños y formatos diferentes, y las baterías extraíbles de discos Winchester.

Almacenamiento de red (Almacenamiento en nube y Servicio de alojamiento de archivos).

El almacenamiento de red es cualquier tipo de almacenamiento de computadora que incluye el hecho de acceder a la información a través de una red informática. Discutiblemente, el almacenamiento de red permite centralizar el «control de información» en una organización y reducir la duplicidad de la información. El almacenamiento en red incluye:

El almacenamiento asociado a red es una memoria secundaria o terciaria que reside en una computadora a la que otra de éstas puede acceder a través de una red de área local, una red de área extensa, una red privada virtual o, en el caso de almacenamiento de archivos en línea, internet.

Las redes de computadoras son computadoras que no contienen dispositivos de almacenamiento secundario. En su lugar, los documentos y otros datos son almacenados en un dispositivo de la red.

Características de las memorias

La división entre primario, secundario, terciario y fuera de línea, se basa en la jerarquía de memoria o distancia desde la CPU. Hay otras formas de caracterizar a los distintos tipos de memoria.

Volatilidad de la información.

La memoria volátil requiere energía constante para mantener la información almacenada. La memoria volátil se suele usar solo en memorias primarias. La memoria RAM es una memoria volátil, ya que pierde información en la falta de energía eléctrica.

La memoria no volátil retendrá la información almacenada incluso si no recibe corriente eléctrica constantemente, como es el caso de la memoria ROM. Se usa para almacenamientos a largo plazo y, por lo tanto, se usa en memorias secundarias, terciarias y fuera de línea.

La memoria dinámica es una memoria volátil que además requiere que periódicamente se refresque la información almacenada, o leída y reescrita sin modificaciones.

Accesibilidad secuencial o aleatoria a información



Dependiendo de la habilidad para acceder a información contigua o no, se puede clasificar en:

Acceso aleatorio, significa que se puede acceder a cualquier localización de la memoria en cualquier momento en el mismo intervalo de tiempo, normalmente pequeño.

Acceso secuencial, significa que acceder a una unidad de información tomará un intervalo de tiempo variable, dependiendo de la unidad de información que fue leída anteriormente. El dispositivo puede necesitar buscar (posicionar correctamente el cabezal de lectura/escritura de un disco), o dar vueltas (esperando a que la posición adecuada aparezca debajo del cabezal de lectura/escritura en un medio que gira continuamente).

Habilidad para cambiar la información.

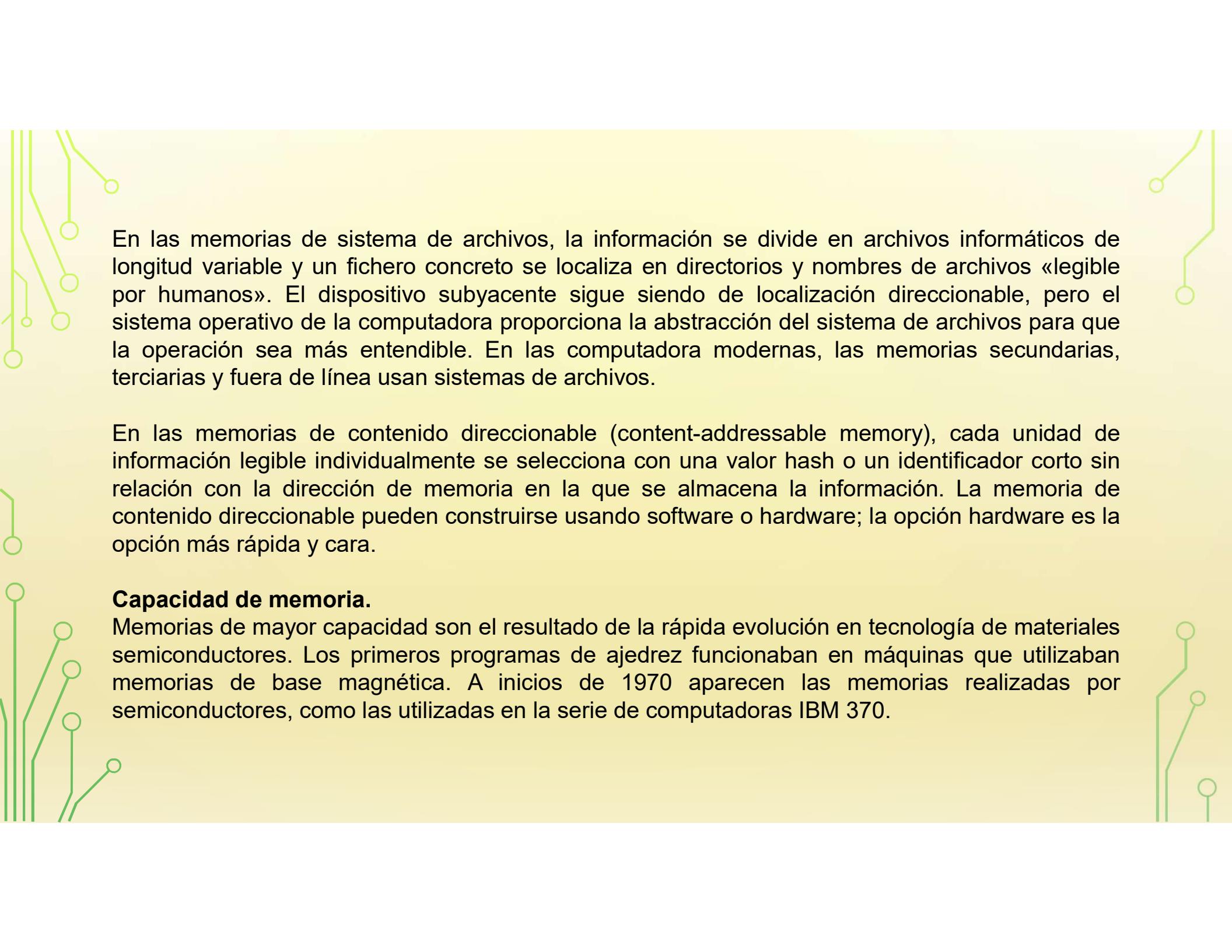
Las memorias de lectura/escritura o memorias cambiables permiten que la información se reescriba en cualquier momento. Una computadora sin algo de memoria de lectura/escritura como memoria principal sería inútil para muchas tareas. Las computadoras modernas también usan habitualmente memorias de lectura/escritura como memoria secundaria.

La memoria de sólo lectura (Read-Only Memory, ROM) retiene la información almacenada en el momento de fabricarse y la memoria de escritura única lectura múltiple (Write Once Read Many, WORM) permite que la información se escriba una sola vez en algún momento tras la fabricación. También están las memorias inmutables, que se utilizan en memorias terciarias y fuera de línea. Un ejemplo son los CD-ROM.

Las memorias de escritura lenta y lectura rápida son memorias de lectura/escritura que permite que la información se reescriba múltiples veces pero con una velocidad de escritura mucho menor que la de lectura. Un ejemplo son los CD-RW.

Direccionamiento de la información

En la memoria de localización direccionable, cada unidad de información accesible individualmente en la memoria se selecciona con su dirección de memoria numérica. En las computadoras modernas, la memoria de localización direccionable se suele limitar a memorias primarias, que se leen internamente por programas de computadora ya que la localización direccionable es muy eficiente, pero difícil de usar para los humanos.



En las memorias de sistema de archivos, la información se divide en archivos informáticos de longitud variable y un fichero concreto se localiza en directorios y nombres de archivos «legible por humanos». El dispositivo subyacente sigue siendo de localización direccionable, pero el sistema operativo de la computadora proporciona la abstracción del sistema de archivos para que la operación sea más entendible. En las computadoras modernas, las memorias secundarias, terciarias y fuera de línea usan sistemas de archivos.

En las memorias de contenido direccionable (content-addressable memory), cada unidad de información legible individualmente se selecciona con un valor hash o un identificador corto sin relación con la dirección de memoria en la que se almacena la información. La memoria de contenido direccionable pueden construirse usando software o hardware; la opción hardware es la opción más rápida y cara.

Capacidad de memoria.

Memorias de mayor capacidad son el resultado de la rápida evolución en tecnología de materiales semiconductores. Los primeros programas de ajedrez funcionaban en máquinas que utilizaban memorias de base magnética. A inicios de 1970 aparecen las memorias realizadas por semiconductores, como las utilizadas en la serie de computadoras IBM 370.

The background of the slide features a subtle, abstract pattern of thin green lines and small white circles, resembling a network or a stylized tree structure.

La velocidad de los computadores se incrementó, multiplicada por 100.000 aproximadamente y la capacidad de memoria creció en una proporción similar. Este hecho es particularmente importante para los programas que utilizan tablas de transposición: a medida que aumenta la velocidad de la computadora se necesitan memorias de capacidad proporcionalmente mayor para mantener la cantidad extra de posiciones que el programa está buscando.

Se espera que la capacidad de procesadores siga aumentando en los próximos años; no es un abuso pensar que la capacidad de memoria continuará creciendo de manera impresionante. Memorias de mayor capacidad podrán ser utilizadas por programas con tablas de Hash de mayor envergadura, las cuales mantendrán la información en forma permanente.

Minicomputadoras: se caracterizan por tener una configuración básica regular que puede estar compuesta por un monitor, unidades de disquete, disco, impresora, etc. Su capacidad de memoria varía de 16 a 256 KiB.

Macrocomputadoras: son aquellas que dentro de su configuración básica contienen unidades que proveen de capacidad masiva de información, terminales (monitores), etc. Su capacidad de memoria varía desde 256 a 512 KiB, también puede tener varios megabytes o hasta gigabytes según las necesidades de la empresa.

Microcomputadores y computadoras personales: con el avance de la microelectrónica en la década de los 70 resultaba posible incluir todos los componentes del procesador central de una computadora en un solo circuito integrado llamado microprocesador. Ésta fue la base de creación de unas computadoras a las que se les llamó microcomputadoras. El origen de las microcomputadoras tuvo lugar en los Estados Unidos a partir de la comercialización de los primeros microprocesadores (INTEL 8008, 8080). En la década de los 80 comenzó la verdadera explosión masiva, de los ordenadores personales (Personal Computer PC) de IBM. Esta máquina, basada en el microprocesador INTEL 8008, tenía características interesantes que hacían más amplio su campo de operaciones, sobre todo porque su nuevo sistema operativo estandarizado (MS-DOS, Microsoft Disk Operating System) y una mejor resolución óptica, la hacían más atractiva y fácil de usar. El ordenador personal ha pasado por varias transformaciones y mejoras que se conocen como XT(Tecnología Extendida), AT(Tecnología Avanzada) y PS/2...

Tecnologías, dispositivos y medios.

Memoria de semiconductor.

La memoria de semiconductor usa circuitos integrados basados en semiconductores para almacenar información. Un chip de memoria de semiconductor puede contener millones de minúsculos transistores o condensadores. Existen memorias de semiconductor de ambos tipos: volátiles y no volátiles. En las computadoras modernas, la memoria principal consiste casi exclusivamente en memoria de semiconductor volátil y dinámica, también conocida como memoria dinámica de acceso aleatorio o más comúnmente RAM, su acrónimo inglés. Con el cambio de siglo, ha habido un crecimiento constante en el uso de un nuevo tipo de memoria de semiconductor no volátil llamado memoria flash. Dicho crecimiento se ha dado, principalmente en el campo de las memorias fuera de línea en computadoras domésticas. Las memorias de semiconductor no volátiles se están usando también como memorias secundarias en varios dispositivos de electrónica avanzada y computadoras especializadas y no especializadas.

Memoria magnética.

Las memorias magnéticas usan diferentes patrones de magnetización sobre una superficie cubierta con una capa magnetizada para almacenar información. Las memorias magnéticas son no volátiles. Se llega a la información usando uno o más cabezales de lectura/escritura. Como el cabezal de lectura/escritura solo cubre una parte de la superficie, el almacenamiento magnético es de acceso secuencial y debe buscar, dar vueltas o las dos cosas. En ‘computadoras modernas’, la superficie magnética es de alguno de estos tipos:

- Disquete, usado para memoria fuera de línea.
- Disco duro, usado para memoria secundario.
- Cinta magnética, usada para memoria terciaria y fuera de línea.

En las ‘primeras computadoras’, el almacenamiento magnético se usaba también como memoria principal en forma de memoria de tambor, memoria de núcleo, memoria en hilera de núcleo, memoria película delgada, memoria de Twistor o memoria de burbuja. Además, a diferencia de hoy, las cintas magnéticas se solían usar como memoria secundaria.

Memoria de disco óptico.

Las memorias en disco óptico almacenan información usando agujeros minúsculos grabados con un láser en la superficie de un disco circular. La información se lee iluminando la superficie con un diodo láser y observando la reflexión. Los discos ópticos son no volátil y de acceso aleatorio. Los siguientes formatos son de uso común:

- CD, CD-ROM, DVD: Memorias de simplemente solo lectura, usada para distribución masiva de información digital (música, vídeo, programas informáticos).
- CD-R, DVD-R, DVD+R: Memorias de escritura única usada como memoria terciaria y fuera de línea.
- CD-RW, DVD-RW, DVD+RW, DVD-RAM: Memoria de escritura lenta y lectura rápida usada como memoria terciaria y fuera de línea.
- Blu-ray: Formato de disco óptico pensado para almacenar vídeo de alta calidad y datos. Para su desarrollo se creó la BDA, en la que se encuentran, entre otros, Sony o Phillips.

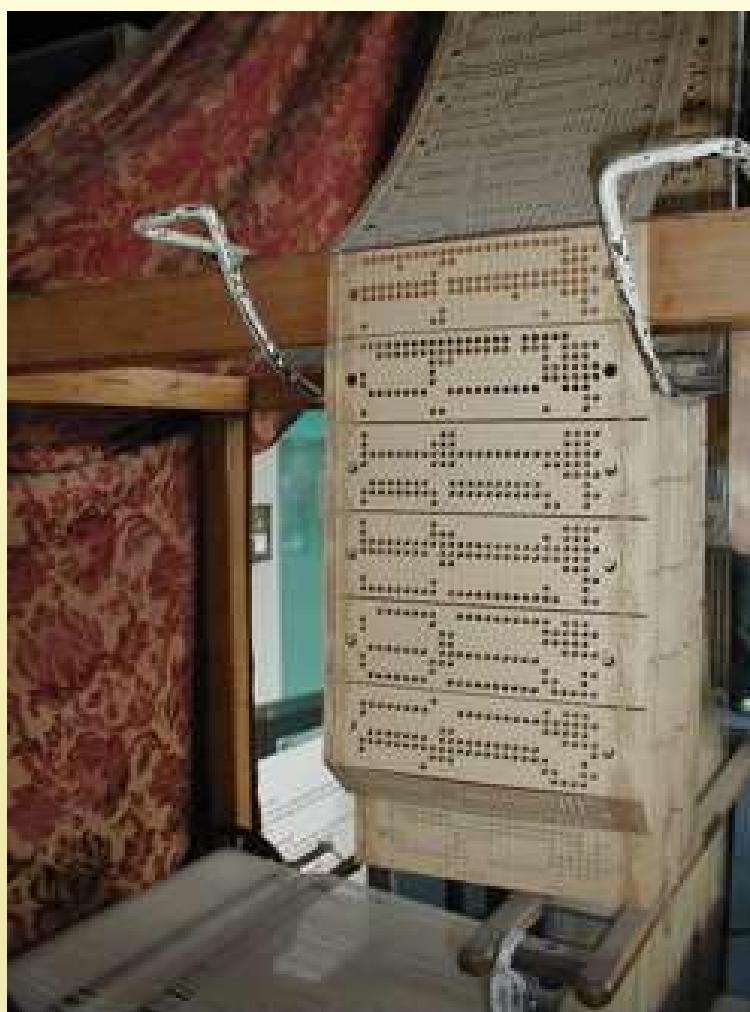
HVD - Discos cambio de fase Dual (Disco óptico).

Memoria de disco magneto-óptico

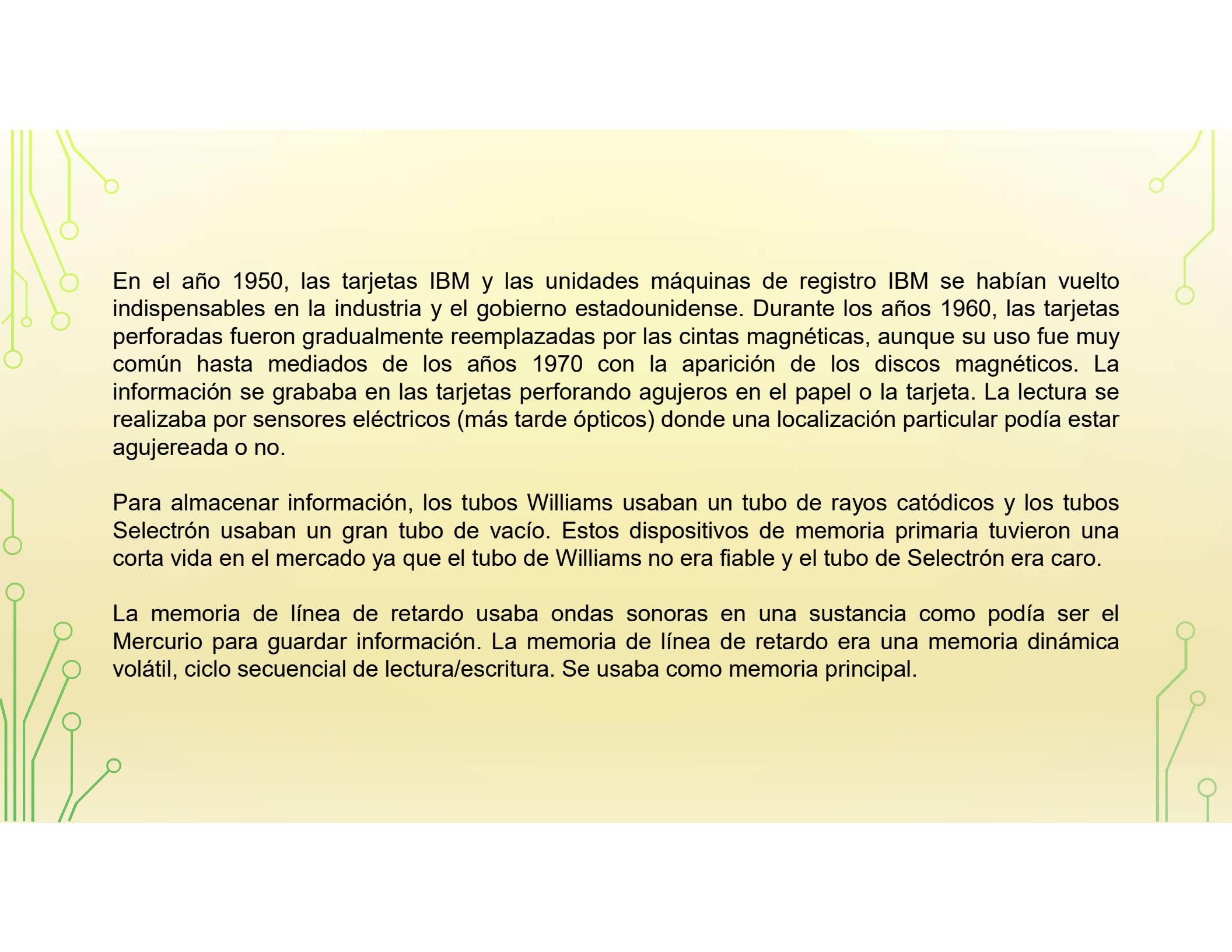
Los discos magneto-ópticos son discos de memoria óptica donde la información se almacena en el estado magnético de una superficie ferromagnética. La información se lee ópticamente y se escribe combinando métodos magnéticos y ópticos. Las memorias de discos magneto ópticos son de tipo no volátiles, de acceso secuencial, de escritura lenta y lectura rápida. Se usa como memoria terciaria y fuera de línea.

Tarjetas perforadas en un telar de Jacquard.

Las tarjetas perforadas fueron utilizados por primera vez por Basile Bouchon para el control de telares textiles en Francia.⁴ En 1801 el sistema de Bouchon fue perfeccionado por Joseph Marie Jacquard, quien desarrolló un telar automático, conocido como telar de Jacquard.⁵ Herman Hollerith desarrolló la tecnología de procesamiento de datos de tarjetas perforadas para el censo de Estados Unidos de 1890 y posteriormente fundó la Tabulating Machine Company, una de las precursoras de IBM. IBM desarrolló la tecnología de la tarjeta perforada como una potente herramienta para el procesamiento de datos empresariales y produjo una línea extensiva de máquinas de registro que utilizaban papel perforado para el almacenamiento de datos y su procesado automático.



Tarjetas perforadas en un telar de Jacquard



En el año 1950, las tarjetas IBM y las unidades máquinas de registro IBM se habían vuelto indispensables en la industria y el gobierno estadounidense. Durante los años 1960, las tarjetas perforadas fueron gradualmente reemplazadas por las cintas magnéticas, aunque su uso fue muy común hasta mediados de los años 1970 con la aparición de los discos magnéticos. La información se grababa en las tarjetas perforando agujeros en el papel o la tarjeta. La lectura se realizaba por sensores eléctricos (más tarde ópticos) donde una localización particular podía estar agujereada o no.

Para almacenar información, los tubos Williams usaban un tubo de rayos catódicos y los tubos Selectrón usaban un gran tubo de vacío. Estos dispositivos de memoria primaria tuvieron una corta vida en el mercado ya que el tubo de Williams no era fiable y el tubo de Selectrón era caro.

La memoria de línea de retardo usaba ondas sonoras en una sustancia como podía ser el Mercurio para guardar información. La memoria de línea de retardo era una memoria dinámica volátil, ciclo secuencial de lectura/escritura. Se usaba como memoria principal.

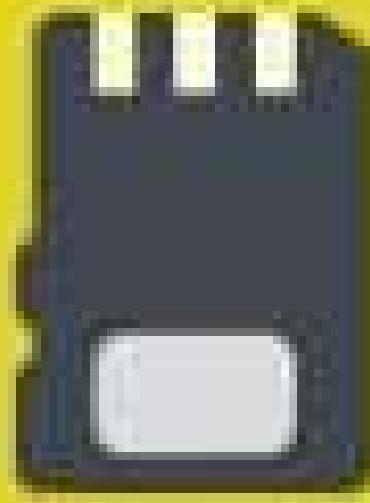
Otros métodos propuestos.

La memoria de cambio de fase usa las fases de un material de cambio de fase para almacenar información. Dicha información se lee observando la resistencia eléctrica variable del material. La memoria de cambio de fase sería una memoria de lectura/escritura no volátil, de acceso aleatorio podría ser usada como memoria primaria, secundaria y fuera de línea. La memoria holográfica almacena ópticamente la información dentro de cristales o fotopolímeros. Las memorias holográficas pueden utilizar todo el volumen del medio de almacenamiento, a diferencia de las memorias de discos ópticos, que están limitadas a un pequeño número de superficies en capas. La memoria holográfica podría ser no volátil, de acceso secuencial y tanto de escritura única como de lectura/escritura. Puede ser usada tanto como memoria secundaria como fuera de línea.

La memoria molecular almacena la información en polímeros que pueden almacenar puntas de carga eléctrica. La memoria molecular puede ser especialmente interesante como memoria principal.

Recientemente se ha propuesto utilizar el spin de un electrón como memoria. Se ha demostrado que es posible desarrollar un circuito electrónico que lea el spin del electrón y lo convierta en una señal eléctrica.

Diferencias entre
BIT BYTE
1KB 1MB
1GB 1TB



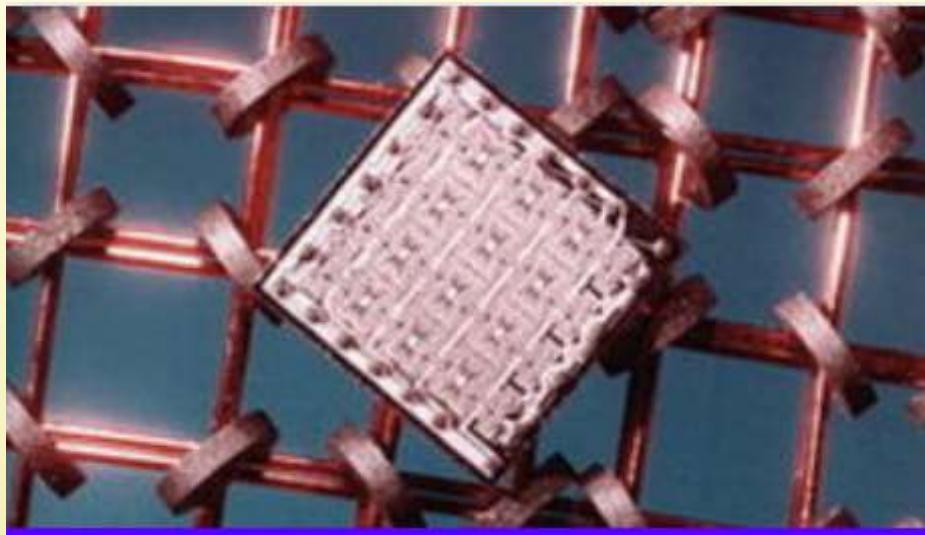
<https://youtu.be/kaoA6B8PT4M>



OCT 4, 1949

Primera memoria RAM

Uno de los primeros tipos de memoria RAM fue la memoria de núcleo magnético, desarrollada entre 1949 y 1952 y usada en muchos computadores hasta el desarrollo de circuitos integrados a finales de los años 60 y principios de los 70.



OCT 4, 1969

Memoria de núcleo magnético

En 1969 fueron lanzadas una de las primeras memorias RAM basadas en semiconductores de silicio por parte de Intel con el integrado 3101 de 64 bits de memoria y para el siguiente año se presentó una memoria DRAM de 1024 bytes, referencia 1103 que se constituyó en un hito, ya que fue la primera en ser comercializada con éxito, lo que significó el principio del fin para las memorias de núcleo magnético.

OCT 4, 1973

MK4096

En 1973 se presentó una innovación que permitió otra miniaturización y se convirtió en estándar para las memorias DRAM: la multiplexación en tiempo de la direcciones de memoria. MOSTEK lanzó la referencia MK4096 de 4096 bytes en un empaque de 16 pines

OCT 4, 1980

FPM-RAM

Fast Page Mode RAM (FPM-RAM) fue inspirado en técnicas como el Burst Mode usado en procesadores como el Intel 486. Se implantó un modo direccionamiento en el que el controlador de memoria envía una sola dirección y recibe a cambio esa y varias consecutivas sin necesidad de generar todas las direcciones.

OCT 4, 1986

BEDO-RAM

Burst Extended Data Output RAM (BEDO-RAM) fue la evolución de la EDO-RAM y competidora de la SDRAM, fue presentada en 1997. Era un tipo de memoria que usaba generadores internos de direcciones y accedía a más de una posición de memoria en cada ciclo de reloj, de manera que lograba un desempeño un 50 % mejor que la EDO.



OCT 4, 1994

EDO RAM

Extended Data Output RAM (EDO-RAM) fue lanzada al mercado en 1994 y con tiempos de accesos de 40 o 30 ns suponía una mejora sobre FPM, su antecesora. La EDO, también es capaz de enviar direcciones contiguas pero direcciona la columna que va utilizar mientras que se lee la información de la columna anterior, dando como resultado una eliminación de estados de espera, manteniendo activo el búfer de salida hasta que comienza el próximo ciclo de lectura.



OCT 4, 2016

MEMORIA RAM ALTA CALIDAD 2016

DDR4 SDRAM es la abreviatura de "memoria dinámica de acceso aleatorio sincronizada de cuarta generación y doble velocidad de datos", la última variante de memoria en computación. DDR4 es capaz de lograr una mayor velocidad y eficiencia gracias a mayores tasas de transferencia y menor voltaje.

DDR5 es la evolución de la memoria RAM DDR4, la cual convivía en las plataformas AM4 de Ryzen, LGA1200, LGA1151 y LGA2066 a lo que escritorio de Intel se refiere. Las principales mejoras son la densidad por chip, el ancho de banda y un menor voltaje.

La primera mejora es que los chips de memoria que van soldados en el módulo tienen más capacidad, permitiendo ver módulos DIMM individuales de 128 GB. Dicho esto, los fabricantes han partido por ofrecer 16 GB como mínimo por módulo, siendo el nuevo “estándar” los packs de 32 GB DDR5.

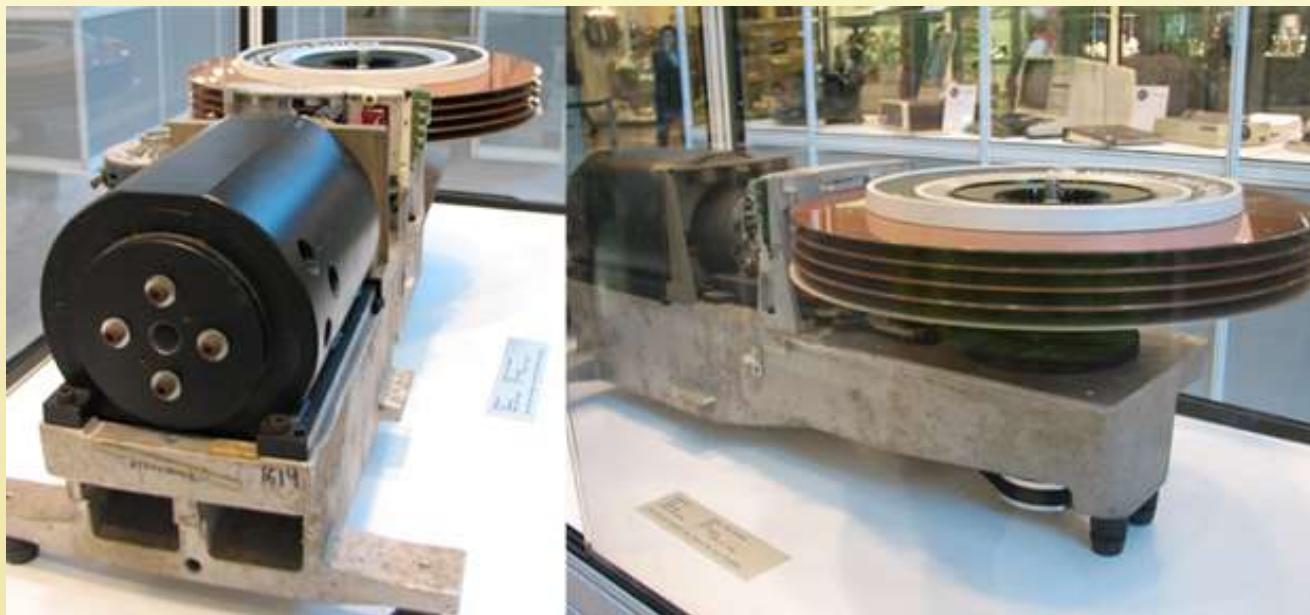
Sin embargo, no tardarán en llegar los packs de 64 GB y más capacidad de memoria; eso sí, hay que tener en cuenta que tenemos la limitación de las placas base a equipar una capacidad límite: 128 GB. Este límite siempre sonaba muy lejano, pero con la llegada de memorias DDR5 ya no lo será.

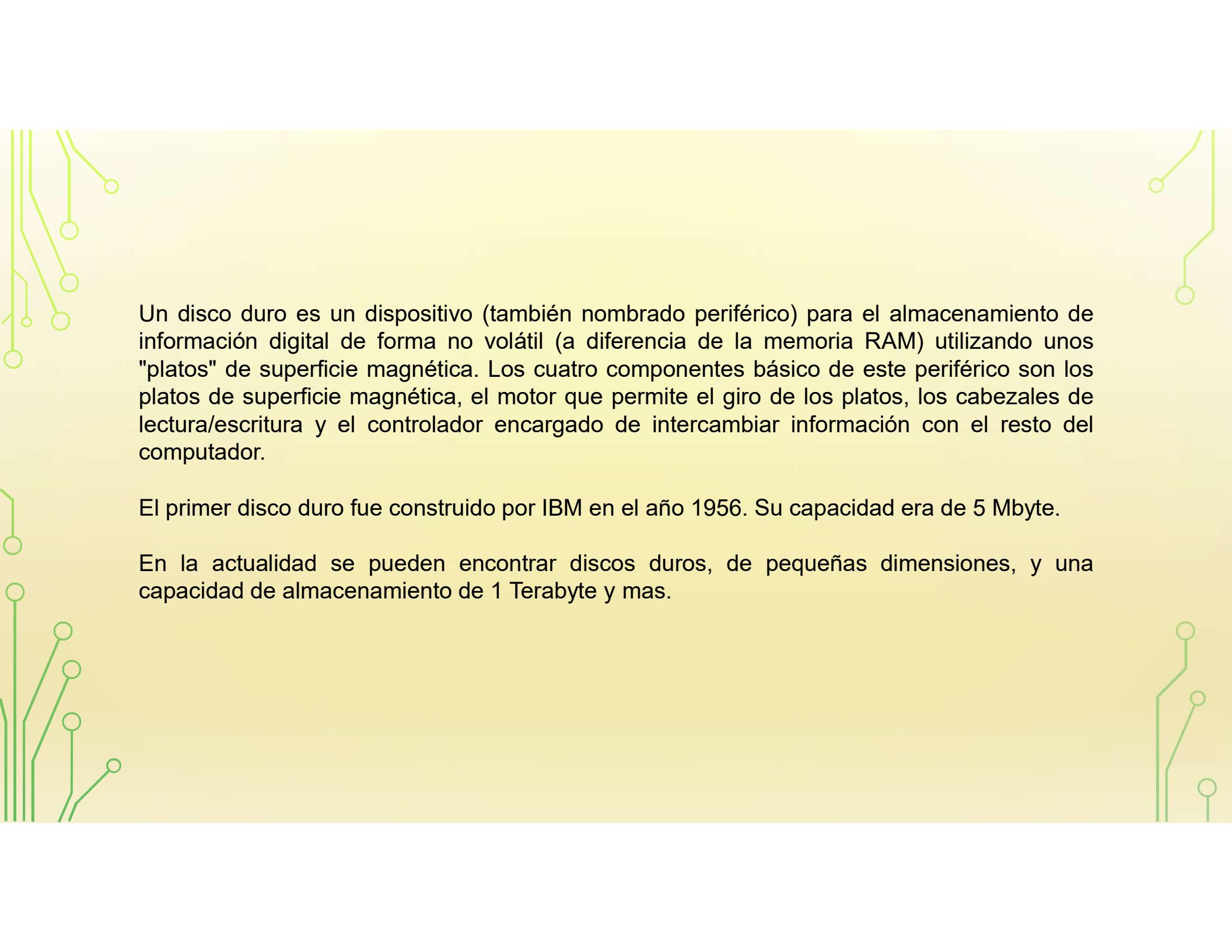
En segundo lugar, la frecuencia de las memorias ha mejorado brutalmente: las primeras memorias partirían de los 4.800 MHz. Recapitulando un poco, las primeras memorias RAM DDR4 venían con una frecuencia de 2133 MHz, que era la que funcionaba por defecto.

SISTEMAS OPERATIVOS 2

LABORATORIO

DISCO DURO

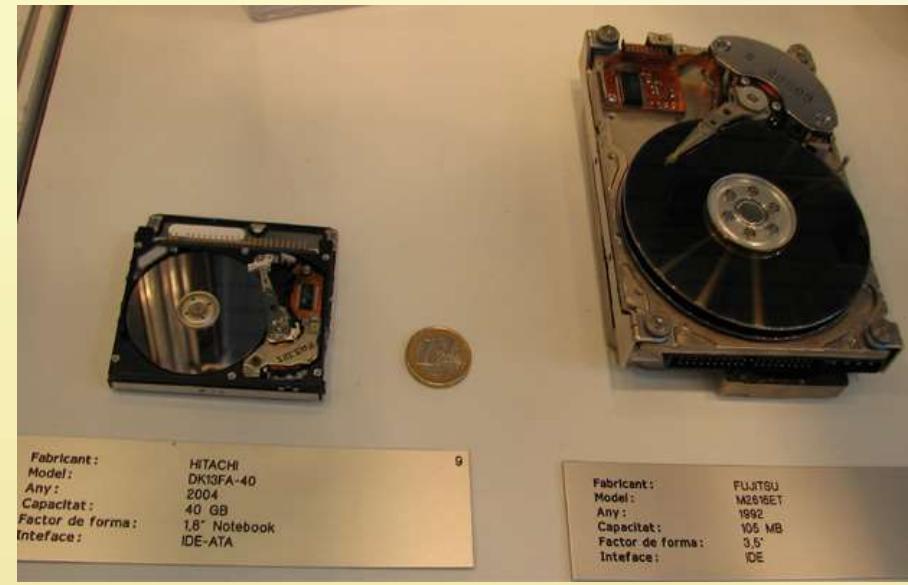
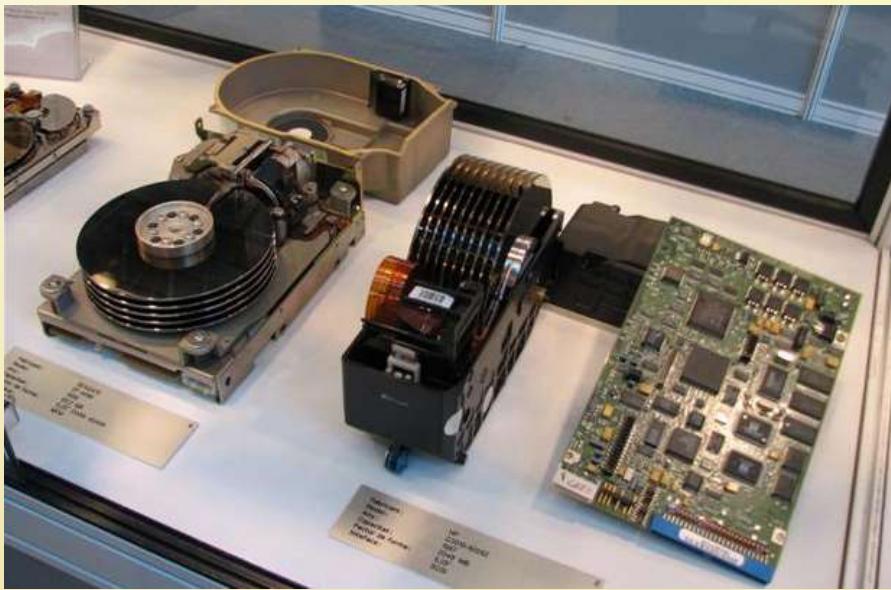


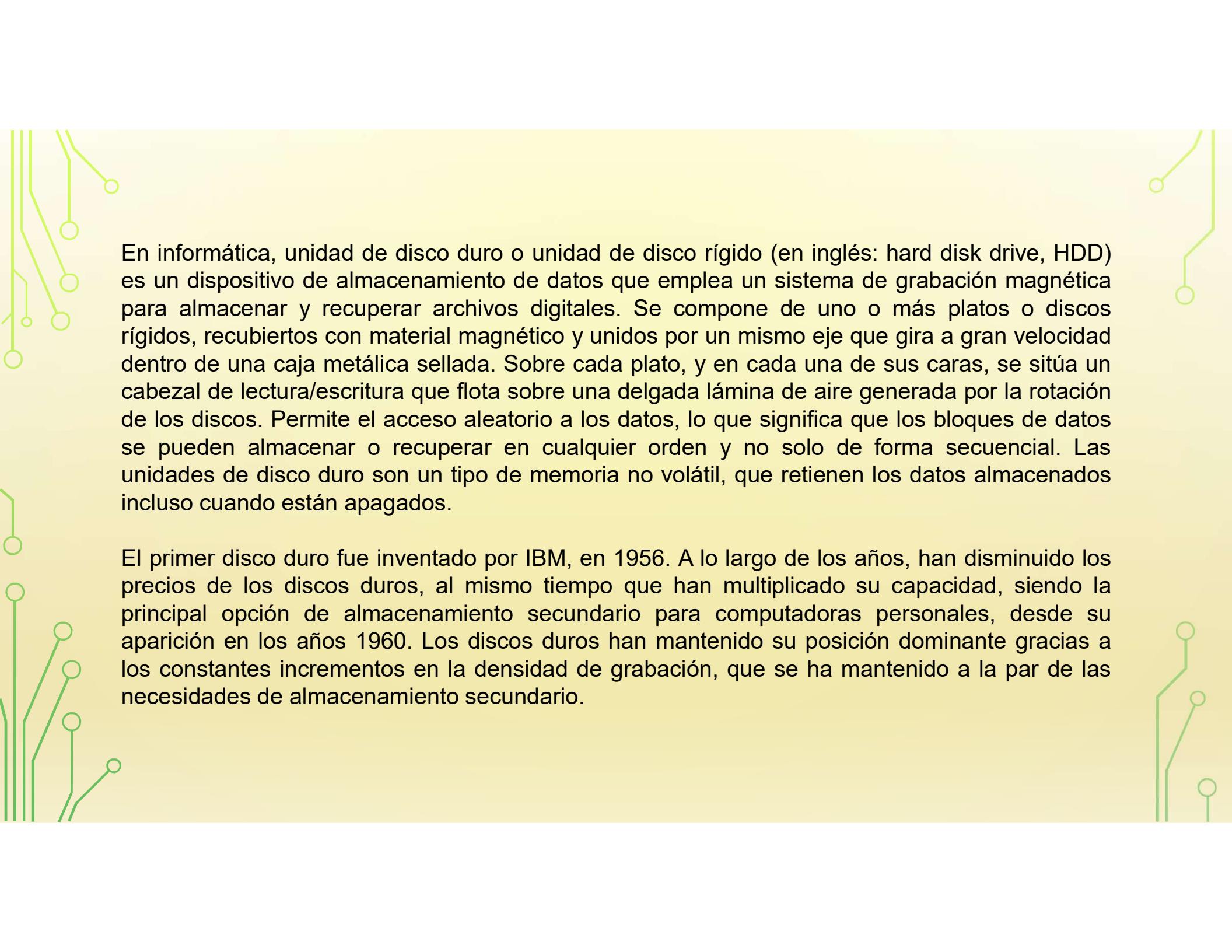


Un disco duro es un dispositivo (también nombrado periférico) para el almacenamiento de información digital de forma no volátil (a diferencia de la memoria RAM) utilizando unos "platos" de superficie magnética. Los cuatro componentes básicos de este periférico son los platos de superficie magnética, el motor que permite el giro de los platos, los cabezales de lectura/escritura y el controlador encargado de intercambiar información con el resto del computador.

El primer disco duro fue construido por IBM en el año 1956. Su capacidad era de 5 Mbyte.

En la actualidad se pueden encontrar discos duros, de pequeñas dimensiones, y una capacidad de almacenamiento de 1 Terabyte y más.





En informática, unidad de disco duro o unidad de disco rígido (en inglés: hard disk drive, HDD) es un dispositivo de almacenamiento de datos que emplea un sistema de grabación magnética para almacenar y recuperar archivos digitales. Se compone de uno o más platos o discos rígidos, recubiertos con material magnético y unidos por un mismo eje que gira a gran velocidad dentro de una caja metálica sellada. Sobre cada plato, y en cada una de sus caras, se sitúa un cabezal de lectura/escritura que flota sobre una delgada lámina de aire generada por la rotación de los discos. Permite el acceso aleatorio a los datos, lo que significa que los bloques de datos se pueden almacenar o recuperar en cualquier orden y no solo de forma secuencial. Las unidades de disco duro son un tipo de memoria no volátil, que retienen los datos almacenados incluso cuando están apagados.

El primer disco duro fue inventado por IBM, en 1956. A lo largo de los años, han disminuido los precios de los discos duros, al mismo tiempo que han multiplicado su capacidad, siendo la principal opción de almacenamiento secundario para computadoras personales, desde su aparición en los años 1960. Los discos duros han mantenido su posición dominante gracias a los constantes incrementos en la densidad de grabación, que se ha mantenido a la par de las necesidades de almacenamiento secundario.

Mejorados continuamente, los discos duros han mantenido esta posición en la era moderna de los servidores y las computadoras personales. Más de 224 compañías han fabricado unidades de disco duro históricamente, aunque después de una extensa consolidación de la industria, la mayoría de las unidades son fabricadas por Seagate, Toshiba y Western Digital. Los discos duros dominan el volumen de almacenamiento producido (exabytes por año) para servidores. Aunque la producción está creciendo lentamente, los ingresos por ventas y los envíos de unidades están disminuyendo debido a que las unidades de estado sólido (SSD) tienen mayores tasas de transferencia de datos, mayor densidad de almacenamiento de área, mejor confiabilidad, y tiempos de acceso y latencia mucho más bajos.

Los ingresos por SSD, la mayoría de los cuales utilizan NAND, exceden ligeramente los de los HDD. Aunque los SSD tienen un costo por bit casi 10 veces mayor, están reemplazando a los discos duros en aplicaciones donde la velocidad, el consumo de energía, el tamaño pequeño y la durabilidad son importantes.

Los tamaños también han variado mucho, desde los primeros discos IBM hasta los formatos estandarizados actualmente: 3,5 pulgadas los modelos para PC y servidores, y 2,5 pulgadas los modelos para dispositivos portátiles. Todos se comunican con la computadora a través del controlador de disco, empleando una interfaz estandarizada. Los más comunes hasta los años 2000 han sido IDE (también llamado ATA o PATA), SCSI/SAS (generalmente usado en servidores y estaciones de trabajo). Desde el 2000 en adelante ha ido masificándose el uso de los SATA. Existe además los discos de canal de fibra (FC), empleados exclusivamente en servidores. Las unidades externas se conectan principalmente por USB.

Para poder utilizar un disco duro, un sistema operativo debe aplicar un formato de bajo nivel que defina una o más particiones. La operación de formateo requiere el uso de una fracción del espacio disponible en el disco, que dependerá del sistema de archivos o formato empleado. Además, los fabricantes de discos duros, unidades de estado sólido y tarjetas flash miden la capacidad de los mismos usando prefijos del Sistema Internacional, que emplean múltiplos de potencias de 1000 según la normativa IEC e IEEE, en lugar de los prefijos binarios, que emplean múltiplos de potencias de 1024, y son los usados por sistemas operativos de Microsoft. Esto provoca que en algunos sistemas operativos sea representado como múltiplos 1024 o como 1000, y por tanto existan confusiones, por ejemplo, un disco duro de 500 GB, en algunos sistemas operativos será representado como 465 GiB (es decir gibibytes; 1 GiB = 1024 MiB) y en otros como 500 GB.

El rendimiento de un disco duro se especifica por el tiempo requerido para mover las cabezas a una pista o cilindro (tiempo de acceso promedio) agregando el tiempo que toma para que el sector deseado se mueva debajo de la cabeza (latencia media, que es una función de la velocidad de rotación física en las revoluciones por minuto) y, finalmente, la velocidad a la que se transmiten los datos (velocidad de datos).

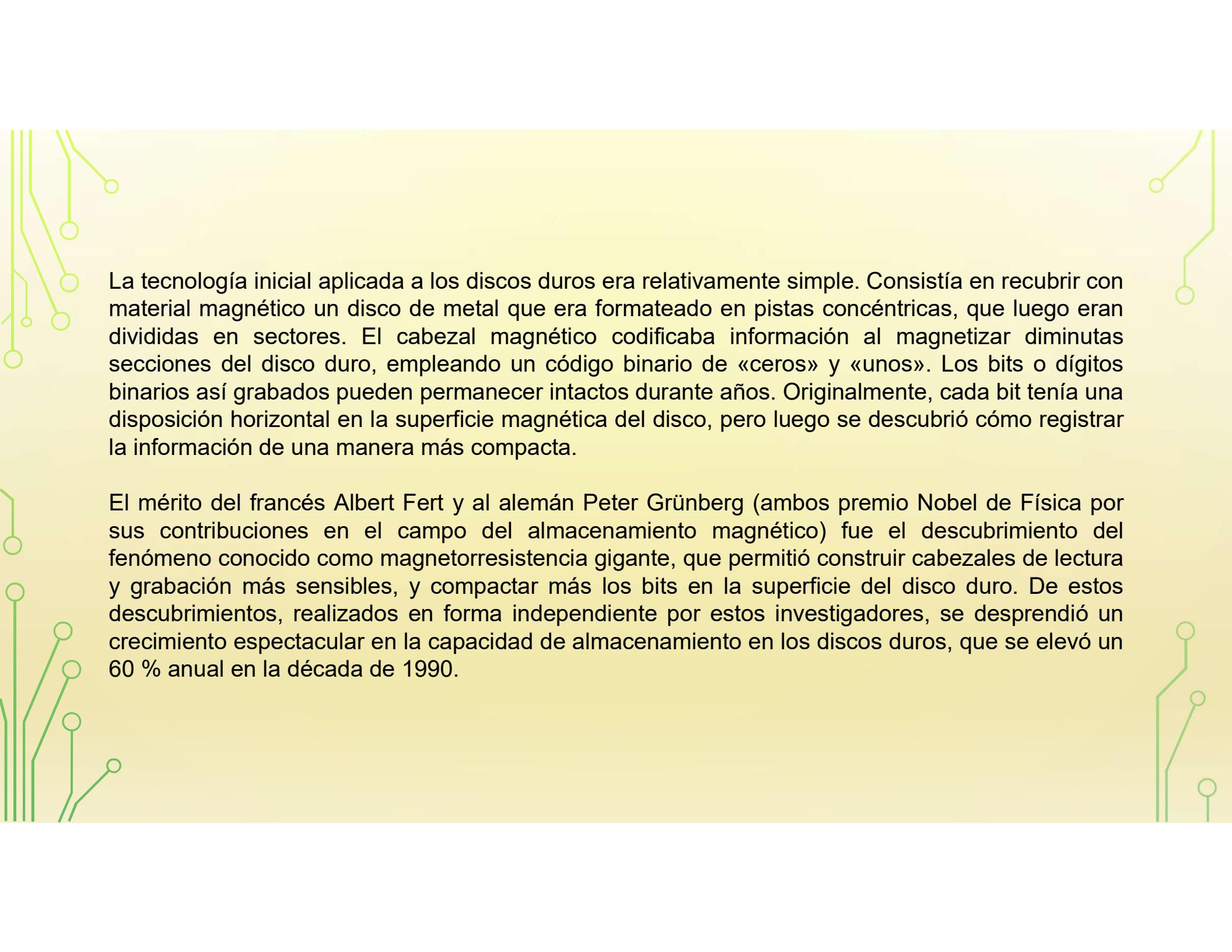


Al principio los discos duros eran extraíbles; sin embargo, hoy en día típicamente vienen todos sellados (a excepción de un agujero de ventilación para filtrar e igualar la presión del aire).

El primer disco duro, aparecido en 1956, fue el Ramac I, presentado con la computadora IBM 350: pesaba una tonelada y su capacidad era de 5 MB. Más grande que una refrigeradora actual, este disco duro trabajaba todavía con válvulas de vacío y requería una consola separada para su manejo.

Su gran mérito consistía en que el tiempo requerido para el acceso era relativamente constante entre algunas posiciones de memoria. Este tipo de acceso se conoce como acceso aleatorio. En cambio, en las cintas magnéticas era necesario enrollar y desenrollar los carretes hasta encontrar el dato buscado, teniendo tiempos de acceso muy dispares para cada posición. Este tipo de acceso se conoce como acceso secuencial.





La tecnología inicial aplicada a los discos duros era relativamente simple. Consistía en recubrir con material magnético un disco de metal que era formateado en pistas concéntricas, que luego eran divididas en sectores. El cabezal magnético codificaba información al magnetizar diminutas secciones del disco duro, empleando un código binario de «ceros» y «unos». Los bits o dígitos binarios así grabados pueden permanecer intactos durante años. Originalmente, cada bit tenía una disposición horizontal en la superficie magnética del disco, pero luego se descubrió cómo registrar la información de una manera más compacta.

El mérito del francés Albert Fert y al alemán Peter Grünberg (ambos premio Nobel de Física por sus contribuciones en el campo del almacenamiento magnético) fue el descubrimiento del fenómeno conocido como magnetoresistencia gigante, que permitió construir cabezales de lectura y grabación más sensibles, y compactar más los bits en la superficie del disco duro. De estos descubrimientos, realizados en forma independiente por estos investigadores, se desprendió un crecimiento espectacular en la capacidad de almacenamiento en los discos duros, que se elevó un 60 % anual en la década de 1990.

En 1992, los discos duros de 3,5 pulgadas alojaban 250 MB, mientras que 10 años después habían superado 40 GB (40 960 MB). A la fecha, ya se dispone en el uso cotidiano con discos duros de más de 5 TB, esto es, 5120 GB (5 242 880 MB).

En 2005 los primeros teléfonos móviles que incluían discos duros fueron presentados por Samsung y Nokia, aunque no tuvieron mucho éxito ya que las memorias flash los acabaron desplazando, debido al aumento de capacidad, mayor resistencia y menor consumo de energía.

Estructura lógica

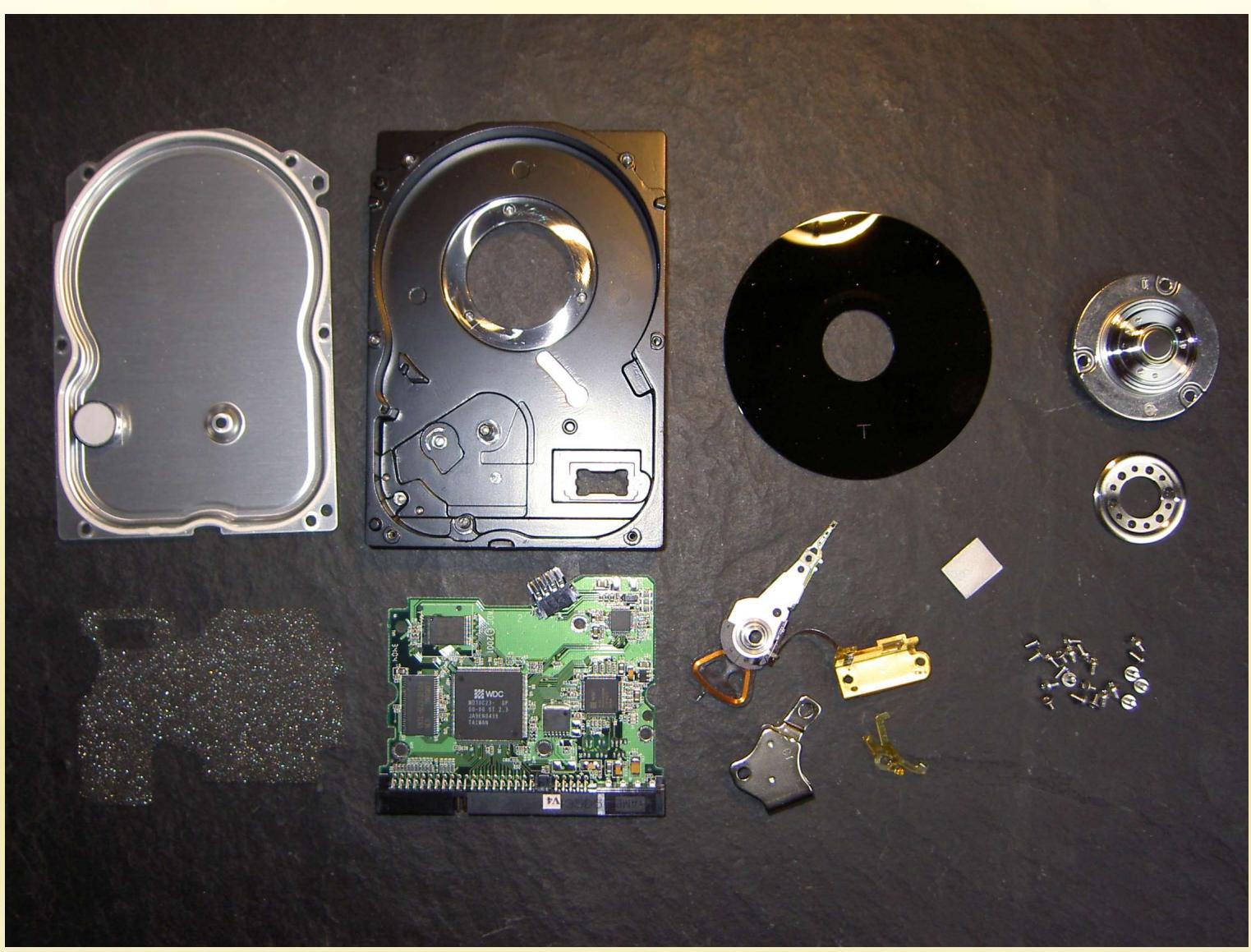
Dentro del disco se encuentran:

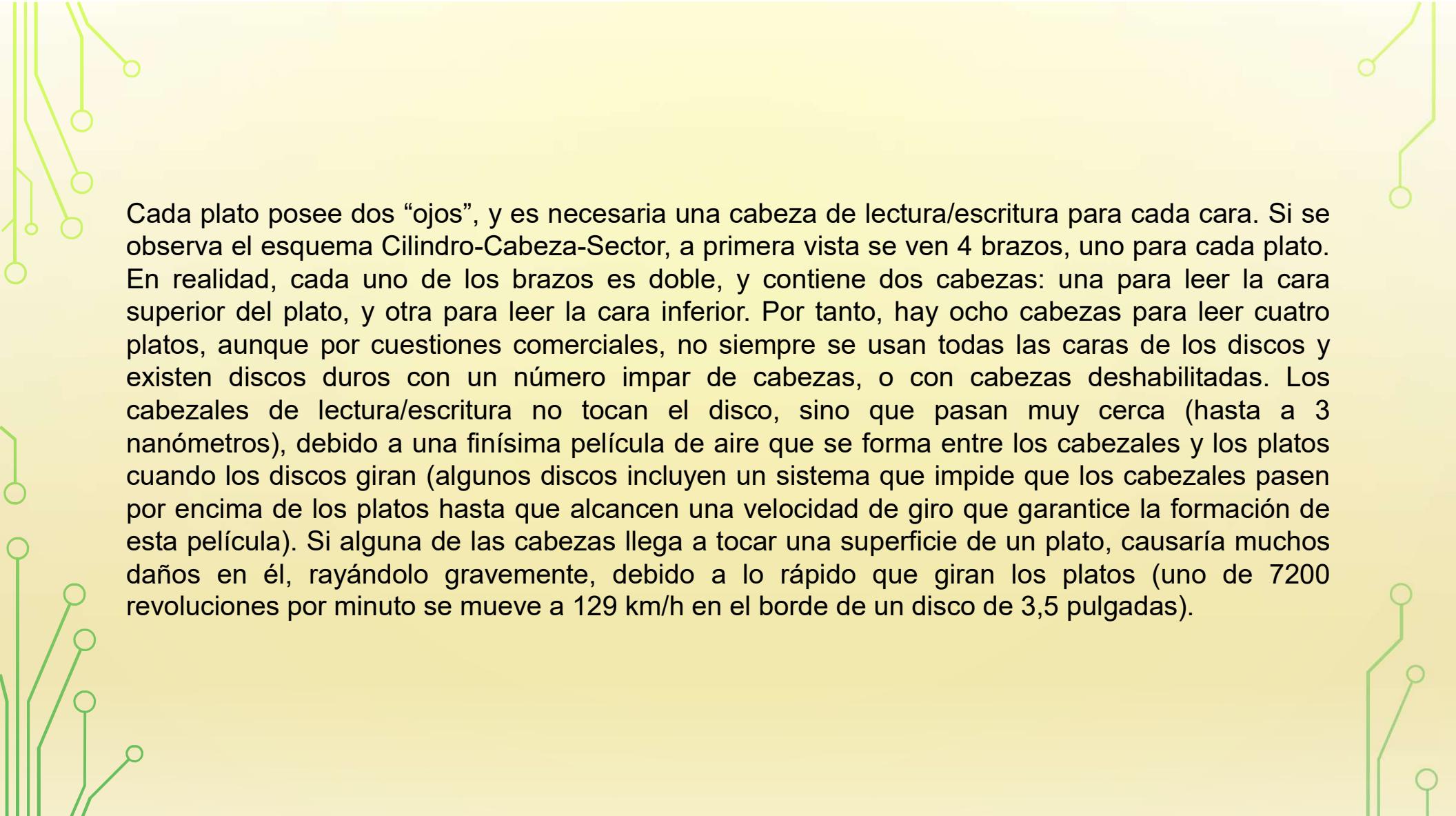
- El registro de arranque principal (Master Boot Record, MBR), en el bloque o sector de arranque, que contiene la tabla de particiones.
- Las particiones de disco, necesarias para poder colocar los sistemas de archivos.

Estructura física

Componentes de una unidad de disco duro. Tapa, carcasa, plato, eje; espuma aislante, circuito impreso de control, cabezal de lectura/escritura, actuador e imán, tornillos.

Dentro de la unidad de disco duro hay uno o varios discos (de aluminio o cristal) concéntricos llamados platos (normalmente entre 2 y 4, aunque pueden ser hasta 6 o 7 según el modelo), y que giran todos a la vez sobre el mismo eje, al que están unidos. El cabezal (dispositivo de lectura y escritura) está formado por un conjunto de brazos paralelos a los platos, alineados verticalmente y que también se desplazan de forma simultánea, en cuya punta están las cabezas de lectura/escritura. Por norma general hay una cabeza de lectura/escritura para cada superficie de cada plato. Los cabezales pueden moverse hacia el interior o el exterior de los platos, lo cual combinado con la rotación de los mismos permite que los cabezales puedan alcanzar cualquier posición de la superficie de los platos.





Cada plato posee dos “ojos”, y es necesaria una cabeza de lectura/escritura para cada cara. Si se observa el esquema Cilindro-Cabeza-Sector, a primera vista se ven 4 brazos, uno para cada plato. En realidad, cada uno de los brazos es doble, y contiene dos cabezas: una para leer la cara superior del plato, y otra para leer la cara inferior. Por tanto, hay ocho cabezas para leer cuatro platos, aunque por cuestiones comerciales, no siempre se usan todas las caras de los discos y existen discos duros con un número impar de cabezas, o con cabezas deshabilitadas. Los cabezales de lectura/escritura no tocan el disco, sino que pasan muy cerca (hasta a 3 nanómetros), debido a una finísima película de aire que se forma entre los cabezales y los platos cuando los discos giran (algunos discos incluyen un sistema que impide que los cabezales pasen por encima de los platos hasta que alcancen una velocidad de giro que garantice la formación de esta película). Si alguna de las cabezas llega a tocar una superficie de un plato, causaría muchos daños en él, rayándolo gravemente, debido a lo rápido que giran los platos (uno de 7200 revoluciones por minuto se mueve a 129 km/h en el borde de un disco de 3,5 pulgadas).

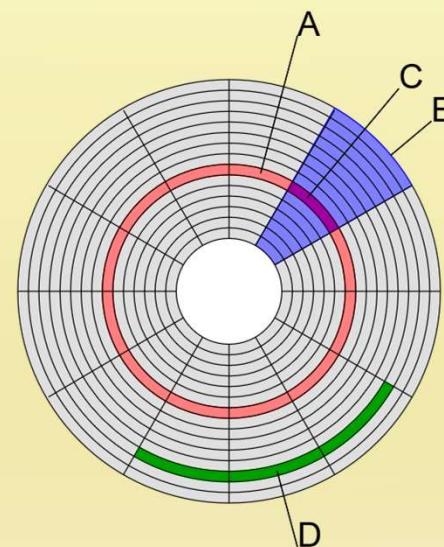
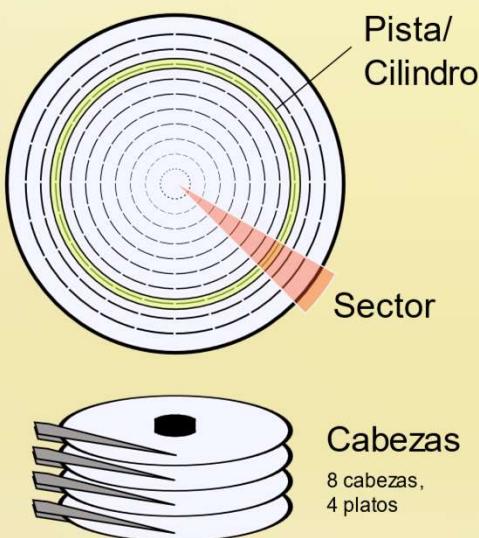
Direccionamiento

Hay varios conceptos para referirse a zonas del disco:

- Plato: cada uno de los discos que hay dentro de la unidad de disco duro.
- Cara: cada uno de los dos lados de un plato.
- Pista: una circunferencia dentro de una cara; la pista cero (0) está en el borde exterior.
- Cilindro: conjunto de varias pistas; son todas las circunferencias que están alineadas verticalmente (una de cada cara).
- Cabezal: número de cabeza o cabezal por cada cara.
- Sector : cada una de las divisiones de una pista. El tamaño del sector no es fijo, siendo el estándar actual 512 bytes, aunque la asociación IDEMA13 ha creado un comité que impulsa llevarlo a 4 KiB. Antiguamente el número de sectores por pista era fijo, lo cual desaprovechaba el espacio significativamente, ya que en las pistas exteriores pueden almacenarse más sectores que en las interiores. Así, apareció la tecnología grabación de bits por zonas (Zone Bit Recording, ZBR) que aumenta el número de sectores en las pistas exteriores, y utiliza más eficientemente el disco duro. Así las pistas se agrupan en zonas de pistas de igual cantidad de sectores. Cuanto más lejos del centro de cada plato se encuentra una zona, esta contiene una mayor cantidad de sectores en sus pistas. Además mediante ZBR, cuando se leen sectores de cilindros más externos la tasa de transferencia de bits por segundo es mayor; por tener la misma velocidad angular que cilindros internos pero mayor cantidad de sectores.

- Sector geométrico: son los sectores contiguos pero de pistas diferentes.
- Clúster: es un conjunto contiguo de sectores.

El primer sistema de direccionamiento que se usó fue el Cilindro-Cabezal-Sector (Cylinder-Head-Sector, CHS), ya que con estos tres valores se puede situar un dato cualquiera del disco. Más adelante se creó otro sistema más sencillo, que actualmente se usa: direccionamiento de bloques lógicos (Logical Block Addressing, LBA), que consiste en dividir el disco entero en sectores y asignar a cada uno un único número.



Estructura de disco magnético:
 A es una pista del disco (roja)
 B es un sector geométrico (azul)
 C es un sector de una pista (magenta)
 D es un grupo de sectores o clúster (verde).

Factor de forma

La primera unidad de disco duro de IBM, la IBM 350, usaba una pila de cincuenta platos de 24 pulgadas, almacenaba 3,75 MB de datos (aproximadamente el tamaño de una imagen digital moderna) y tenía un tamaño comparable a dos refrigeradores grandes. En 1962, IBM presentó el IBM 1311, que usaba seis platos de 14 pulgadas (tamaño nominal) en un paquete extraíble y era aproximadamente del tamaño de una lavadora. Este se convirtió en un tamaño de plato estándar durante muchos años, utilizado también por otros fabricantes. El IBM 2314 usó platos del mismo tamaño en un paquete de once e introdujo el diseño de "unidad en un cajón". a veces llamado el "horno de pizza", aunque el "cajón" no era la unidad completa. En la década de 1970, los discos duros se ofrecían en gabinetes independientes de diferentes dimensiones que contenían de uno a cuatro discos duros.

A partir de finales de la década de 1960, se ofrecieron unidades que encajaban completamente en un chasis que se montaba en un bastidor de 19 pulgadas. Los RK05 y RL01 de Digital Equipment Corporation fueron los primeros ejemplos que utilizaron platos individuales de 14 pulgadas en paquetes extraíbles, la unidad completa encajaba en un espacio de rack de 10,5 pulgadas de alto (seis unidades de rack). A mediados y finales de la década de 1980, el Fujitsu Eagle de tamaño similar, que usaba (casualmente) platos de 10,5 pulgadas, era un producto popular.



Seis unidades de disco duro con carcasa abierta mostrando platos y cabezales; 8, $5\frac{1}{4}$, $3\frac{1}{2}$, $2\frac{1}{2}$, $1\frac{1}{8}$ y 1 pulgadas de diámetro de los discos que representan.



Una unidad de disco duro de 2,5 pulgadas (63,5 mm) de 6495 MB en comparación con una unidad de disco duro anterior de 5,25 pulgadas de altura completa y 110 MB

Características de un disco duro.

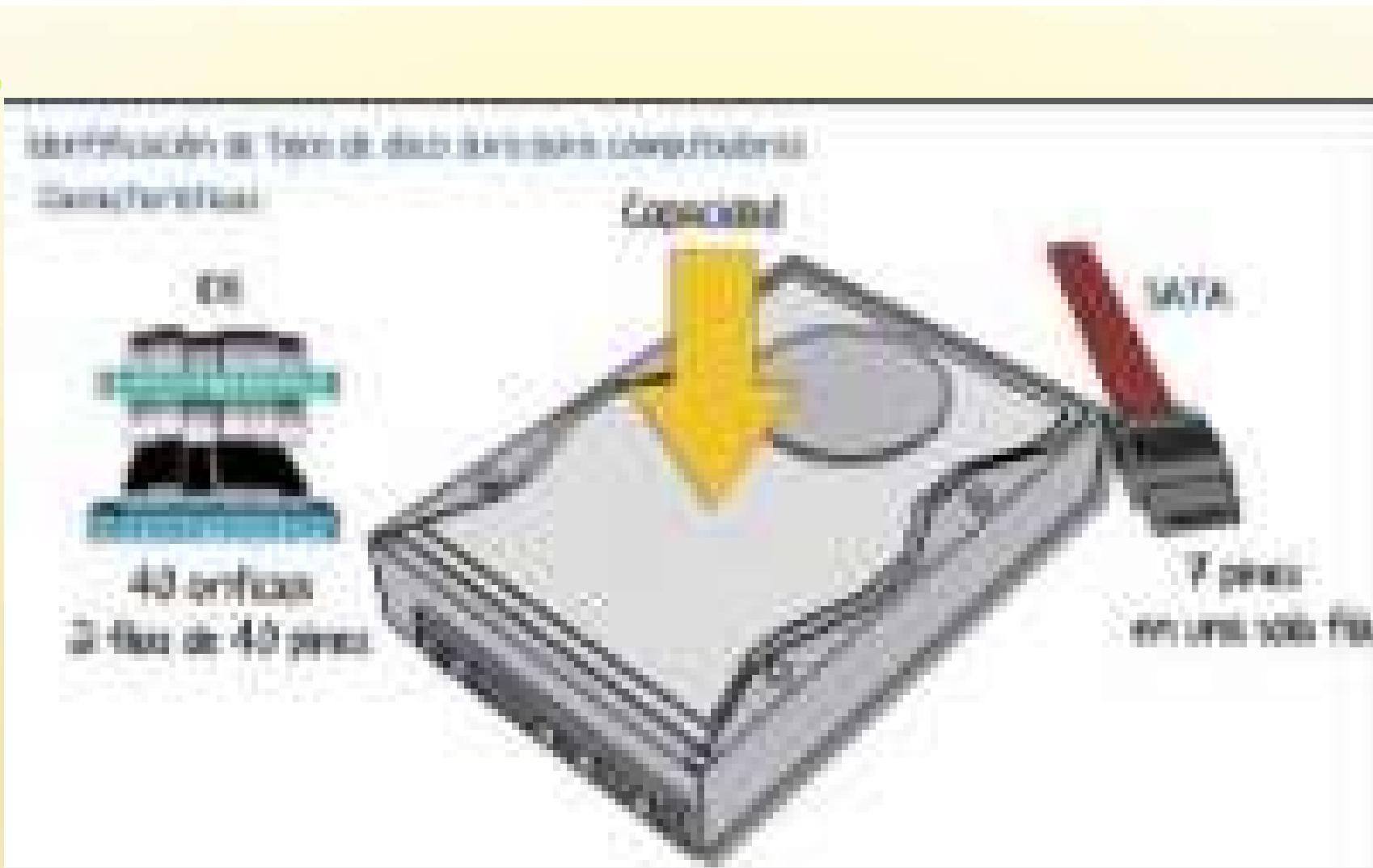
Las características que se deben tener en cuenta en un disco duro son:

- **Tiempo medio de acceso:** tiempo medio que tarda la aguja en situarse en la pista y el sector deseado; es la suma del Tiempo medio de búsqueda (situarse en la pista), Tiempo de lectura/escritura y la Latencia media (situarse en el sector).
- **Tiempo medio de búsqueda:** tiempo medio que tarda la aguja en situarse en la pista deseada; es la mitad del tiempo empleado por la aguja en ir desde la pista más periférica hasta la más central del disco.
- **Tiempo de lectura/escritura:** tiempo medio que tarda el disco en leer o escribir nueva información: Depende de la cantidad de información que se quiere leer o escribir, el tamaño de bloque, el número de cabezales, el tiempo por vuelta y la cantidad de sectores por pista.
- **Latencia media:** tiempo medio que tarda la aguja en situarse en el sector deseado; es la mitad del tiempo empleado en una rotación completa del disco.
- **Tasa de transferencia:** velocidad a la que puede transferir la información a la computadora una vez que la aguja está situada en la pista y sector correctos. Puede ser velocidad sostenida o de pico.

- **Velocidad de rotación:** Es la velocidad a la que gira el disco duro, más exactamente, la velocidad a la que giran el/los platos del disco, que es donde se almacenan magnéticamente los datos. La regla es: a mayor velocidad de rotación, más alta será la transferencia de datos, pero también mayor será el ruido y mayor será el calor generado por el disco duro. Se mide en número revoluciones por minuto (RPM). No debe comprarse un disco duro IDE de menos de 5400 RPM (ya hay discos IDE de 7200 RPM), a menos que te lo den a un muy buen precio, ni un disco SCSI de menos de 7200 RPM (los hay de 10.000 RPM). Una velocidad de 5400 RPM permitirá una transferencia entre 80MB y 110MB por segundo con los datos que están en la parte exterior del cilindro o plato, algo menos en el interior. revoluciones por minuto de los platos. A mayor velocidad de rotación, menor latencia media.

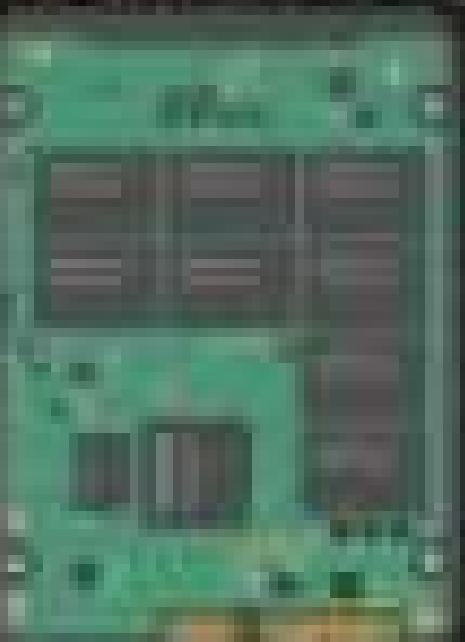
Otras características son:

- **Caché de pista:** es una memoria tipo flash dentro del disco duro.
- **Interfaz:** medio de comunicación entre el disco duro y el ordenador. Puede ser IDE/ATA, SCSI, SATA, USB, Firewire, Serial Attached SCSI
- **Landz:** zona sobre las que aparcan las cabezas una vez se apaga la computadora.



<https://youtu.be/pAsEMGiY8GY>

CUÁLES ES LA DIFERENCIA?



SSD

VS



HDD

<https://youtu.be/CxUVjbO05ms>

SISTEMAS OPERATIVOS 2

LABORATORIO

DISPOSITIVOS DE ENTRADA Y SALIDA



Los dispositivos de entrada y salida o unidades de entrada/salida son los equipos físicos conectados a la computadora. Estos dispositivos permiten comunicar información entre el usuario y la computadora o manejar un soporte de información.

Son también llamados periféricos de computadora o periféricos de entrada y salida porque están separados de la unidad central de procesamiento.

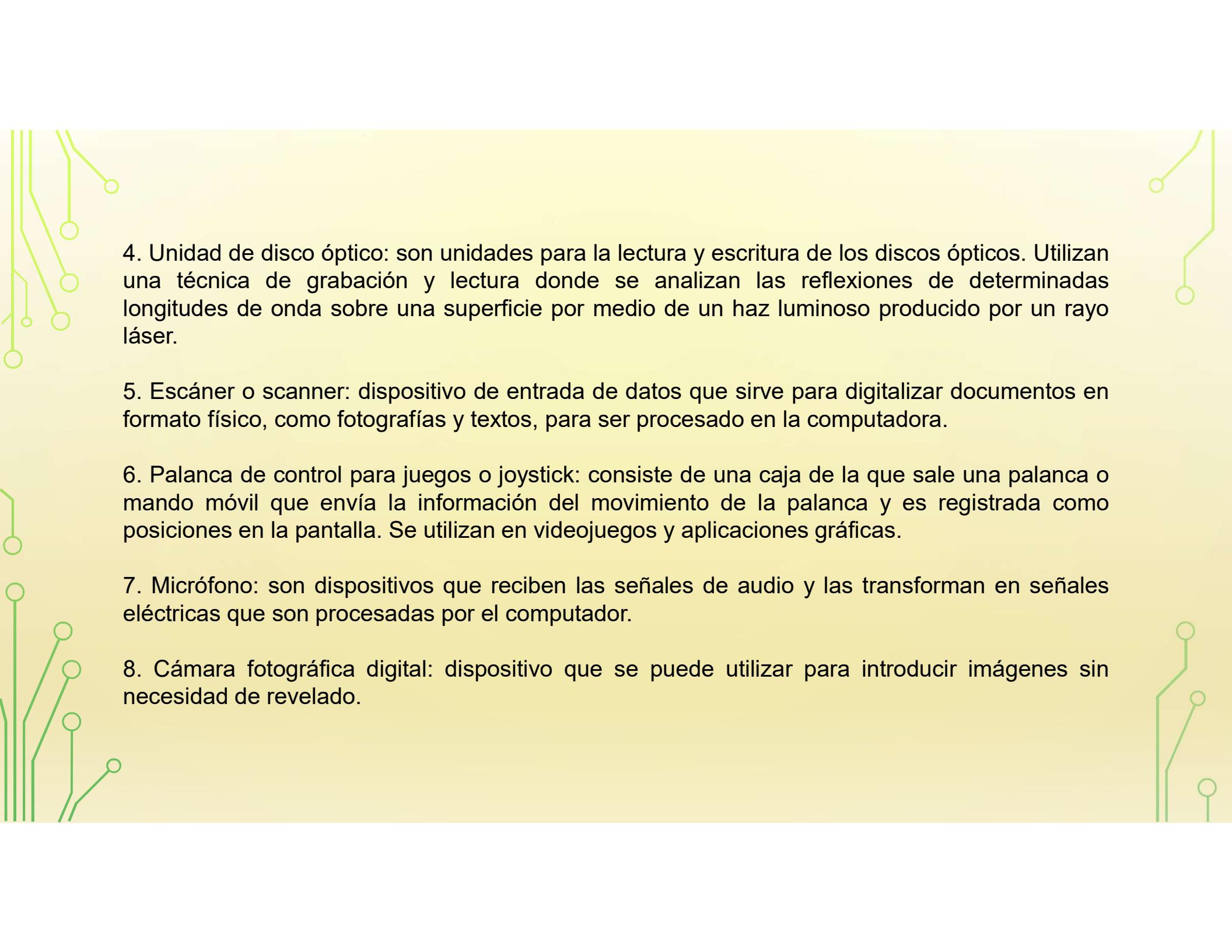
Los dispositivos de entrada/salida se clasifican por las funciones que pueden realizar en dispositivos de entrada, dispositivos de salida y dispositivos de entrada y salida.

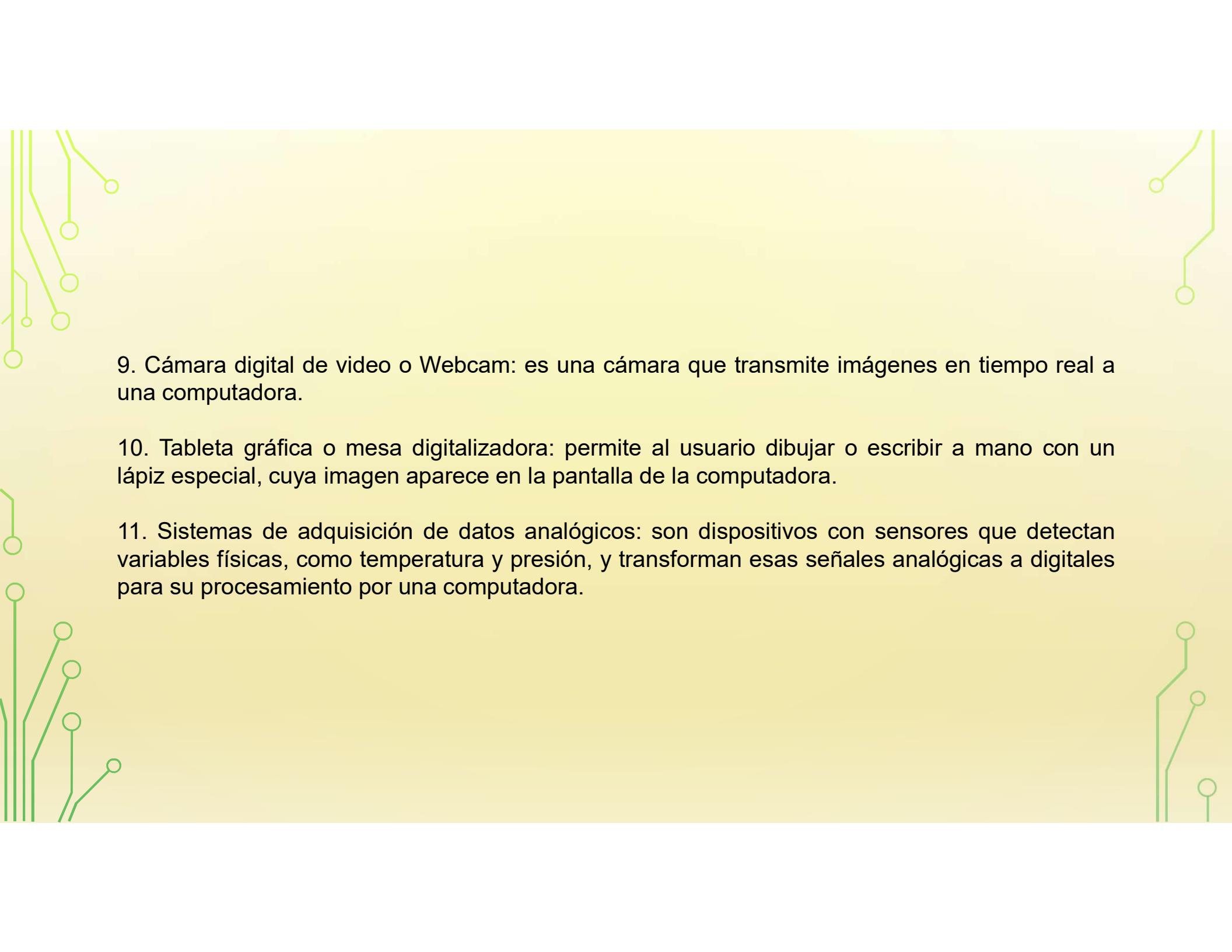
Dispositivos de entrada

Los dispositivos de entrada son aquellos equipos encargados de introducir datos en la memoria central de la computadora para su tratamiento. A través de ellos se transforma la información de entrada en señales eléctricas.

Ejemplos de dispositivos de entrada podemos mencionar los siguientes:

1. Teclado: permite la comunicación entre el usuario y la computadora. Dispone de un conjunto de teclas agrupadas en cuatro bloques denominados alfabetico, numérico, de control y teclas de función.
2. Ratón o mouse de computadora: es una unidad de entrada constituida por una pequeña cajetilla con controles que se adapta a la mano y permite el movimiento del cursor en la pantalla. Existen ratones mecánicos, ópticos y opto-mecánicos.
3. Lectora de código de barras: se utiliza un lápiz óptico o un haz luminoso formado por un rayo láser capaz de realizar una imagen tridimensional que permite leer el código en cualquier posición.

- 
4. Unidad de disco óptico: son unidades para la lectura y escritura de los discos ópticos. Utilizan una técnica de grabación y lectura donde se analizan las reflexiones de determinadas longitudes de onda sobre una superficie por medio de un haz luminoso producido por un rayo láser.
 5. Escáner o scanner: dispositivo de entrada de datos que sirve para digitalizar documentos en formato físico, como fotografías y textos, para ser procesado en la computadora.
 6. Palanca de control para juegos o joystick: consiste de una caja de la que sale una palanca o mando móvil que envía la información del movimiento de la palanca y es registrada como posiciones en la pantalla. Se utilizan en videojuegos y aplicaciones gráficas.
 7. Micrófono: son dispositivos que reciben las señales de audio y las transforman en señales eléctricas que son procesadas por el computador.
 8. Cámara fotográfica digital: dispositivo que se puede utilizar para introducir imágenes sin necesidad de revelado.

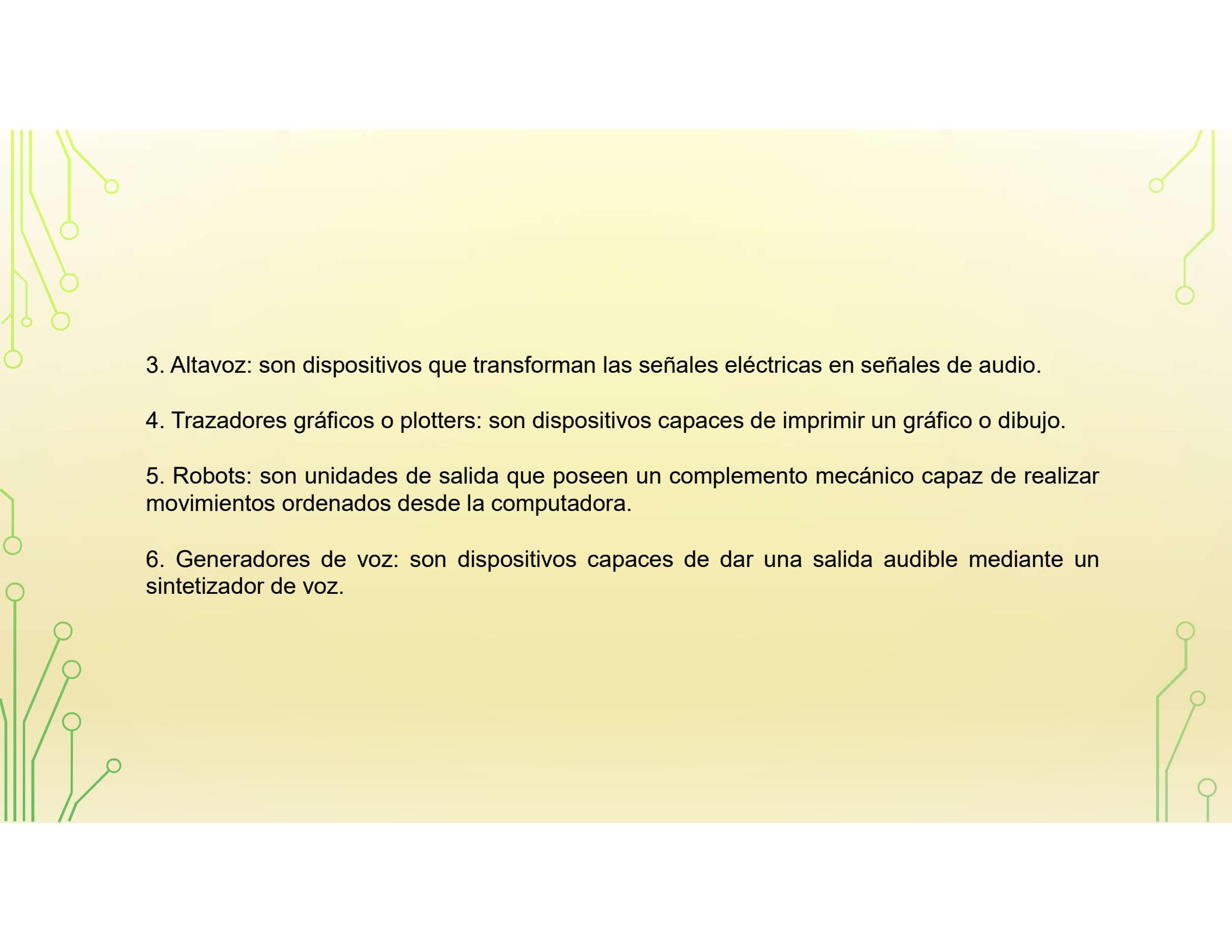
- 
- The background of the slide features a decorative pattern of thin green lines and small white circles. These lines form various shapes, including vertical columns, diagonal paths, and small clusters, primarily located in the corners and along the edges of the slide.
9. Cámara digital de video o Webcam: es una cámara que transmite imágenes en tiempo real a una computadora.
 10. Tableta gráfica o mesa digitalizadora: permite al usuario dibujar o escribir a mano con un lápiz especial, cuya imagen aparece en la pantalla de la computadora.
 11. Sistemas de adquisición de datos analógicos: son dispositivos con sensores que detectan variables físicas, como temperatura y presión, y transforman esas señales analógicas a digitales para su procesamiento por una computadora.

Dispositivos de salida

Los dispositivos de salida son los equipos que presentan la información al usuario de forma comprensible, ya sea a través de imágenes, texto, sonidos o táctil. Estos realizan la función de extraer datos de la memoria central hacia el exterior.

Como ejemplos de dispositivos de salida encontramos:

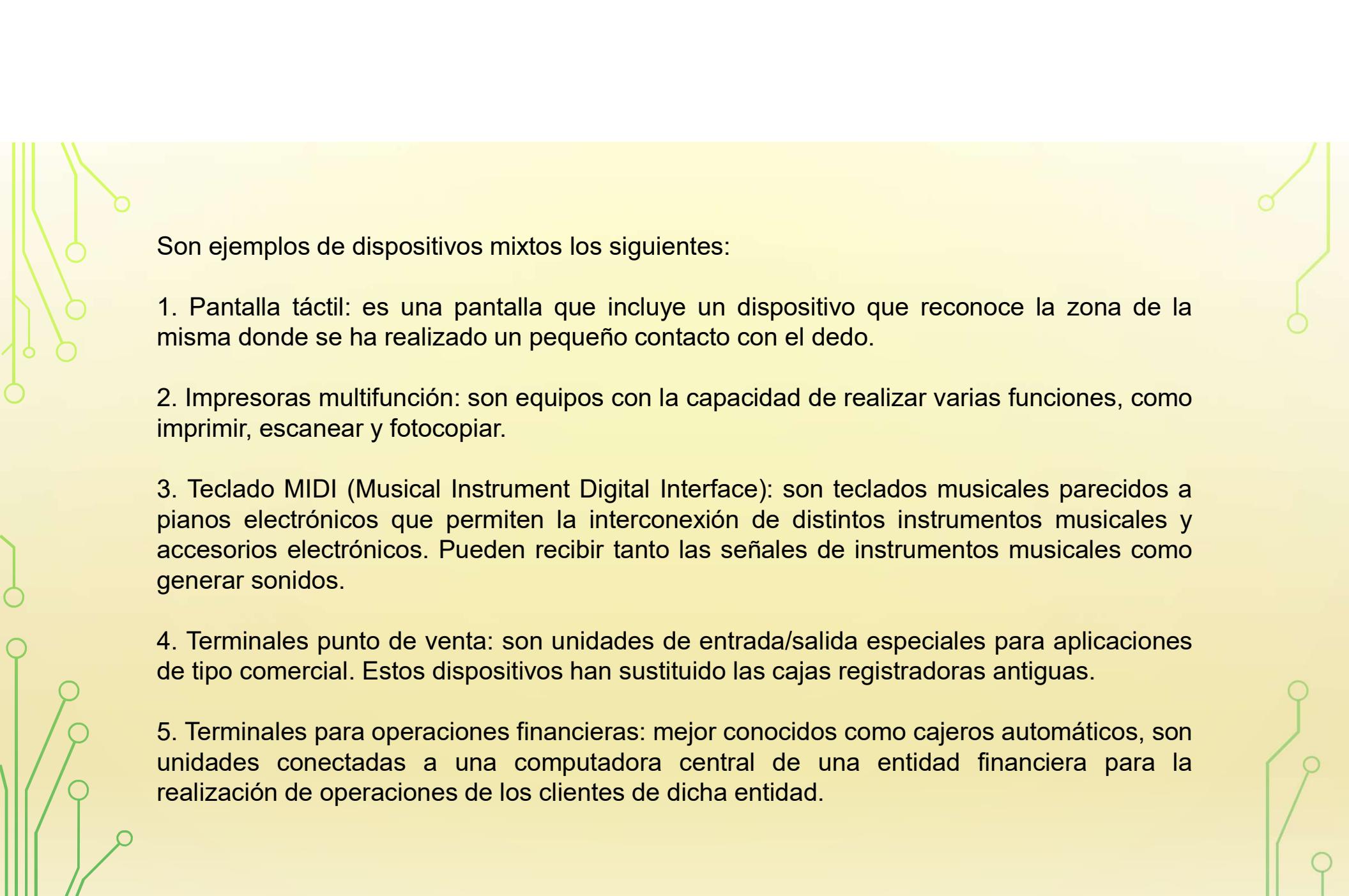
1. Pantalla o display: consiste en un sistema de representación mediante configuraciones de puntos luminosos denominados píxeles. La resolución de pantalla es el número de píxeles que posee. En las computadoras de escritorio o desktop se le conoce como monitor.
2. Impresoras: son unidades de salida de datos soportados en papel. Existen diversos tipos de impresoras, entre ellas las impresoras térmicas, electrostáticas, de tinta y láser.

- 
- The background of the slide features abstract green line art and small circles. On the left side, there are several vertical lines of varying heights, some ending in small circles. In the center, there are more complex, branching line structures with circles at their ends. On the right side, there are additional vertical lines and a few isolated circles. The overall aesthetic is minimalist and modern.
3. Altavoz: son dispositivos que transforman las señales eléctricas en señales de audio.
 4. Trazadores gráficos o plotters: son dispositivos capaces de imprimir un gráfico o dibujo.
 5. Robots: son unidades de salida que poseen un complemento mecánico capaz de realizar movimientos ordenados desde la computadora.
 6. Generadores de voz: son dispositivos capaces de dar una salida audible mediante un sintetizador de voz.

Dispositivos mixtos.

Los dispositivos de entrada y salida o dispositivos mixtos permiten la introducción y extracción de datos en la memoria central.

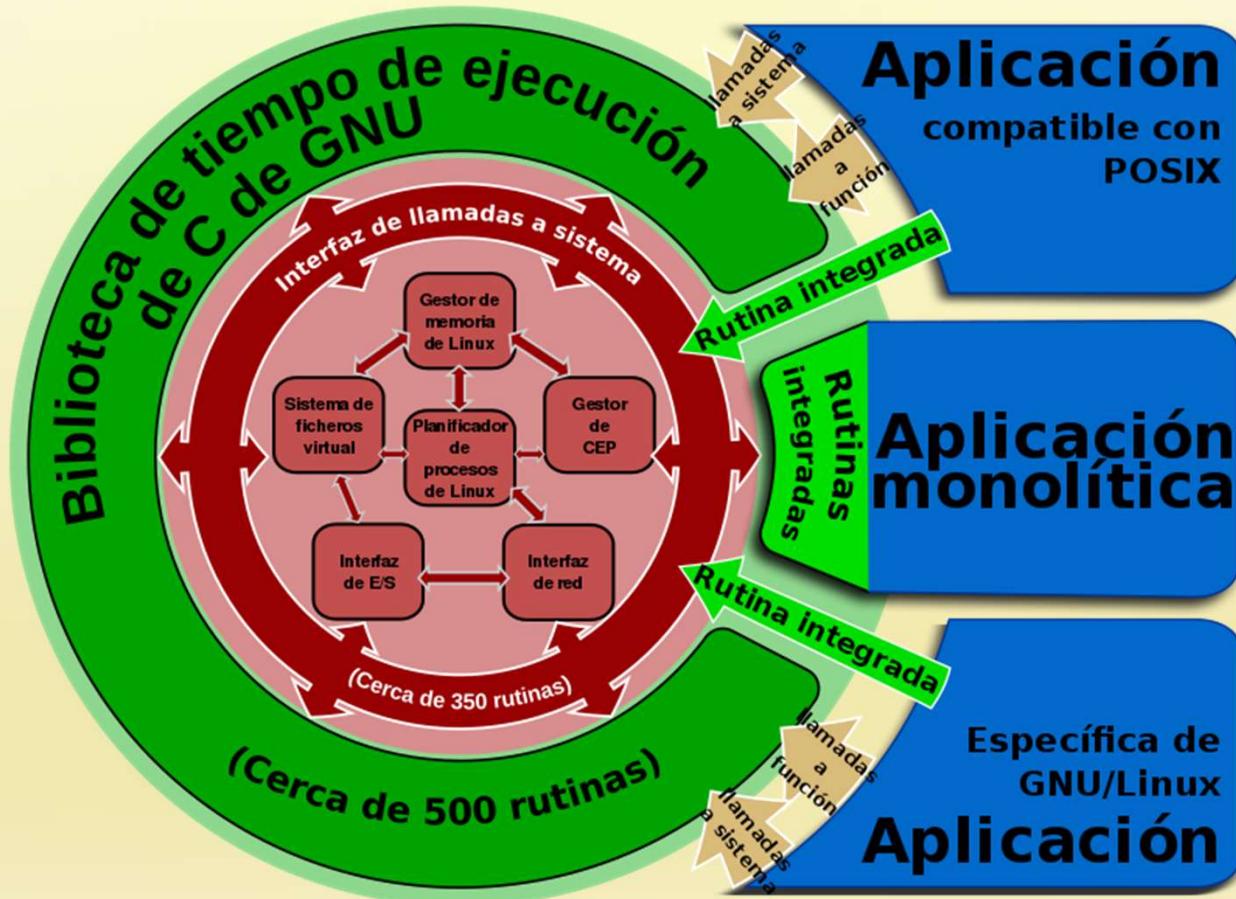




Son ejemplos de dispositivos mixtos los siguientes:

1. Pantalla táctil: es una pantalla que incluye un dispositivo que reconoce la zona de la misma donde se ha realizado un pequeño contacto con el dedo.
2. Impresoras multifunción: son equipos con la capacidad de realizar varias funciones, como imprimir, escanear y fotocopiar.
3. Teclado MIDI (Musical Instrument Digital Interface): son teclados musicales parecidos a pianos electrónicos que permiten la interconexión de distintos instrumentos musicales y accesorios electrónicos. Pueden recibir tanto las señales de instrumentos musicales como generar sonidos.
4. Terminales punto de venta: son unidades de entrada/salida especiales para aplicaciones de tipo comercial. Estos dispositivos han sustituido las cajas registradoras antiguas.
5. Terminales para operaciones financieras: mejor conocidos como cajeros automáticos, son unidades conectadas a una computadora central de una entidad financiera para la realización de operaciones de los clientes de dicha entidad.

System calls



¿qué son y para qué se emplean?

En el mundo actual, los sistemas operativos deben ofrecer a los usuarios no solo el mayor confort posible, sino también la máxima estabilidad y seguridad. Para esto, los desarrolladores de sistemas como Linux o Windows se ocupan, entre otras cosas, de minimizar el riesgo de posibles complicaciones en el sistema causados por una negligencia personal involuntaria o por ataques dirigidos desde el exterior. Una de las medidas más importantes tomadas para este propósito es la separación estricta del núcleo del sistema operativo y los programas de aplicación o procesos de usuario. Esto quiere decir que los programas y procesos que no pertenecen al sistema no tienen acceso directo a la CPU y a la memoria, sino que dependen de las llamadas al sistema.

¿Qué es una llamada al sistema (syscall)?

Una llamada al sistema o system call es un método utilizado por los programas de aplicación para comunicarse con el núcleo del sistema. En los sistemas operativos modernos, esto es necesario cuando una aplicación o proceso de usuario necesita transmitir o leer información del hardware, de otros procesos o del propio núcleo. De este modo, la llamada es el punto de enlace entre el modo de usuario y el modo de núcleo, los dos modos cruciales de privilegio y seguridad para el procesamiento de las instrucciones de la CPU en los sistemas informáticos.

Antes de que la llamada al sistema termine de procesarse y se transmitan o reciban los datos correspondientes, el núcleo del sistema toma el control del programa o proceso. La ejecución se interrumpe durante este período. Una vez realizada la acción solicitada por una llamada al sistema, el núcleo renuncia al control y el código continúa desde el punto en el que se inició previamente la llamada al sistema.

¿Para qué se necesitan las system calls?

Las system calls están estrechamente ligadas al modelo moderno de sistema operativo con modo usuario y modo núcleo, que se introdujo en respuesta al creciente número de procesos que se ejecutan simultáneamente en la memoria principal (RAM) de las computadoras. De este modo, cada proceso individual tiene sus propios datos con derechos de acceso especiales y, solo si se distribuyen los recursos correctamente, el sistema y los programas pueden ejecutarse según lo esperado.

El modo núcleo de mayor privilegio es aquí la instancia de control decisiva, porque, como mencionamos, en este no solo se ejecutan todos los servicios y procesos del sistema en sí, sino también las acciones críticas del sistema de los programas de aplicación que en el modo usuario están bloqueadas. Para esto, es necesario que el programa realice una llamada al sistema, que en la mayoría de los casos es simplemente una cuestión de acceso a la potencia de cálculo (CPU) o a las estructuras de memoria (memoria de trabajo y espacio de disco duro). Por ejemplo, si una aplicación requiere más potencia de cálculo o espacio de almacenamiento, o si se solicita un archivo no específico de la aplicación (abrir, leer, editar, etc.), es necesario realizar una system call.

¿Qué tipos de llamadas al sistema hay?

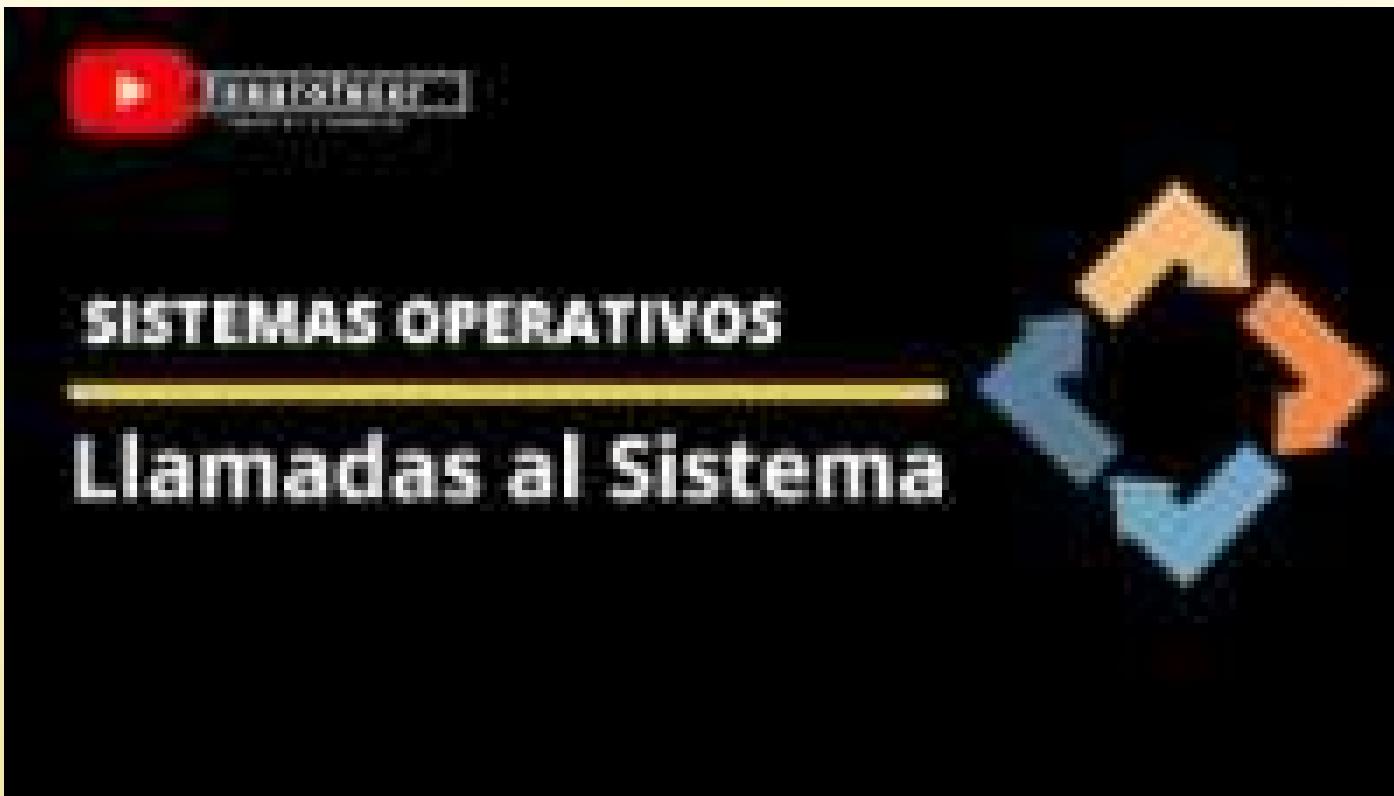
En principio, como se dijo arriba, todas las llamadas al sistema se pueden utilizar como unidades de control para la comunicación entre los procesos de aplicación y el sistema operativo o el hardware. Además, las llamadas al sistema establecidas también se pueden clasificar en diferentes categorías, por lo que se han establecido en particular estos cinco tipos:

- Control de procesos: todos los procesos de un sistema informático deben controlarse para que en cualquier momento se puedan detener u otros procesos los puedan dirigir. Para esto, las llamadas al sistema de esta categoría supervisan, por ejemplo, el inicio o la ejecución o la detención/terminación de procesos.
- Gestión de archivos: los programas de aplicación requieren este tipo de llamadas al sistema para acceder a las operaciones típicas con archivos. Estos métodos de manipulación de archivos incluyen la creación, eliminación, apertura, cierre, escritura y lectura (create, delete, open, close, write y read).

- Gestión de dispositivos: la categoría “Gestión de dispositivos” incluye todas las llamadas al sistema que sirven para solicitar o administrar los recursos de hardware necesarios, como la potencia de computación o el espacio de almacenamiento.
- Gestión de la información: los procesos tienen mucha información asociada, y la puntualidad y la integridad son muy importantes. Para intercambiar o solicitar información, los programas de aplicación utilizan llamadas al sistema para la gestión o el mantenimiento de la información.
- Comunicación entre procesos: solo se puede garantizar una interacción fluida entre el sistema operativo y los distintos programas de aplicación si los procesos individuales están coordinados. Con este fin, es indispensable la comunicación a través de las correspondientes llamadas al sistema.

Windows y Linux: visión general de las llamadas al sistema

La medida en que los tipos de llamadas al sistema enumerados se pueden implementar o realizar depende principalmente del hardware y de la arquitectura del sistema utilizados, pero también del sistema operativo. En Linux, por ejemplo, las llamadas al sistema se almacenan directamente en el núcleo de Linux en la “Tabla de llamadas al sistema”. Cada entrada de esta tabla tiene asignados un número único y una función específica que se ejecutará en el modo núcleo. Para ejecutar cualquier llamada al sistema Linux, se carga el número en la memoria de la CPU y después se carga mediante una interrupción de software 128 (llamada a una subfunción del sistema operativo que interrumpe la ejecución del programa en el modo de usuario).



<https://youtu.be/tffvKcs83jE>

La funcionalidad es similar en los sistemas operativos Windows, pero aquí una llamada al sistema siempre se convierte primero internamente: una función de biblioteca de la API de Windows (abreviatura WinAPI) se convierte automáticamente en una llamada al sistema que el sistema operativo puede leer, incluyendo un número único que hace referencia a la función deseada en el modo núcleo.

En la siguiente tabla se enumeran algunos ejemplos concretos de llamadas al sistema de Windows y Linux:

Tipo de system call	Función	Linux	Windows
Control del proceso	Crear proceso	fork()	CreateProcess()
Control del proceso	Terminar proceso	exit()	ExitProcess()
Gestión de archivos	Crear/abrir archivo	open()	CreateFile()
Gestión de archivos	Leer archivo	read()	ReadFile()
Gestión de archivos	Editar archivo	write()	WriteFile()
Gestión de archivos	Cerrar archivo	close()	CloseHandle()
Gestión de dispositivos	Abrir dispositivo	read()	ReadConsole()
Gestión de dispositivos	Cerrar dispositivo	close()	CloseConsole()
Gestión de la información	Definición de un intervalo de tiempo específico	alarm()	SetTimer()
Gestión de la información	Pausa (por ejemplo, de un proceso)	sleep()	Sleep()
Comunicación	Crear Pipe (memoria intermedia para el flujo de datos entre dos procesos)	pipe()	CreatePipe()
Comunicación	Creación de una memoria compartida (Shared memory)	shmget()	CreateFileMapping()

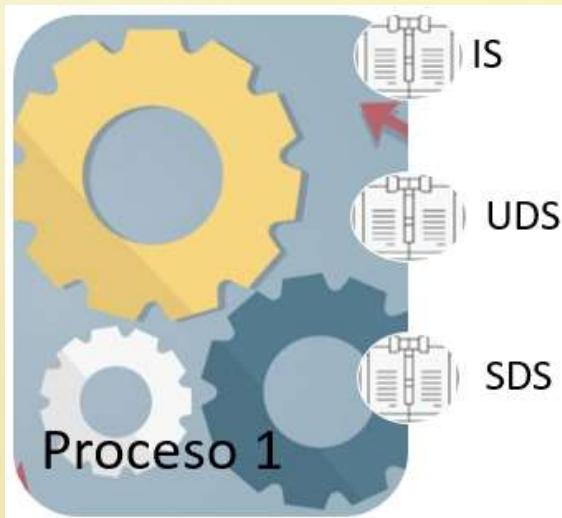
Llamadas al sistema para gestión de procesos.

El sistema operativo UNIX dispone de un conjunto de llamadas al sistema que definen una poderosa interfaz para la programación de aplicaciones (API) que involucren múltiples procesos; abriendo las puertas a la programación concurrente. Este interfaz suministra al desarrollador de software herramientas tanto para la creación, sincronización y comunicación de nuevos procesos, como la capacidad de ejecutar nuevos programas.

Entre los aspectos más destacados de la gestión de procesos en UNIX se encuentra la forma en que éstos se crean y cómo se ejecutan nuevos programas. Aunque se mostrarán las llamadas al sistema correspondientes más adelante en esta práctica, es conveniente presentar una visión inicial conjunta que permita entender mejor la forma en que estas llamadas se utilizan.

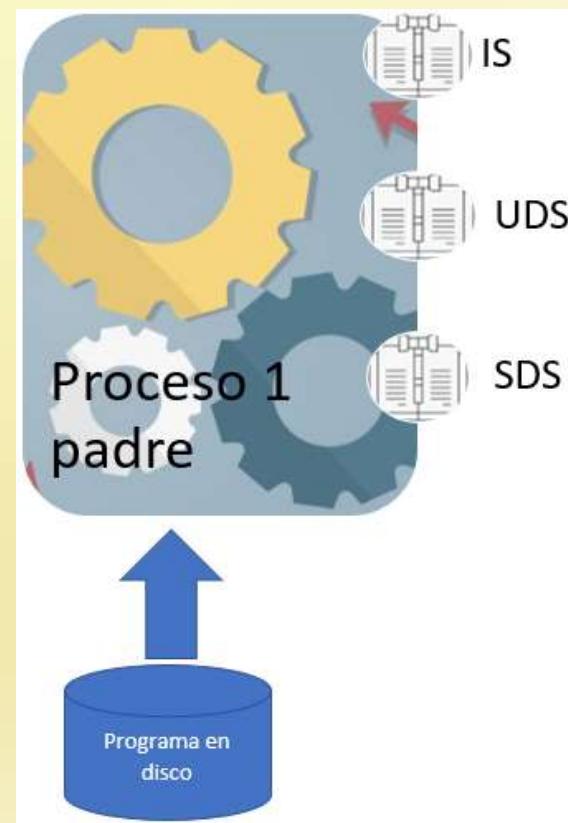
El kernel crea un nuevo proceso, proceso hijo, realizando una copia (clonación) del proceso que realiza la llamada al sistema fork (proceso padre). Así, salvo el PID y el PPID los dos procesos serán inicialmente idénticos. De esta forma los nuevos procesos obtienen una copia de los recursos del padre (heredan el entorno).

El proceso 1 realiza una llamada al sistema (exec) para ejecutar las instrucciones de un nuevo programa



IS: Segmento de instrucciones.
UDS: Segmento de datos de usuario.
SDS: Segmento de datos del sistema.

El Kernel reinicializa los segmentos de instrucciones y de datos de usuario con los datos del programa en disco especificado en la llamada. De esta forma el mismo proceso puede ejecutar un programa diferente.



Sin embargo no se ejecuta ningún nuevo programa, para conseguir esto, uno de los procesos ha de realizar otra llamada al sistema, exec, para reinicializar (recubrir) sus segmentos de datos de usuario e instrucciones a partir de un programa en disco. En este caso no aparece ningún proceso nuevo.

Cuando un proceso termina (muere), el sistema operativo lo elimina recuperando sus recursos para que puedan ser usados por otros.

Creación de procesos. System call fork

```
#include <unistd.h>
int fork ()
```

fork crea un nuevo proceso; pero no lo inicia desde un nuevo programa. Los segmentos de datos de usuario y del sistema y el segmento de instrucciones del nuevo proceso (hijo) son copias casi exactas del proceso que realizó la llamada (padre). El valor de retorno de fork es:

Proceso hijo	0
Proceso Padre	PID del proceso hijo
Fork fracasa y no puede crear un nuevo proceso	-1

El proceso hijo hereda la mayoría de los atributos del proceso padre, ya que se copian de su segmento de datos del sistema. Sólo algunos atributos difieren entre ambos:

- **PID**
- **archivos**: El hijo obtiene una copia de la tabla de descriptores de archivo del proceso padre, con lo que comparten el puntero del archivo. Si uno de los procesos cambia ese puntero (realiza una operación de E/S) la siguiente operación de E/S realizada por el otro proceso se hará a partir de la posición indicada por el puntero modificado por el primer proceso. Sin embargo, al existir dos tablas de descriptores de archivos si un proceso cierra su descriptor, el otro no se ve afectado.

Terminación de procesos. System call exit y wait

```
#include <unistd.h>
void exit (int status)
```

exit finaliza al proceso que la llamó, con un código de estado igual al byte menos significativo del parámetro entero status. Todos los descriptores de archivo abiertos son cerrados y sus buffers sincronizados. Si hay procesos hijo cuando el padre ejecuta un exit, el PPID de los hijos se cambia a 1 (proceso init). Es la única llamada al sistema que nunca retorna.

El valor del parámetro status se utiliza para comunicar al proceso padre la forma en que el proceso hijo termina. Por convenio, este valor suele ser 0 si el proceso termina correctamente y cualquier otro valor en caso de terminación anormal. El proceso padre puede obtener este valor a través de la llamada al sistema wait.

```
#include <unistd.h>
int wait (int *statusp)
```



Si hay varios procesos hijos, wait espera hasta que uno de ellos termina. No es posible especificar por qué hijo se espera. wait retorna el PID del hijo que termina (o -1 si no se crearon hijos o si ya no hay hijos por los que esperar) y almacena el código del estado de finalización del proceso hijo (parámetro status en su llamada al sistema exit) en la dirección apuntada por el parámetro statusp.

Un proceso puede terminar en un momento en el que su padre no le esté esperando. Como el kernel debe asegurar que el padre pueda esperar por cada proceso, los procesos hijos por los que el padre no espera se convierten en procesos zombi (se descartan su segmentos pero siguen ocupando una entrada en la tabla de procesos del kernel). Cuando el padre realiza una llamada wait, el proceso hijo es eliminado de la tabla de procesos.

No es obligatorio que todo proceso padre espere a sus hijos.

Un proceso puede terminar por:

Causa de terminación	Contenido de *statusp
Llamada al sistema exit	byte más a la derecha = 0 byte de la izquierda contiene el valor del parámetro status de exit
Recibe una señal	En los 7 bits más a la derecha se almacena el número de señal que terminó con el proceso. Si el 8º bit más a la derecha está a 1 el proceso fue detenido por el kernel y se generó un volcado del proceso en un archivo core.
Caída del sistema (p.e: pérdida de la alimentación del equipo.)	Todos los procesos desaparecen bruscamente. No hay nada que devolver.

Ejecución de nuevos programas. *System call exec.*

La llamada al sistema exec permite remplazar los segmentos de instrucciones y de datos de usuario por otros nuevos a partir de un archivo ejecutable en disco, con lo que se consigue que un proceso deje de ejecutar instrucciones de un programa y comience a ejecutar instrucciones de un nuevo programa. exec no crea ningún proceso nuevo.

Como el proceso continua activo su segmento de datos del sistema apenas es perturbado, la mayoría de sus atributos permanecen inalterados. En particular, los descriptores de archivos abiertos permanecen abiertos después de un exec. Esto es importante puesto que algunas funciones de la librería C (como printf) utilizan buffers internos para aumentar el rendimiento de la E/S; si un proceso realiza un exec y no se han volcado (sincronizado) antes los buffers internos, los datos de estos buffers se perderán. Por ello es habitual cerrar los descriptores abiertos antes de realizar una llamada al sistema exec.

El resultado de la llamada al sistema exec sólo está disponible si la llamada fracasa (-1).

Descripción de los argumentos:

Nombre del argumento	Descripción
path, file	nombre del nuevo programa a ejecutar con su trayectoria. Ejemplo: "/bin/cp" Las versiones de exec que utilizan file en lugar de path utilizan la variable de entorno PATH para localizar el programa a ejecutar, por lo que en esos casos no es necesario especificar la trayectoria al programa si este se encuentra en alguno de los directorios especificados en PATH
arg0	primer argumento del programa. Por convención suele asignarse el nombre del programa sin la trayectoria. Ejemplo: "cp"
arg1 ... argN null	Conjunto de parámetros que recibe el programa para su ejecución. Ejemplo: toupper.c seguridad. El parámetro formal null debe ser remplazado por el parámetro real NULL.
argv	Matriz de punteros a cadenas de caracteres. Estas cadenas de caracteres constituyen la lista de argumentos disponibles para el nuevo programa. El último de los punteros debe ser NULL. Por convención, este array debe contener al menos un elemento (nombre del programa).
envp	Matriz de punteros a cadenas de caracteres. Estas cadenas de caracteres constituyen el entorno de ejecución del nuevo programa.

A continuación se presenta una tabla comparativa de las diferentes versiones de exec:

Versión	Formato argumentos	Paso del entorno	¿Utiliza PATH?
execl	lista	automático	no
execv	array	automático	no
execle	lista	manual	no
execve	array	manual	no
execlp	lista	automático	sí
execvp	array	automático	sí

El nuevo programa puede acceder a los argumentos a través de argc y argv de su función main.

Envío de señales entre procesos

Se define una señal, como un mensaje enviado a un proceso determinado. Este mensaje no es más que un número entero. Un proceso cuando recibe una señal puede optar por tres posibles alternativas para procesarla:

- Ignorar la señal recibida.
- Ejecutar una acción por defecto.
- Ejecutar una acción determinada, especificada por el usuario en el propio proceso.

Algunas de las señales más comunes, utilizadas en UNIX son:

- SIGINT => Señal de interrupción: Enviada desde el terminal (teclado) por medio de Ctrl+c.
- SIGQUIT => Enviada desde el terminal por el carácter quit (Ctrl+\). El proceso activo termina produciendo un archivo core.
- SIGKILL => Mata un proceso.

El resto de señales están definidas en el archivo de cabecera /usr/include/signal.h

Prioridades de las señales.

Todas las señales tienen la misma prioridad. A diferencia de las interrupciones Hardware, las señales se procesan siguiendo la filosofía FIFO (First In First Out). Cuando una señal se envía a un proceso, se ejecuta el manejador correspondiente, y mientras este manejador no termine su ejecución, ninguna otra señal podrá ser recibida por el proceso anterior.

Asignación de un manejador de señal

Como se ha comentado anteriormente, puede ocurrir que un proceso no quiera ejecutar el manejador por defecto asociado a un tipo de señal. Para este caso, existe una función signal() en la biblioteca standard ANSI C, la cual tiene la siguiente sintaxis, (para una información más detallada consultar el manual):

signal (sig, func)

donde:

sig => Número entero que representa una señal definida en /usr/include/signal.h

func => Especifica la dirección de un manejador de señal, dada por el usuario en el proceso.

Para las señales SIGKILL, SIGSTOP y SIGCONT no es posible asignar un manejador que no sea el de defecto. Existen dos manejadores predefinidos en /usr/include/signal.h:

SIG_DFL => Manejador por defecto

SIG_IGN => Manejador que ignora la señal recibida.

Envío de señales a otros procesos.

Para poder enviar una señal a otro proceso, es necesario realizar la llamada al sistema `kill()`. Su formato o sintaxis es, (para una información más detallada consultar el manual):
`ret = kill (pid, sig)`

donde:

`pid` => Identificador del proceso al cual va dirigida la señal.
`sig` => Señal enviada.

`ret` => 0 (Éxito) 1 (Error)

La llamada al sistema `pause()`, provoca la suspensión de la ejecución del proceso, hasta que se recibe una señal. Siempre retorna -1.

SISTEMAS OPERATIVOS 2

LABORATORIO

LLAMADAS AL SISTEMA: MANEJO DE FICHEROS Y DIRECTORIOS.



Llamadas al sistema para la administración de archivos.

Administrador de archivos: aplicación que provee acceso a archivos y facilita el realizar operaciones con ellos.

Archivo: conjunto de información que se encuentra almacenada o guardada en la memoria principal del ordenador, en el disco duro, en el disquete flexible o en los discos compactos.

Características del sistema de archivos

- Contiene programas de administración que controla la creación, borrado y acceso de archivos de datos y programas.
- Mantiene el registro de la ubicación física de los archivos.
- Llamadas mas usadas: read & write (mismos parámetros).

Llamada Lseek

- Proceso de forma aleatoria.
- Capaz de leer los registros del archivo en cualquier orden.
- Con cada archivo hay un apuntador asociado, el cual indica la posición actual en el archivo.
- Cada operación de lectura (read) de la posición en el archivo con la cual iniciar.
- Una operación especial (seek) establece la posición de trabajo pudiendo luego leerse el archivo secuencialmente.

Lseek tiene parámetros:

1. Descriptor del archivo,
2. Posición en el archivo,
3. Indica si la posición en el archivo es relativa al inicio del mismo, a la posición actual o al final del archivo.

El valor devuelto por lseek es la posición absoluta en el archivo (en bytes) después de modificar el apuntador.

Llamada STAT

Funciona para mostrar información del archivo (su tamaño, la hora de la ultima modificación).
Tiene dos parámetros:

1. Especifica el archivo que se va a inspeccionar,
2. Es un apuntador a una estructura en donde se va a colocar la información.

Llamadas al sistema para manejo de archivos

open

La llamada al sistema open es el primer paso que debe ejecutar todo proceso que quiera acceder a los datos de un archivo. Su sintaxis es:

```
fd = open (pathname, flags, modes);
```

donde:

pathname: nombre del archivo.

flags: modo de apertura, para lectura, escritura, etc.

modes: permisos del archivo en caso de que se deba crear.

fd: entero que representa el descriptor del archivo.

La llamada retorna un descriptor de archivo que será utilizado cada vez que deseemos realizar una operación en el archivo. El algoritmo que el sistema operativo ejecuta cada vez que se hace una llamada al sistema open es:

```
open(nombre archivo, modo apertura, permisos del archivo) {  
    convertir el nombre del archivo a inode (bloquear inode para garantizar su uso exclusivo);  
    if (el archivo no existe o no tiene los permisos adecuados)  
        return (error);  
    asignar una entrada en la tabla de archivos abiertos que contiene:  
        puntero al inode del archivo  
        offset=0  
        contador indica los descriptores de archivo que apuntan a esta entrada;  
    asignar una entrada en la tabla de descriptores del proceso que realiza la llamada contiene:  
        puntero a la entrada de la tabla de archivos abiertos;  
    if (O_TRUNC)  
        liberar todos los bloques del archivo;  
        desbloquear inode;  
    return (indice correspondiente a la entrada asignada en la tabla de descriptores de archivo);  
} /* open */
```

A continuación se muestra la representación gráfica de la tabla de archivos, la tabla de inodos y las tablas de descriptores de los procesos A y B después de ejecutarse:
para el proceso A:

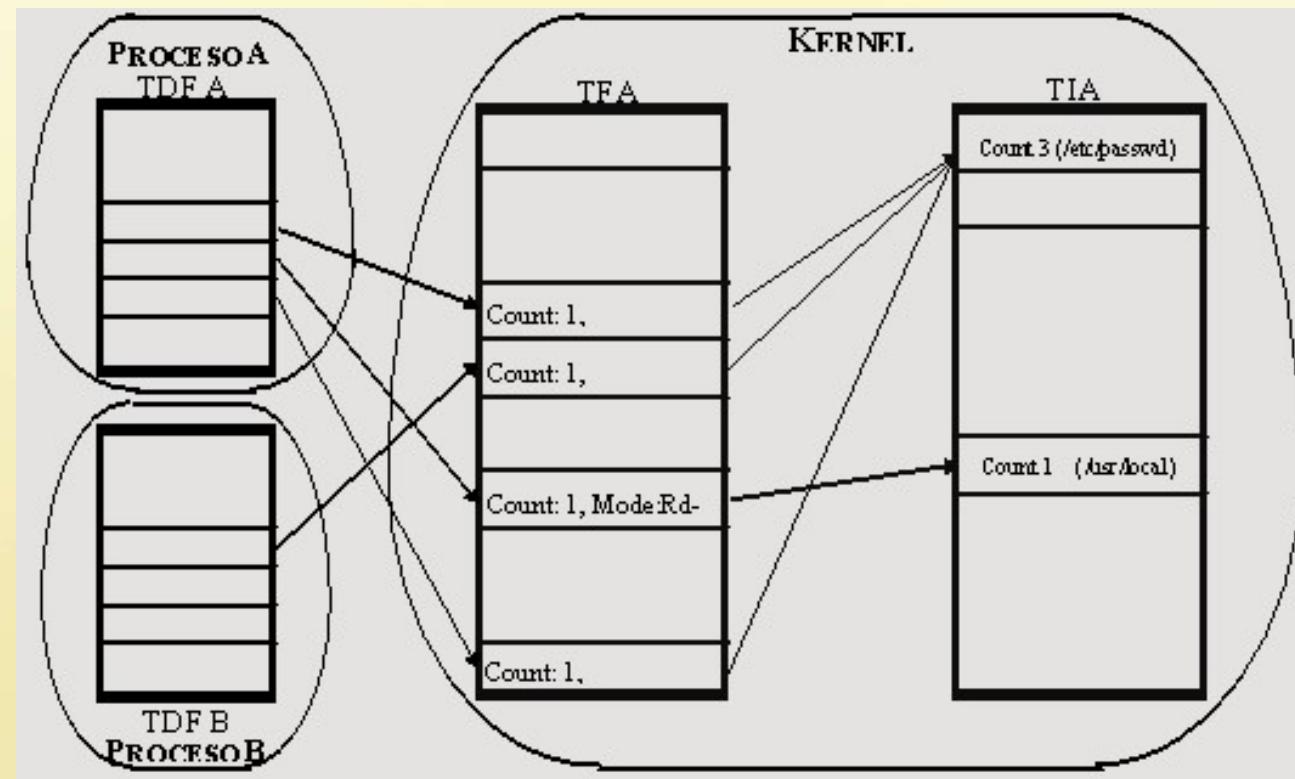
```
fd1=open("/etc/passwd",O_RDONLY);
```

```
fd2=open("/local",O_RDWR);
```

```
fd3=open("/etc/passwd",O_WRONLY);
```

para el proceso B:

```
fd1=open("/etc/passwd",O_RDONLY);
```



read

Su sintaxis es:

```
number = read (fd, buffer, count);
```

donde:

fd: descriptor de archivo que devuelve la llamada open

buffer: dirección del buffer donde se van a colocar los datos leídos

count: número de bytes que el usuario quiere leer

number: número de bytes leídos si es positivo en caso contrario la ejecución no ha sido correcta

El algoritmo que el sistema operativo ejecuta cada vez que se hace una llamada al sistema read es:

```
read (descriptor, dirección, nº de bytes) {  
    buscar la entrada de la tabla de archivos abiertos que corresponde al descriptor;  
    comprobar los permisos de acceso;  
    seleccionar parámetros en la u-area;  
    Leer el inode de la tabla de archivos;  
    bloquear inode;  
    leer el offset del archivo que está en la tabla de archivos;  
    while(no se realice la lectura) {  
        convertir el offset del archivo a un número de bloque;  
        calcular el offset dentro del bloque, nº de bytes a leer;  
        if( el nº de bytes a leer es 0)  
            break;  
        leer el bloque;  
        copiar los datos del buffer del sistema a la dirección del usuario;  
        actualizar los campos de la u-area: el offset y contador de lectura;  
        release buffer;  
    } /* while */  
    desbloquear inode;  
    actualizar el offset de las tablas de archivos;  
    return (nº total de bytes leidos);  
} /* read */
```

Los I/O parámetros de la u-area son: mode lectura o escritura, count nº de bytes a leer o escribir, offset del archivo, address dirección donde se copiaran los datos, flag indica si la dirección esta en la memoria del usuario o en la del kernel.

write

La sintaxis es:

```
number=write (fd, buffer, count);
```

donde el significado de las variables es el mismo que para la llamada al sistema read. El algoritmo que utiliza el sistema operativo para escribir un archivo es similar al que utiliza para leerlo.

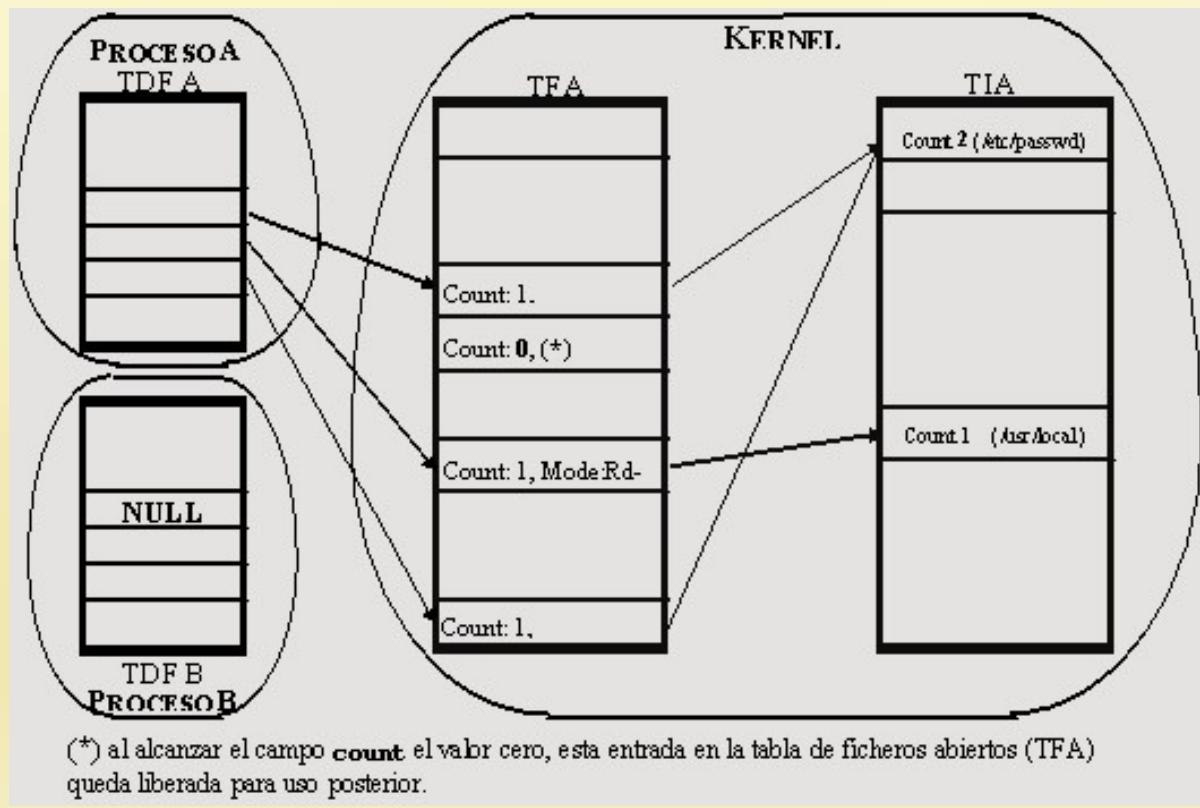
close

Un proceso cierra un archivo cuando no lo va a utilizar más. La sintaxis de la llamada close es:

```
close(fd);
```

donde fd es el descriptor del archivo.

Si cerramos el archivo abierto por el proceso B, la representación gráfica de la tabla de archivos, la tabla de inodes y las tablas de descriptores de los procesos quedaría:



Llamadas al sistema para la administración de directorios.

Llamadas MKDIR y RMDIR

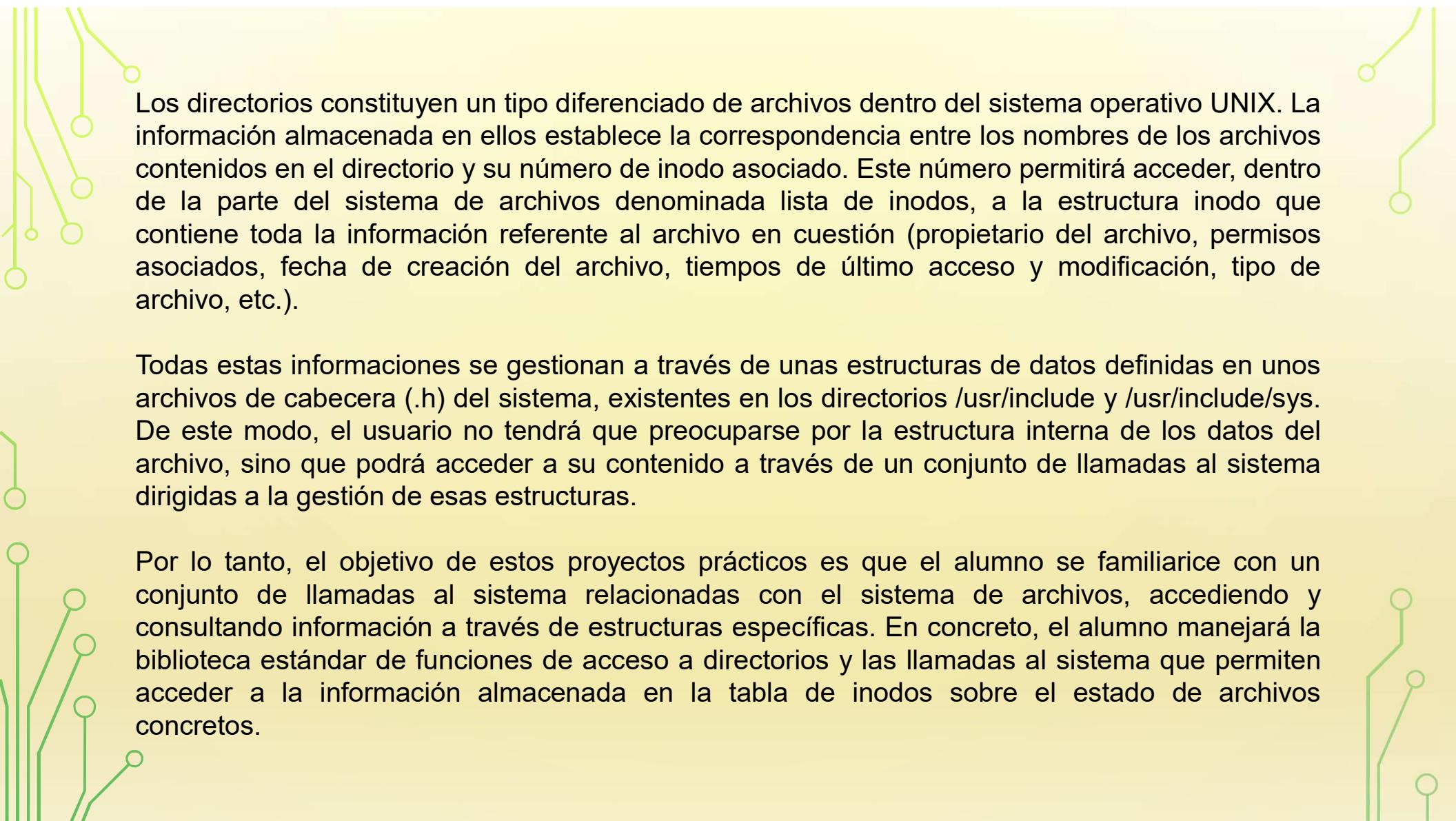
Crean y eliminan directorios vacíos, respectivamente.

Llamada Link.

Su propósito es permitir que aparezca el mismo archivo bajo dos o mas nombres, a menudo en distintos directorios.

Un uso común es para permitir que varios miembros del mismo equipo de programación comparten un archivo común.

Cada uno de ellos puede ver el archivo en su propio directorio.



Los directorios constituyen un tipo diferenciado de archivos dentro del sistema operativo UNIX. La información almacenada en ellos establece la correspondencia entre los nombres de los archivos contenidos en el directorio y su número de inodo asociado. Este número permitirá acceder, dentro de la parte del sistema de archivos denominada lista de inodos, a la estructura inodo que contiene toda la información referente al archivo en cuestión (propietario del archivo, permisos asociados, fecha de creación del archivo, tiempos de último acceso y modificación, tipo de archivo, etc.).

Todas estas informaciones se gestionan a través de unas estructuras de datos definidas en unos archivos de cabecera (.h) del sistema, existentes en los directorios /usr/include y /usr/include/sys. De este modo, el usuario no tendrá que preocuparse por la estructura interna de los datos del archivo, sino que podrá acceder a su contenido a través de un conjunto de llamadas al sistema dirigidas a la gestión de esas estructuras.

Por lo tanto, el objetivo de estos proyectos prácticos es que el alumno se familiarice con un conjunto de llamadas al sistema relacionadas con el sistema de archivos, accediendo y consultando información a través de estructuras específicas. En concreto, el alumno manejará la biblioteca estándar de funciones de acceso a directorios y las llamadas al sistema que permiten acceder a la información almacenada en la tabla de inodos sobre el estado de archivos concretos.

Las llamadas al sistema de acceso a directorios que se manejarán en la presente práctica son: opendir, closedir, readdir y seekdir. Estas llamadas operan sobre una estructura de datos opaca denominada DIR, y cuya definición y descripción de sus campos se suele encontrar en el archivo de cabecera del sistema <dirent.h>, localizado en el directorio /usr/include. A continuación resumimos brevemente los campos de la estructura así como la sintaxis de las llamadas al sistema fundamentales:

```
typedef struct __dирdesc {  
    int dd_fd; /* descriptor del archivo asociado al directorio */  
    long dd_loc; /* offset en el buffer actual */  
    long dd_size; /* número de datos devuelto por getdirentries */  
    char *dd_buf; /* buffer de datos */  
    long dd_bufsize;  
    long dd_size; /* tamaño del buffer */  
    ...  
} DIR;
```

NOTA: al ser una estructura opaca tanto su localización como su estructura puede variar de unos sistemas a otros; por lo que no se recomienda el acceso a sus campos si se pretende llevar a cabo un desarrollo portable.

La llamada para abrir un archivo de directorio es opendir, cuya sintaxis es:

```
DIR *opendir (char *dirname)
```

donde, dirname es una cadena de caracteres que contiene el path name del directorio al que se va a abrir, y tras su ejecución correcta, devuelve un puntero a una estructura tipo DIR. Si se produce un error en la apertura del directorio, el valor devuelto será NULL.

La llamada para cerrar un archivo de directorio es closedir, cuya sintaxis es:

```
int closedir(DIR *dirp)
```

donde, dirp es un puntero a una estructura tipo DIR previamente abierta. Si la ejecución es correcta, devolverá 0, y en caso contrario, -1, indicando el tipo de error producido a través de la variable global *errno*.

La información contenida en un archivo de directorio está formada por lo que se denominan entradas del directorio. Para poder acceder cómodamente a ellas, se emplea la llamada al sistema readdir, cuya sintaxis es:

```
struct dirent *readdir(DIR *dirp)
```

donde dirp es un puntero a una estructura DIR previamente abierta. Su funcionamiento correcto hace que readdir lea la entrada donde se encuentre situado el puntero de lectura de ese directorio, actualizando el puntero a la siguiente entrada. De este modo se permite una lectura de tipo secuencial. readdir devuelve la entrada leída a través de un puntero a una estructura de tipo struct dirent (definida al igual que DIR en el archivo de cabecera <dirent.h>). La estructura dirent tiene la siguiente composición:

```
typedef struct dirent {  
    ino_t d_ino; /* Número de inode asociado a esa entrada */  
    short d_reclen; /* Longitud de la entrada al directorio */  
    short d_namlen; /* Longitud del string que hay en d_name */  
    char d_name[256]; /* Nombre del archivo */  
};
```

El manejo del puntero de acceso a los elementos del directorio se hace necesario para ofrecer toda la flexibilidad posible en el desarrollo de aplicaciones. Este manejo se realiza con las llamadas al sistema seekdir, telldir y rewinddir. La primera permitirá situar el puntero de lectura en un elemento determinado del directorio mientras que la segunda devolverá su posición actual y la tercera permite devolver situar el puntero al principio del directorio. Su sintaxis puede consultarse, al igual que para las funciones anteriores, mediante el uso de la orden man o bien en la bibliografía recomendada.

Conocido el número de inodo asociado a un archivo, puede ser necesario acceder a la información contenida en el inodo. Para ello, se emplean las llamadas al sistema stat, Istat (la diferencia entre ambas es que Istat devuelve el symlink y stat el archivo al que apunta symlink) y fstat, las cuales devuelven la información almacenada en la tabla de inodos sobre el estado de un archivo concreto. La sintaxis de ambas llamadas es:

```
int stat(char *path, struct stat *buf)
int Istat(char *path, struct stat *buf)
int fstat(int fildes, struct stat *buf)
```

siendo la diferencia el que la primera reciba como primer parámetro un puntero al path del archivo, mientras la segunda emplea un archivo ya abierto.

Estas llamadas devuelven la información almacenada en una estructura del tipo struct stat, la cual definida en el archivo de cabecera `<sys/stat.h>` y cuyos tipos incluidos se definen en el archivo `<sys/types.h>` (normalmente como alias a tipos base como el short, int o long). Entre sus campos, el que indica el modo del archivo contiene información heterogénea distribuida en sus bits. Para acceder a esa información existen un conjunto de constantes definidas en `<sys/mode.h>` que aplicadas mediante una operación and (&) sobre el campo correspondiente, devuelven la información requerida.

La estructura stat, devuelta por las llamadas indicadas, consta de distintos campos, los cuales dependen de la versión de UNIX que estemos utilizando, de cualquier forma sus campos estándar se citan a continuación.

<code>dev_t st_dev</code>	Número del dispositivo que contiene al inodo. <i>Major number</i> (8 bits de mayor peso) y <i>minor number</i> (8 bits de menor peso)
<code>ino_t st_ino</code>	Número de inodo
<code>ushort st_mode</code>	16 bits que contienen el tipo y los permisos del archivo
<code>uid_t st_uid</code>	Identificador de usuario (UID) propietario del archivo
<code>gid_t st_gid</code>	Identificador del grupo (GID) al que pertenece el propietario del archivo
<code>dev_t st_rdev</code>	Identificador de dispositivo. Sólo tiene sentido para archivos de dispositivo tipo carácter o bloque
<code>off_t st_size</code>	Tamaño del archivo en <i>bytes</i>
<code>time_t st_atime</code>	Fecha del último acceso
<code>time_t st_mtime</code>	Fecha de la última modificación del archivo
<code>time_t st_ctime</code>	Fecha del último cambio de la información administrativa. Todas las fechas están dadas en segundos con respecto a las 00:00:00 GMT del día 1 de Enero de 1970

Con respecto al campo *st_mode*, señalar que los 9 bits de menor peso son los que indican los derechos de acceso para el propietario, el grupo y el resto de usuarios y los 4 de mayor peso determinan el tipo de archivo acorde con la siguiente tabla:

1000	Ordinario
0100	Directorio
0010	Especial modo carácter
1100	Especial modo bloque
0001	Fifo
1010	Enlace simbólico

MISCELÁNEA DE LLAMADAS AL SISTEMA

La llamada a chdir cambia el directorio de trabajo actual. Después de la llamada chdir("/usr/ast/prueba"); una instrucción para abrir el archivo xyz abrirá /usr/ast/prueba/xyz.

El concepto de un directorio de trabajo elimina la necesidad de escribir nombres de ruta absolutos (extensos) todo el tiempo.

En UNIX, cada archivo tiene un modo que se utiliza por protección. El modo incluye los bits leer-escribir-ejecutar para el propietario, grupo y los demás. La llamada al sistema chmod hace posible modificar el modo de un archivo. Por ejemplo, para que un archivo sea de sólo lectura para todos excepto el propietario, podríamos ejecutar chmod("archivo", 0644);



La llamada al sistema kill es la forma en que los usuarios y los procesos de usuario envían señales. Si un proceso está preparado para atrapar una señal específica, y luego ésta llega, se ejecuta un manejador de señales. Si el proceso no está preparado para manejar una señal, entonces su llegada mata el proceso (de aquí que se utilice ese nombre para la llamada). POSIX define varios procedimientos para tratar con el tiempo. Por ejemplo, time sólo devuelve la hora actual en segundos, en donde 0 corresponde a Enero 1, 1970 a medianoche (justo cuando el día empezaba y no terminaba).

En las computadoras que utilizan palabras de 32 bits, el valor máximo que puede devolver time es de $2^{32} - 1$ segundos (suponiendo que se utiliza un entero sin signo).

Este valor corresponde a un poco más de 136 años. Así, en el año 2106 los sistemas UNIX de 32 bits se volverán locos, algo parecido al famoso problema del año 2000 (Y2K) que hubiera causado estragos con las computadoras del mundo en el 2000, si no fuera por el masivo esfuerzo que puso la industria de IT para corregir el problema.

Si actualmente usted tiene un sistema UNIX de 32 bits, se le recomienda que lo intercambie por uno de 64 bits antes del año 2106.

LA API WIN32 DE WINDOWS

La API Win32 de Windows y UNIX difieren de una manera fundamental en sus respectivos modelos de programación.

Un programa de UNIX consiste en código que realiza una cosa u otra, haciendo llamadas al sistema para realizar ciertos servicios. En contraste, un programa de Windows es por lo general manejado por eventos.

El programa principal espera a que ocurra cierto evento y después llama a un procedimiento para manejarlo. Los eventos comunes son las teclas que se oprimen, el ratón que se desplaza, un botón de ratón que se oprime o un CD-ROM que se inserta. Después, los manejadores se llaman para procesar el evento, actualizar la pantalla y actualizar el estado interno del programa. Desde luego que Windows también tiene llamadas al sistema.

Con UNIX, hay casi una relación de uno a uno entre las llamadas al sistema (por ejemplo, read) y los procedimientos de biblioteca (por ejemplo, read) que se utilizan para invocar las llamadas al sistema. En otras palabras, para cada llamada al sistema, es raro que haya un procedimiento de biblioteca que sea llamado para invocarlo POSIX tiene aproximadamente 100 llamadas a procedimientos.

Administración de procesos

Llamada Descripción

pid = fork() Crea un proceso hijo, idéntico al padre

pid = waitpid(pid, &statloc, opciones) Espera a que un hijo termine

s = execve(nombre, argv, entornp) Reemplaza la imagen del núcleo de un proceso
exit(estado) Termina la ejecución de un proceso y devuelve el estado

Administración de archivos

Llamada Descripción

fd = open(archivo, como, ...) Abre un archivo para lectura, escritura o ambas

s = close(fd) Cierra un archivo abierto

`n = read(fd, bufer, nbytes)` Lee datos de un archivo y los coloca en un búfer
`n = write(fd, bufer, nbytes)` Escribe datos de un búfer a un archivo

`posición = lseek(fd, desplazamiento, Desplaza el apuntador del archivo dedonde)`

`s = stat(nombre, &buf)` Obtiene la información de estado de un archivo

Administración del sistema de directorios y archivos

Llamada Descripción

`s = mkdir(nombre, modo)` Crea un nuevo directorio
`s = rmdir(nombre)` Elimina un directorio vacío
`s = link(nombre1, nombre2)` Crea una nueva entrada llamada nombre2, que apunta a nombre1
`s = unlink(nombre)` Elimina una entrada de directorio
`s = mount(especial, nombre, bandera)` Monta un sistema de archivos
`s = umount(especial)` Desmonta un sistema de archivos



Llamadas varias

Llamada Descripción

s = chdir(nombredir) Cambia el directorio de trabajo

s = chmod(nombre, modo) Cambia los bits de protección de un archivo

s = kill(pid, senial) Envía una señal a un proceso

segundos = tiempo(&segundos) Obtiene el tiempo transcurrido desde Ene 1, 1970



SISTEMAS OPERATIVOS 2

LABORATORIO

SISTEMA DE FICHEROS PROCFS



Sistema de ficheros procfs.

En los sistemas operativos tipo Unix, procfs es la abreviatura de sistema de ficheros de procesos (process filesystem). Un pseudo sistema de ficheros que se utiliza para permitir el acceso a la información del núcleo sobre los procesos. Dado que proc no es un sistema de ficheros real, no consume ningún espacio de almacenamiento, y sólo consume una limitada cantidad de memoria.

El sistema de archivos se monta con frecuencia en /proc. Está soportado bajo Solaris, BSD y GNU/Linux, el último de los cuales lo extiende para incluir datos que no son propios de los procesos.

La implementación de la versión de /proc utilizada en la octava edición de Unix, estaba implementada por Tom J. Killina, quien presentó un escrito titulado Processes as Files en USENIX, junio de 1984. Se diseñó para reemplazar la llamada al sistema ptrace utilizada en el seguimiento de los procesos.

Roger Falulkner y Ron Gomes portaron el /proc de V8 al SVr4, y publicaron un escrito titulado El Sistema de Ficheros de Procesos y el Modelo de Procesos en UNIX System V en USENIX, enero de 1991. Este tipo de procfs soportaba la creación de procesos, pero los ficheros sólo podían ser accedidos mediante las funciones read(), write(), e ioctl().

BSD implementó /proc con subdirectorios para cada proceso, y la habilidad de acceder a la memoria, registros, y el estado actual. En Solaris 2.6 /proc (finalizado en 1996) también tuvo un directorio diferente para cada proceso, además de un fichero especial ctl que permitía el control, seguimiento y manipulación de los procesos.

Bajo GNU/Linux, /proc proporciona información sobre cualquier proceso en ejecución en /proc/PID, pero además incluye:

Un enlace simbólico al proceso actual en /proc/self

Información sobre el hardware, núcleo y la configuración de módulos

Acceso a las opciones dinámicamente configurables del núcleo bajo /proc/sys

Las utilidades básicas que utilizan /proc bajo Linux se encuentran en el paquete procps, y necesitan que /proc esté montado para realizar su función.

En el núcleo 2.6, la mayoría de los ficheros no relacionados con los procesos que se encontraban en /proc se movieron a otro sistema de ficheros virtual llamado sysfs (montado en /sys).

El kernel de Linux tiene dos funciones primarias: controlar el acceso a los dispositivos físicos de la computadora y establecer cuándo y cómo los procesos interactuarán con estos dispositivos. El directorio `/proc` — también llamado el sistema de archivos proc — contiene una jerarquía de archivos especiales que representan el estado actual del kernel — permitiendo a las aplicaciones y usuarios mirar detenidamente en la vista del kernel del sistema.

Dentro del directorio `/proc`, se puede encontrar una gran cantidad de información con detalles sobre el hardware del sistema y cualquier proceso que se esté ejecutando actualmente. Además, algunos de los archivos dentro del árbol de directorios `/proc` pueden ser manipulados por los usuarios y aplicaciones para comunicar al kernel cambios en la configuración.

El sistema de archivos `/proc` contiene un sistema de archivos imaginario o virtual. Este no existe físicamente en disco, sino que el núcleo lo crea en memoria. Se utiliza para ofrecer información relacionada con el sistema (originalmente acerca de procesos, de aquí su nombre).

Algunos de los archivos más importantes son:

`/proc/1`

Un directorio con información acerca del proceso número 1. Cada proceso tiene un directorio debajo de `/proc` cuyo nombre es el número de identificación del proceso (PID).

`/proc/cpuinfo`

Información acerca del procesador: su tipo, marca, modelo, rendimiento, etc.

`/proc/devices`

Lista de controladores de dispositivos configurados dentro del núcleo que está en ejecución.

`/proc/dma`

Muestra los canales DMA que están siendo utilizados.

`/proc/filesystems`

Lista los sistemas de archivos que están soportados por el kernel.

/proc/interrupts

Muestra la interrupciones que están siendo utilizadas, y cuantas de cada tipo ha habido.

/proc/ioports

Información de los puertos de E/S que se estén utilizando en cada momento.

/proc/kcore

Es una imagen de la memoria física del sistema. Este archivo tiene exactamente el mismo tamaño que la memoria física, pero no existe en memoria como el resto de los archivos bajo /proc, sino que se genera en el momento en que un programa accede a este. (Recuerde: a menos que copie este archivo en otro lugar, nada bajo /proc usa espacio en disco).

/proc/kmsg

Salida de los mensajes emitidos por el kernel. Estos también son redirigidos hacia syslog.

/proc/ksyms

Tabla de símbolos para el kernel.

/proc/loadavg

El nivel medio de carga del sistema; tres indicadores significativos sobre la carga de trabajo del sistema en cada momento.

/proc/meminfo

Información acerca de la utilización de la memoria física y del archivo de intercambio.

/proc/modules

Indica los módulos del núcleo que han sido cargados hasta el momento.

/proc/net

Información acerca del estado de los protocolos de red.

/proc/self

Un enlace simbólico al directorio de proceso del programa que esté observando a /proc. Cuando dos procesos observan a /proc, obtienen diferentes enlaces. Esto es principalmente una conveniencia para que sea fácil para los programas acceder a su directorio de procesos.

/proc/stat

Varias estadísticas acerca del sistema, tales como el número de fallos de página que han tenido lugar desde el arranque del sistema.

/proc/uptime

Indica el tiempo en segundos que el sistema lleva funcionando.

/proc/version

Indica la versión del núcleo

Conviene aclarar que aunque los archivos anteriores tienden a ser archivos de texto fáciles de leer, algunas veces pueden tener un formato que no sea fácil de interpretar. Por ello existen muchos comandos que solamente leen los archivos anteriores y les dan un formato distinto para que la información sea fácil de entender. Por ejemplo, el comando free, lee el archivo /proc/meminfo y convierte las cantidades dadas en bytes a kilobytes (además de agregar un poco más de información extra).

Sistema de archivos virtual.

En Linux, todo se guarda en archivos. La mayoría de usuarios están familiarizados con los dos primeros tipos de archivos, de texto y binarios. Sin embargo, el directorio `/proc/` contiene otro tipo de archivos llamado archivo virtual. Por esta razón, es que a menudo se hace referencia a `/proc/` como un sistema de archivos virtual.

Estos archivos virtuales poseen cualidades únicas. En primer lugar, la mayoría de ellos tienen un tamaño de 0 bytes. Sin embargo, cuando se visualiza el archivo, éste puede contener una gran cantidad de información. Además, la mayoría de configuraciones del tiempo y las fechas reflejan el tiempo y fecha real, lo que es un indicativo de que están siendo constantemente modificados.



Los archivos virtuales tales como `/proc/interrupts`, `/proc/meminfo`, `/proc/mounts`, y `/proc/partitions` proporcionan una vista rápida actualizada del hardware del sistema. Otros, como `/proc/filesystems` y el directorio `/proc/sys/`, proveen información de configuración del sistema e interfaces.

Para propósitos organizacionales, los archivos que contienen información sobre un tópico similar se agrupan en directorios virtuales y sub-directorios. Por ejemplo, `/proc/ide/` contiene información sobre los dispositivos IDE. De la misma forma, los directorios de procesos contienen información sobre cada proceso ejecutándose en el sistema.

Visualización de archivos virtuales

Mediante el uso de los comandos `cat`, `more`, o `less` en los archivos dentro del directorio `/proc/`, los usuarios pueden inmediatamente acceder una cantidad enorme de información acerca del sistema. Por ejemplo, para desplegar el tipo de CPU que tiene un equipo, escriba `cat /proc/cpuinfo` para recibir una salida similar a lo siguiente:

```
processor : 0
vendor_id : AuthenticAMD
cpu family : 5
model : 9
model name : AMD-K6(tm) 3D+ Processor
stepping : 1
cpu MHz : 400.919
cache size : 256 KB
fdiv_bug : no
hlt_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 1
wp : yes
flags : fpu vme de pse tsc msr mce cx8 pge mmx syscall 3dnow k6_mtrr
bogomips : 799.53
```

Como puede ver en el sistema de archivos /proc/, alguna información tiene sentido, mientras que otras áreas aparecen en un código extraño. Por eso es que existen utilidades para extraer información de los archivos virtuales y mostrarla en una forma útil. Ejemplos de estas utilidades incluyen lspci, apm, free, y top.

Cambiar archivos virtuales

Como regla general, la mayoría de los archivos virtuales dentro del directorio /proc solamente se pueden leer. Sin embargo, algunos se pueden usar para ajustar la configuración del kernel. Esto ocurre con los archivos del subdirectorio /proc/sys/.

Para cambiar el valor de un archivo virtual, use el comando echo y el símbolo mayor que (>) para redirigir el nuevo valor al archivo. Por ejemplo, para cambiar el nombre del host rápidamente escriba:

```
echo www.example.com > /proc/sys/kernel/hostname
```

Otros archivos actúan como conmutadores binarios o booleanos. Si escribe cat /proc/sys/net/ipv4/ip_forward verá el valor 0 o el valor 1. El valor 0 indica que el kernel no está realizando el reenvío de paquetes. Si usa el comando echo para cambiar el valor del archivo ip_forward a 1, el kernel activará inmediatamente el reenvío de paquetes.

Fundamentos técnicos del virtual filesystem

Una de las razones por la que GNU/Linux permite acceder a cada sistema de archivos real como un subdirectorio del sistema de archivos virtual, es para homogeneizar la forma en la que se accede a ciertas estructuras de datos dentro del sistema operativo.

De esta manera la interfaz que se utiliza para acceder a archivos reales de disco también puede ser utilizada con cualquier otro tipo de archivo virtual, y éste mecanismo le da una flexibilidad enorme al sistema operativo, y simplifica la forma en que las aplicaciones pueden acceder a los datos de discos y del sistema operativo, y simplifica también las técnicas de desarrollo de aplicaciones.

En sistemas Windows existe una unidad de disco por cada disco/partición, con sus archivos, y un panel de control para administrar muchos dispositivos y algunas características del sistema.

En Linux mediante el sistema de archivos virtual también se pueden administrar dispositivos, e incluso administrar el estado de ejecución de cada uno de los procesos que están corriendo, porque mucha de la información de los procesos en ejecución también se ve como una serie de archivos en disco, pero se encuentran en memoria o, alternativamente, en el espacio de almacenamiento de intercambio (swap). En Windows ver el estado de los procesos de esta forma es imposible.

El directorio `/dev` contiene archivos virtuales de dispositivos, se «ven» como archivos, pero son en realidad archivos especiales, o nodos, y representan los dispositivos del sistema.

`/proc` contiene archivos virtuales de administración de procesos, por ejemplo, si listamos los procesos del sistema y filtramos (para reducir la salida) por alguna palabra particular, por ejemplo, «firefox», tendríamos la siguiente salida:

```
$ ps fax|grep pumpa  
6806  tty1  SI  2:01 pumpa  
29892pts/0  S+  0:00 | _ grep pumpa
```

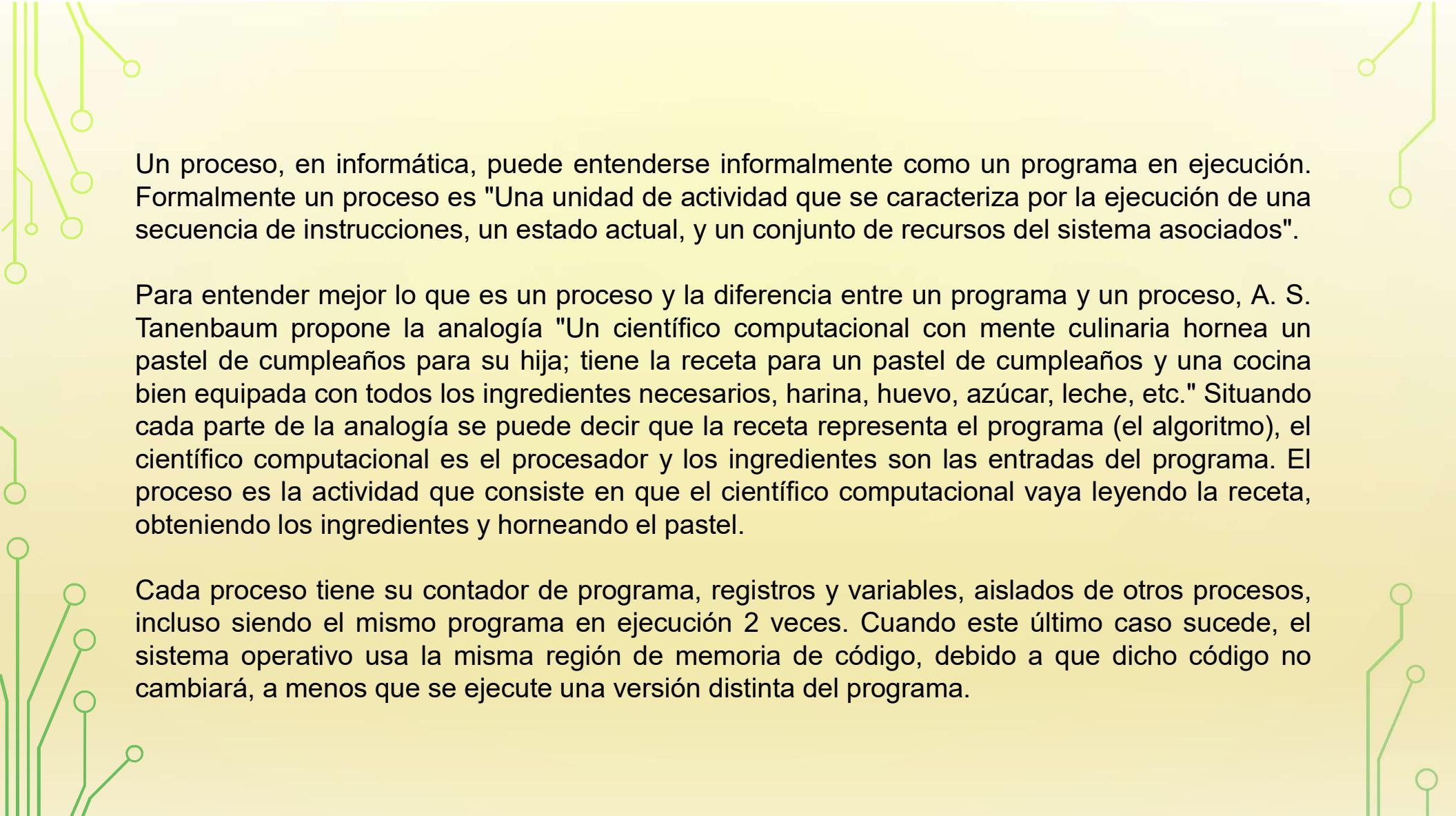
Donde la primer columna corresponde al PID, o process ID de los procesos que contienen la palabra «pumpa» en su línea de ejecución.

SISTEMAS OPERATIVOS 2

LABORATORIO

PROCESO

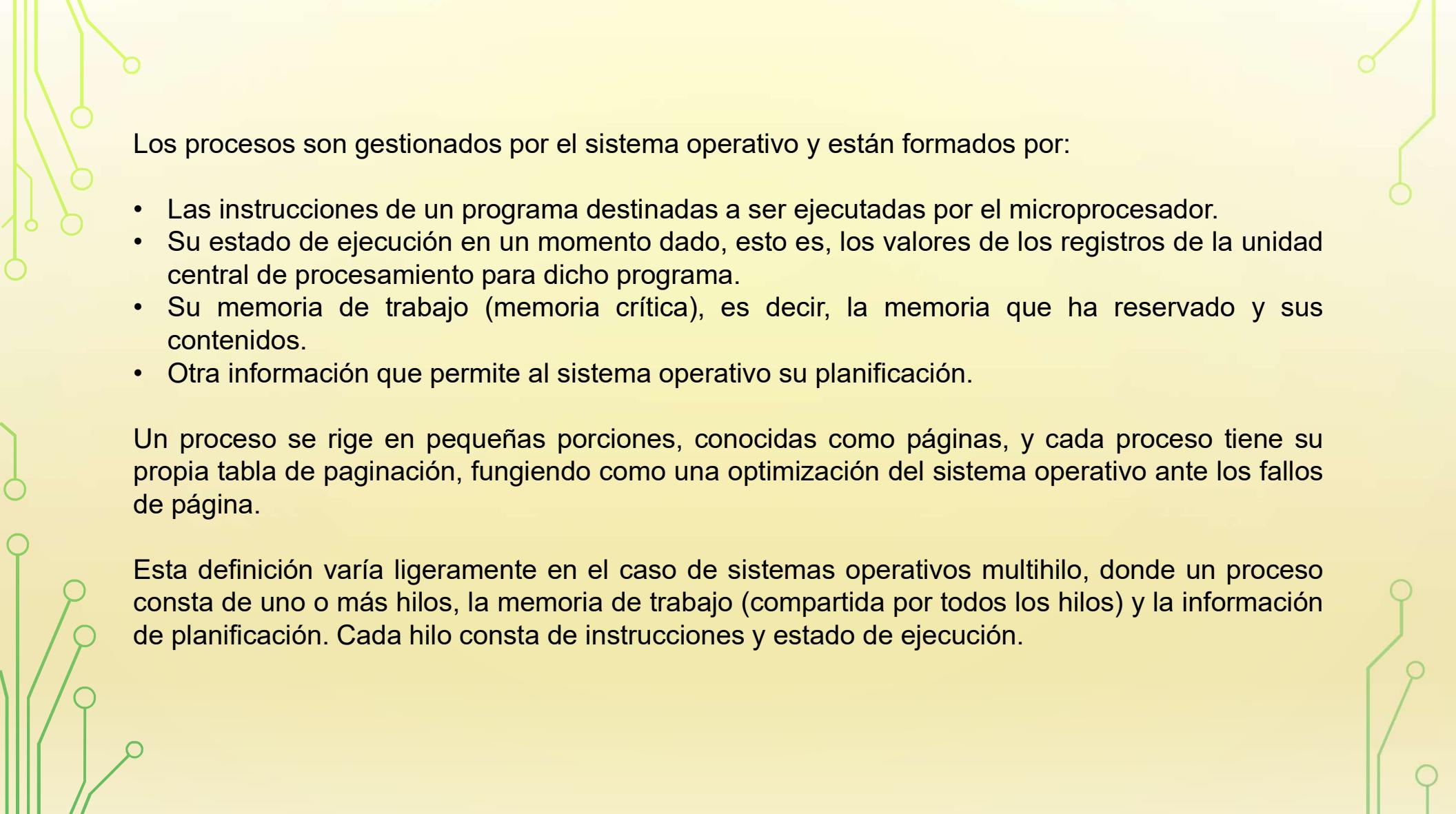
```
sergio@sergio-VirtualBox:~$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START  TIME COMMAND
root        1  0.1  0.2 168028 11364 ?        Ss  16:07  0:02 /sbin/init spla
root        2  0.0  0.0     0     0 ?        S   16:07  0:00 [kthreadd]
root        3  0.0  0.0     0     0 ?        I<  16:07  0:00 [rcu_gp]
root        4  0.0  0.0     0     0 ?        I<  16:07  0:00 [rcu_par_gp]
root        5  0.0  0.0     0     0 ?        I<  16:07  0:00 [slub_flushwq]
root        6  0.0  0.0     0     0 ?        I<  16:07  0:00 [netns]
root        8  0.0  0.0     0     0 ?        I<  16:07  0:00 [kworker/0:0H]
root        9  0.0  0.0     0     0 ?        I   16:07  0:02 [kworker/u8:0-e]
root       10  0.0  0.0     0     0 ?        I<  16:07  0:00 [mm_percpu_wq]
root       11  0.0  0.0     0     0 ?        S   16:07  0:00 [rcu_tasks_rude]
root       12  0.0  0.0     0     0 ?        S   16:07  0:00 [rcu_tasks_trac
root       13  0.0  0.0     0     0 ?        S   16:07  0:00 [ksoftirqd/0]
root       14  0.0  0.0     0     0 ?        I   16:07  0:01 [rcu_sched]
root       15  0.0  0.0     0     0 ?        S   16:07  0:00 [migration/0]
root       16  0.0  0.0     0     0 ?        S   16:07  0:00 [idle_inject/0]
root       18  0.0  0.0     0     0 ?        S   16:07  0:00 [cpuhp/0]
root       19  0.0  0.0     0     0 ?        S   16:07  0:00 [cpuhp/1]
root       20  0.0  0.0     0     0 ?        S   16:07  0:00 [idle_inject/1]
root       21  0.0  0.0     0     0 ?        S   16:07  0:00 [migration/1]
root       22  0.0  0.0     0     0 ?        S   16:07  0:00 [ksoftirqd/1]
root       24  0.0  0.0     0     0 ?        I<  16:07  0:00 [kworker/1:0H-e
root       25  0.0  0.0     0     0 ?        S   16:07  0:00 [cpuhp/2]
root       26  0.0  0.0     0     0 ?        S   16:07  0:00 [idle_inject/2]
root       27  0.0  0.0     0     0 ?        S   16:07  0:00 [migration/2]
```



Un proceso, en informática, puede entenderse informalmente como un programa en ejecución. Formalmente un proceso es "Una unidad de actividad que se caracteriza por la ejecución de una secuencia de instrucciones, un estado actual, y un conjunto de recursos del sistema asociados".

Para entender mejor lo que es un proceso y la diferencia entre un programa y un proceso, A. S. Tanenbaum propone la analogía "Un científico computacional con mente culinaria hornea un pastel de cumpleaños para su hija; tiene la receta para un pastel de cumpleaños y una cocina bien equipada con todos los ingredientes necesarios, harina, huevo, azúcar, leche, etc." Situando cada parte de la analogía se puede decir que la receta representa el programa (el algoritmo), el científico computacional es el procesador y los ingredientes son las entradas del programa. El proceso es la actividad que consiste en que el científico computacional vaya leyendo la receta, obteniendo los ingredientes y horneando el pastel.

Cada proceso tiene su contador de programa, registros y variables, aislados de otros procesos, incluso siendo el mismo programa en ejecución 2 veces. Cuando este último caso sucede, el sistema operativo usa la misma región de memoria de código, debido a que dicho código no cambiará, a menos que se ejecute una versión distinta del programa.

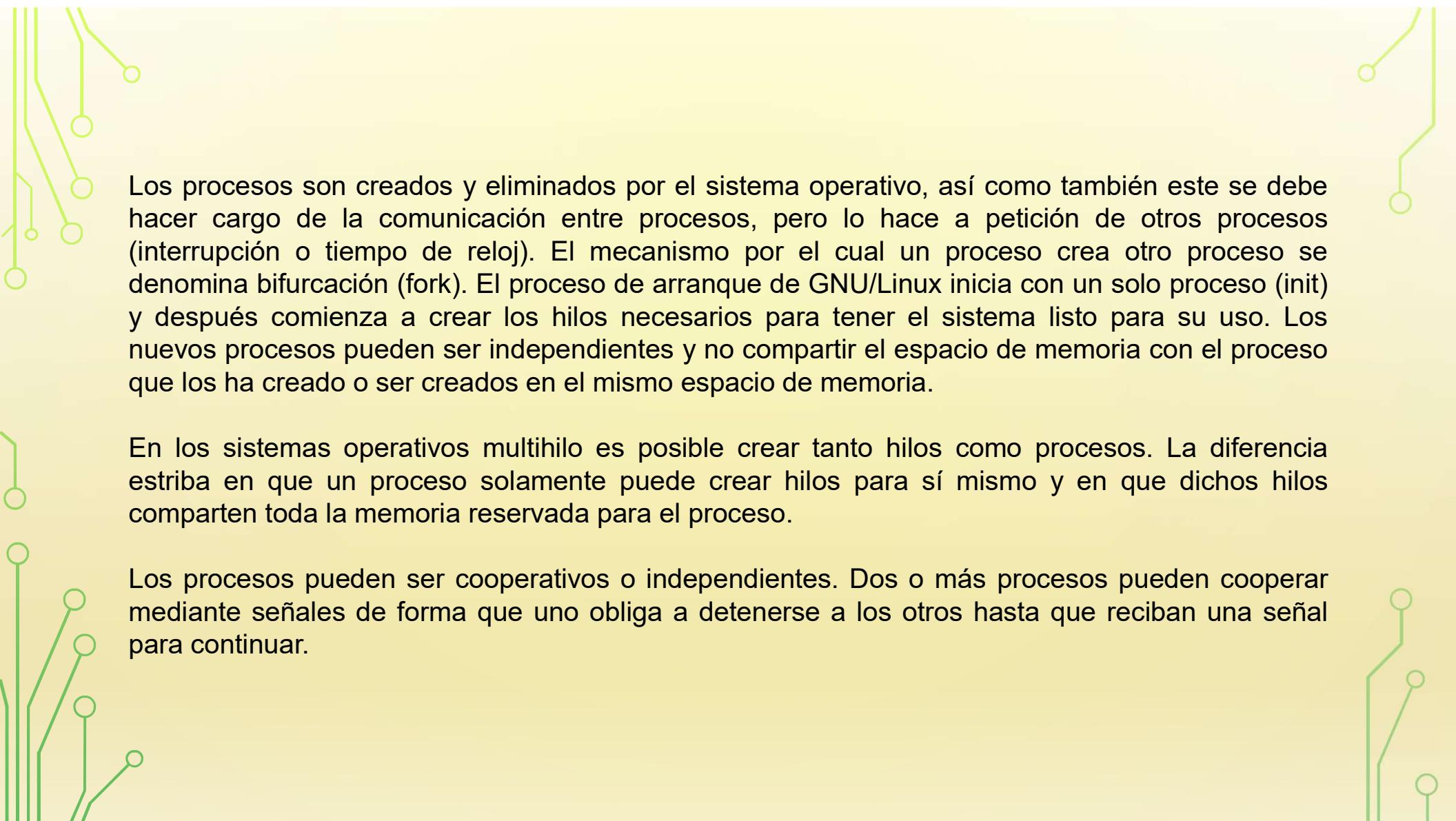


Los procesos son gestionados por el sistema operativo y están formados por:

- Las instrucciones de un programa destinadas a ser ejecutadas por el microprocesador.
- Su estado de ejecución en un momento dado, esto es, los valores de los registros de la unidad central de procesamiento para dicho programa.
- Su memoria de trabajo (memoria crítica), es decir, la memoria que ha reservado y sus contenidos.
- Otra información que permite al sistema operativo su planificación.

Un proceso se rige en pequeñas porciones, conocidas como páginas, y cada proceso tiene su propia tabla de paginación, fungiendo como una optimización del sistema operativo ante los fallos de página.

Esta definición varía ligeramente en el caso de sistemas operativos multihilo, donde un proceso consta de uno o más hilos, la memoria de trabajo (compartida por todos los hilos) y la información de planificación. Cada hilo consta de instrucciones y estado de ejecución.



Los procesos son creados y eliminados por el sistema operativo, así como también este se debe hacer cargo de la comunicación entre procesos, pero lo hace a petición de otros procesos (interrupción o tiempo de reloj). El mecanismo por el cual un proceso crea otro proceso se denomina bifurcación (fork). El proceso de arranque de GNU/Linux inicia con un solo proceso (init) y después comienza a crear los hilos necesarios para tener el sistema listo para su uso. Los nuevos procesos pueden ser independientes y no compartir el espacio de memoria con el proceso que los ha creado o ser creados en el mismo espacio de memoria.

En los sistemas operativos multihilo es posible crear tanto hilos como procesos. La diferencia estriba en que un proceso solamente puede crear hilos para sí mismo y en que dichos hilos comparten toda la memoria reservada para el proceso.

Los procesos pueden ser cooperativos o independientes. Dos o más procesos pueden cooperar mediante señales de forma que uno obliga a detenerse a los otros hasta que reciban una señal para continuar.

- Se usa una variable de tipo semáforo para sincronizar los procesos.
- Si un proceso está esperando una señal, se suspende hasta que la señal se envíe.
- Se mantiene una cola de procesos en espera en el semáforo.
- La forma de elegir los procesos de la cola en espera es mediante una política first in first out.

La sincronización explícita entre procesos es un caso particular del estado "bloqueado". En este caso, el suceso que permite desbloquear un proceso no es una operación de entrada/salida, sino una señal generada a propósito por el programador desde otro proceso.

Hay cuatro eventos principales que provocan la creación de procesos:

- El arranque del sistema.
- La ejecución, desde un proceso, de una llamada al sistema para la creación de otro proceso.
- Una petición de usuario para crear un proceso.
- El inicio de un trabajo por lotes.

Los procesos pueden contener uno o más hilos, haciendo más eficiente las tareas, asimismo la complejidad de los algoritmos de sincronización, ya que podría ocurrir la condición de carrera muy a menudo, inclusive los indeseados interbloqueos.

Creación de un proceso

Hasta el día de hoy existen sólo 4 formas de crear un proceso:

1. Arranque del sistema.
2. En la ejecución, desde un proceso, de una llamada al sistema para la creación del proceso.
3. Una petición deliberada del usuario para crear un proceso.
4. El inicio de un trabajo por lotes.

La forma de creación de procesos en Unix es a través de una llamada al sistema fork la cual creará un proceso hijo en total semejanza al padre, hasta que el recién proceso decida cambiar su imagen en memoria, incluso obtener sus propios descriptores de archivos abiertos.

Terminación de un proceso

El ciclo de vida de un proceso es fácil, depende de la creación, la ejecución de instrucciones y la terminación. Cabe señalar que un proceso en el transcurso de su ciclo puede estar en diferentes estados.

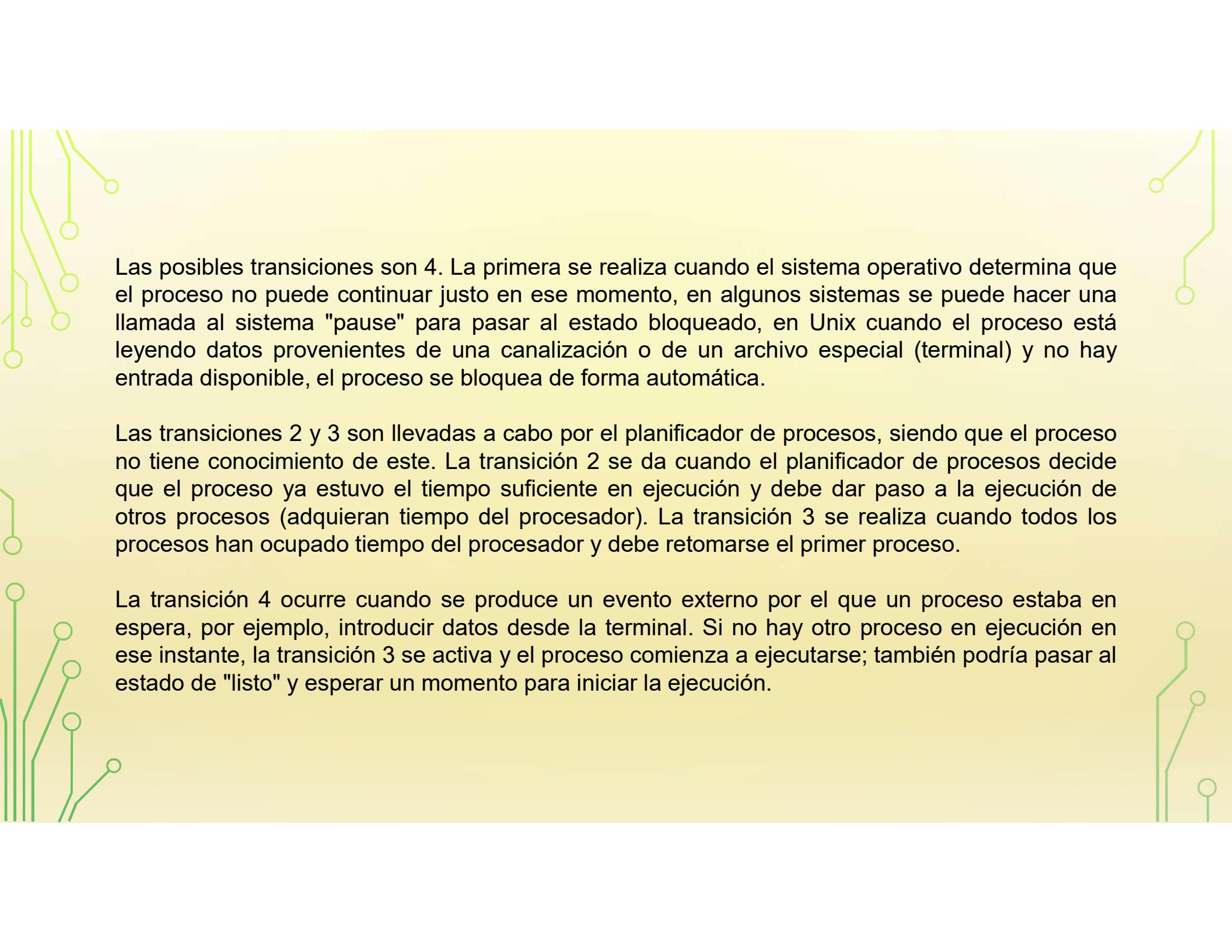
- Salida normal.
- Salida por error.
- Error fatal.
- Eliminado por otro proceso.

Estados de un proceso

Los estados de un proceso obedecen a su participación y disponibilidad dentro del sistema operativo y surgen de la necesidad de controlar la ejecución de cada proceso. Los procesadores sólo pueden ejecutar un solo proceso a la vez, turnándolos para el uso de este. Existen procesos no apropiativos o cooperativos que básicamente ocupan todo el tiempo del procesador hasta que ellos deciden dejarlo. Los procesos apropiativos son aquellos que ocupan por un período de tiempo el procesador hasta que una interrupción o señal llega al procesador para hacer el cambio de proceso, a esto se le conoce como cambio de contexto.

Los posibles estados que puede tener un proceso son ejecución, bloqueado y listo:

- Ejecución, es un proceso que está haciendo uso del procesador.
- Bloqueado, No puede ejecutarse hasta que un evento externo sea llevado a cabo.
- Listo, ha dejado disponible al procesador para que otro proceso pueda ocuparlo.



Las posibles transiciones son 4. La primera se realiza cuando el sistema operativo determina que el proceso no puede continuar justo en ese momento, en algunos sistemas se puede hacer una llamada al sistema "pause" para pasar al estado bloqueado, en Unix cuando el proceso está leyendo datos provenientes de una canalización o de un archivo especial (terminal) y no hay entrada disponible, el proceso se bloquea de forma automática.

Las transiciones 2 y 3 son llevadas a cabo por el planificador de procesos, siendo que el proceso no tiene conocimiento de este. La transición 2 se da cuando el planificador de procesos decide que el proceso ya estuvo el tiempo suficiente en ejecución y debe dar paso a la ejecución de otros procesos (adquieran tiempo del procesador). La transición 3 se realiza cuando todos los procesos han ocupado tiempo del procesador y debe retomarse el primer proceso.

La transición 4 ocurre cuando se produce un evento externo por el que un proceso estaba en espera, por ejemplo, introducir datos desde la terminal. Si no hay otro proceso en ejecución en ese instante, la transición 3 se activa y el proceso comienza a ejecutarse; también podría pasar al estado de "listo" y esperar un momento para iniciar la ejecución.

Tipos de procesos

Existen dos tipos de procesos, aquellos que se ejecutan en modo kernel y aquellos que se ejecutan en modo usuario. Los primeros son más lentos por las llamadas al sistema que realizan, sin embargo, son más seguros por la integridad que representan. Cuando hablamos de los procesos de usuario, podemos decir que el sistema operativo podría no ser multiproceso, ya que se vale de librerías (como pthread) para hacer un multiplexado y dar la apariencia de trabajar como multiproceso.

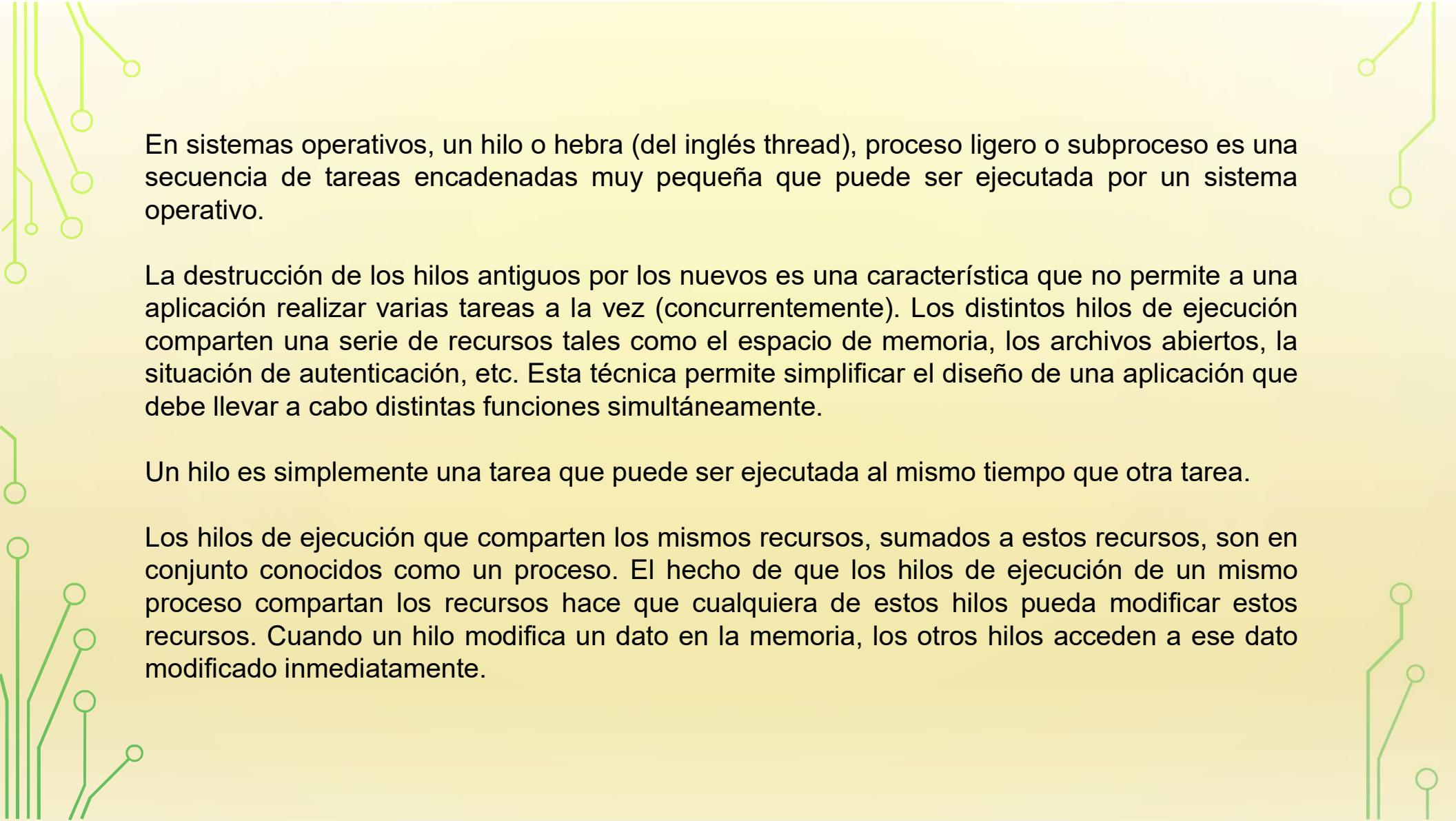
Podría pensarse en otra clasificación, como son los procesos en primer plano y procesos en segundo plano. Los primeros interactúan con el usuario, es decir, el usuario proporciona los datos que el proceso utilizará. Los segundos, son creados para tareas bien definidas y no necesitan la intervención del usuario, por ejemplo, se puede tener un proceso en segundo plano para revisar la temperatura del disco duro constantemente, éstos también son conocidos como demonios.

Para ver los procesos en sistemas Linux, contamos con el comando ‘ ps ’, que listará (de múltiples formas según las opciones que le pasemos) todos los procesos que se encuentran corriendo en nuestro equipo.

- ps aux (muestra todos los procesos del sistema)
- ps axjf (que mostrará un árbol jerárquico con la ruta del programa al que pertenece el proceso)
- ps aux | grep bash
- top
- top –d 5 (Donde 5 es el número de segundos a transcurrir entre cada muestreo)
- top –u toushiro (Donde Toushiro es el usuario del cual queremos mostrar los procesos)
- htop
- kill [PID del proceso]

HILO

```
sergio@sergio-VirtualBox:~$ ps -ef | grep /sbin/init
root      1      0 0 16:07 ?    00:00:02 /sbin/init splash
sergio   6277  5264 0 17:38 pts/0  00:00:00 grep --color=auto /sbin/init
sergio@sergio-VirtualBox:~$
```

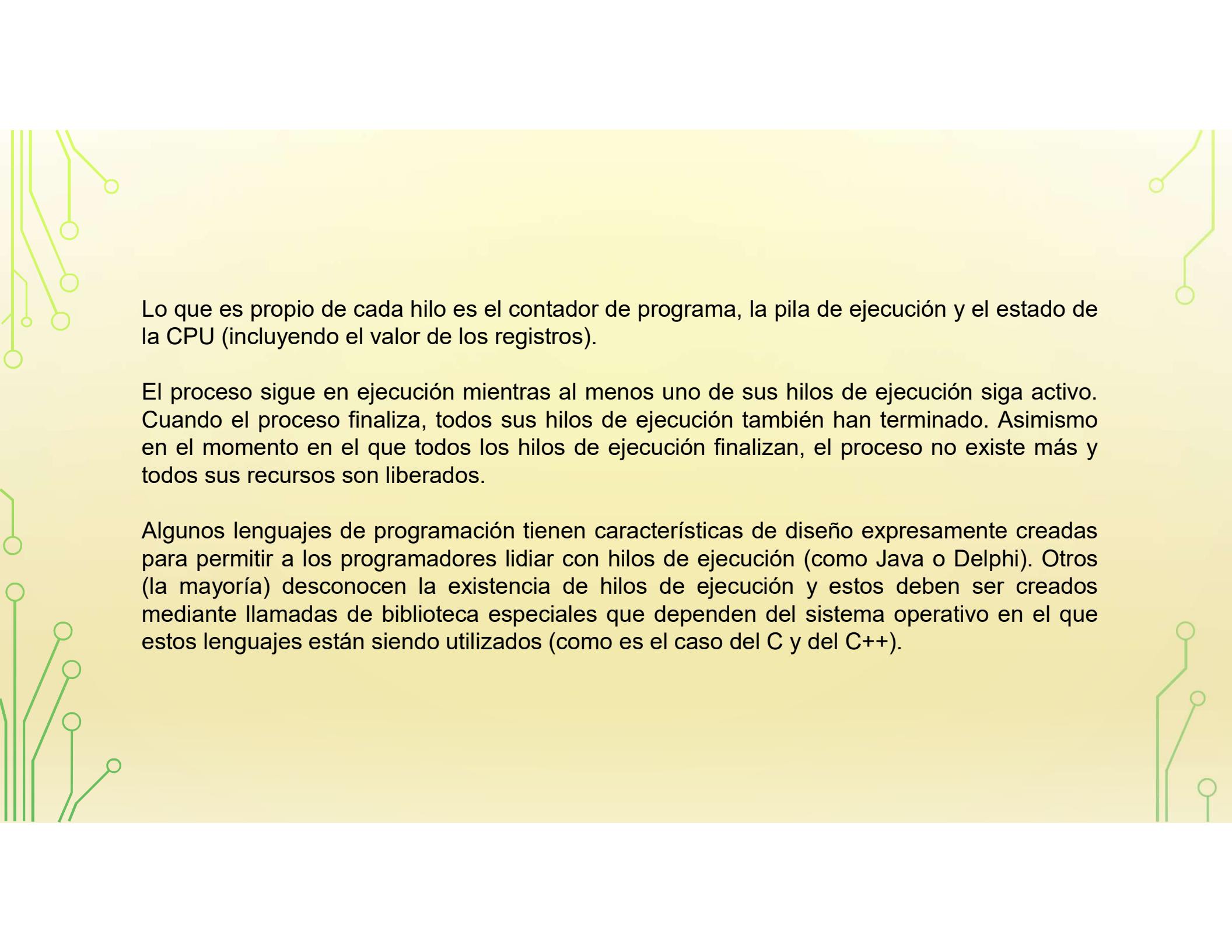


En sistemas operativos, un hilo o hebra (del inglés thread), proceso ligero o subproceso es una secuencia de tareas encadenadas muy pequeña que puede ser ejecutada por un sistema operativo.

La destrucción de los hilos antiguos por los nuevos es una característica que no permite a una aplicación realizar varias tareas a la vez (concurrentemente). Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, la situación de autenticación, etc. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente.

Un hilo es simplemente una tarea que puede ser ejecutada al mismo tiempo que otra tarea.

Los hilos de ejecución que comparten los mismos recursos, sumados a estos recursos, son en conjunto conocidos como un proceso. El hecho de que los hilos de ejecución de un mismo proceso comparten los recursos hace que cualquiera de estos hilos pueda modificar estos recursos. Cuando un hilo modifica un dato en la memoria, los otros hilos acceden a ese dato modificado inmediatamente.



Lo que es propio de cada hilo es el contador de programa, la pila de ejecución y el estado de la CPU (incluyendo el valor de los registros).

El proceso sigue en ejecución mientras al menos uno de sus hilos de ejecución siga activo. Cuando el proceso finaliza, todos sus hilos de ejecución también han terminado. Asimismo en el momento en el que todos los hilos de ejecución finalizan, el proceso no existe más y todos sus recursos son liberados.

Algunos lenguajes de programación tienen características de diseño expresamente creadas para permitir a los programadores lidiar con hilos de ejecución (como Java o Delphi). Otros (la mayoría) desconocen la existencia de hilos de ejecución y estos deben ser creados mediante llamadas de biblioteca especiales que dependen del sistema operativo en el que estos lenguajes están siendo utilizados (como es el caso del C y del C++).

Diferencias entre hilos y procesos

Los hilos se distinguen de los tradicionales procesos en que los procesos son –generalmente– independientes, llevan bastante información de estados, e interactúan solo a través de mecanismos de comunicación dados por el sistema. Por otra parte, muchos hilos generalmente comparten otros recursos de forma directa. En muchos de los sistemas operativos que dan facilidades a los hilos, es más rápido cambiar de un hilo a otro dentro del mismo proceso, que cambiar de un proceso a otro. Este fenómeno se debe a que los hilos comparten datos y espacios de direcciones, mientras que los procesos, al ser independientes, no lo hacen. Al cambiar de un proceso a otro el sistema operativo (mediante el dispatcher) genera lo que se conoce como overhead, que es tiempo desperdiciado por el procesador para realizar un cambio de contexto (context switch), en este caso pasar del estado de ejecución (running) al estado de espera (waiting) y colocar el nuevo proceso en ejecución. En los hilos, como pertenecen a un mismo proceso, al realizar un cambio de hilo el tiempo perdido es casi inapreciable..

Funcionalidad de los hilos

Al igual que los procesos, los hilos poseen un estado de ejecución y pueden sincronizarse entre ellos para evitar problemas de compartición de recursos. Generalmente, cada hilo tiene una tarea específica y determinada, como forma de aumentar la eficiencia del uso del procesador.

Estados de un hilo

Los principales estados de los hilos son: Ejecución, Listo y Bloqueado. No tiene sentido asociar estados de suspensión de hilos ya que es un concepto de proceso. En todo caso, si un proceso está expulsado de la memoria principal (RAM), todos sus hilos deberán estarlo ya que todos comparten el espacio de direcciones del proceso.

Cambio de estados

Creación: Cuando se crea un proceso se crea un hilo para ese proceso. Luego, este hilo puede crear otros hilos dentro del mismo proceso, proporcionando un puntero de instrucción y los argumentos del nuevo hilo. El hilo tendrá su propio contexto y su propio espacio de la columna, y pasará al final de los Listos.

Bloqueo: Cuando un hilo necesita esperar por un suceso, se bloquea (salvando sus registros de usuario, contador de programa y punteros de pila). Ahora el procesador podrá pasar a ejecutar otro hilo que esté al principio de los Listos mientras el anterior permanece bloqueado.

Desbloqueo: Cuando el suceso por el que el hilo se bloqueó se produce, el mismo pasa a la final de los Listos.

Terminación: Cuando un hilo finaliza se liberan tanto su contexto como sus columnas.

Ventajas de los hilos contra procesos

Si bien los hilos son generados a partir de la creación de un proceso, podemos decir que un proceso es un hilo de ejecución, conocido como Monohilo. Pero las ventajas de los hilos se dan cuando hablamos de Multihilos, que es cuando un proceso tiene múltiples hilos de ejecución los cuales realizan actividades distintas, que pueden o no ser cooperativas entre sí. Los beneficios de los hilos se derivan de las implicaciones de rendimiento.

1. Se tarda mucho menos tiempo en crear un hilo nuevo en un proceso existente que en crear un proceso. Algunas investigaciones llevan al resultado que esto es así en un factor de 10.
2. Se tarda mucho menos en terminar un hilo que un proceso, ya que cuando se elimina un proceso se debe eliminar el BCP1 del mismo, mientras que un hilo se elimina su contexto y pila.
3. Se tarda mucho menos tiempo en cambiar entre dos hilos de un mismo proceso.
4. Los hilos aumentan la eficiencia de la comunicación entre programas en ejecución. En la mayoría de los sistemas en la comunicación entre procesos debe intervenir el núcleo para ofrecer protección de los recursos y realizar la comunicación misma. En cambio, entre hilos pueden comunicarse entre sí sin la invocación al núcleo. Por lo tanto, si hay una aplicación que debe implementarse como un conjunto de unidades de ejecución relacionadas, es más eficiente hacerlo con una colección de hilos que con una colección de procesos separados.

Sincronización de hilos

Todos los hilos comparten el mismo espacio de direcciones y otros recursos como pueden ser archivos abiertos. Cualquier modificación de un recurso desde un hilo afecta al entorno del resto de los hilos del mismo proceso. Por lo tanto, es necesario sincronizar la actividad de los distintos hilos para que no interfieran unos con otros o corrompan estructuras de datos.

Una ventaja de la programación multihilo es que los programas operan con mayor velocidad en sistemas de computadores con múltiples CPUs (sistemas multiprocesador o a través de grupo de máquinas) ya que los hilos del programa se prestan verdaderamente para la ejecución concurrente. En tal caso el programador necesita ser cuidadoso para evitar condiciones de carrera (problema que sucede cuando diferentes hilos o procesos alteran datos que otros también están usando), y otros comportamientos no intuitivos. Los hilos generalmente requieren reunirse para procesar los datos en el orden correcto. Es posible que los hilos requieran de operaciones atómicas para impedir que los datos comunes sean cambiados o leídos mientras estén siendo modificados, para lo que usualmente se utilizan los semáforos. El descuido de esto puede generar interbloqueo.

Formas de multihilos

Los sistemas operativos generalmente implementan hilos de dos maneras:

Multihilo apropiativo: permite al sistema operativo determinar cuándo debe haber un cambio de contexto. La desventaja de esto es que el sistema puede hacer un cambio de contexto en un momento inadecuado, causando un fenómeno conocido como inversión de prioridades y otros problemas.

Multihilo cooperativo: depende del mismo hilo abandonar el control cuando llega a un punto de detención, lo cual puede traer problemas cuando el hilo espera la disponibilidad de un recurso.

El soporte de hardware para multihilo se encuentra disponible desde hace mucho tiempo, en los 386 por ejemplo http://en.wikipedia.org/wiki/Compaq_SystemPro. Hace relativamente poco tiempo esta característica es utilizada por el gran público, soportada nativamente por los Intel en el Pentium Pro y los Pentium II y III en la versión doméstica. Fue eliminada en los Celeron al descubrirse que podía ser desbloqueado y posteriormente reintroducido en el Pentium 4, bajo el nombre de HyperThreading.

Usos más comunes

Los usos más comunes son en tecnologías SMPP y SMS para la telecomunicaciones aquí hay muchísimos procesos corriendo a la vez y todos requiriendo de un servicio.

Trabajo interactivo y en segundo plano

Por ejemplo, en un programa de hoja de cálculo un hilo puede estar visualizando los menús y leer la entrada del usuario mientras que otro hilo ejecuta las órdenes y actualiza la hoja de cálculo. Esta medida suele aumentar la velocidad que se percibe en la aplicación, permitiendo que el programa pida la orden siguiente antes de terminar la anterior.

Procesamiento asíncrono

Los elementos asíncronos de un programa se pueden implementar como hilos. Un ejemplo es cómo los software de procesamiento de texto guardan archivos temporales cuando se está trabajando en dicho programa. Se crea un hilo que tiene como función guardar una copia de respaldo mientras se continua con la operación de escritura por el usuario sin interferir en la misma. Son como 2 programas independientes.

Aceleración de la ejecución

Se pueden ejecutar, por ejemplo, un lote mientras otro hilo lee el lote siguiente de un dispositivo.

Estructuración modular de los programas

Puede ser un mecanismo eficiente para un programa que ejecuta una gran variedad de actividades, teniendo las mismas bien separadas mediante hilos que realizan cada una de ellas.

Implementaciones

Hay dos grandes categorías en la implementación de hilos:

- Hilos a nivel de usuario.
- Hilos a nivel de núcleo.

También conocidos como ULT (user level thread) y KLT (kernel level thread).

Hilos a nivel de usuario (ULT)

En una aplicación ULT pura, todo el trabajo de gestión de hilos lo realiza la aplicación, y el núcleo o kernel no es consciente de la existencia de hilos. Es posible programar una aplicación como multihilo mediante una biblioteca de hilos. La misma contiene el código para crear y destruir hilos, intercambiar mensajes y datos entre hilos, para planificar la ejecución de hilos y para salvar y restaurar el contexto de los hilos.

Todas las operaciones descritas se llevan a cabo en el espacio de usuario de un mismo proceso. El núcleo continua planificando el proceso como una unidad y asignándole un único estado (Listo, bloqueado, etc.)

Ventajas de los ULT

- El intercambio de los hilos no necesita los privilegios del modo núcleo, porque todas las estructuras de datos están en el espacio de direcciones de usuario de un mismo proceso. Por lo tanto, el proceso no debe cambiar a modo núcleo para gestionar hilos. Se evita la sobrecarga de cambio de modo y con esto el sobrecoste u overhead.
- Se puede realizar una planificación específica. Dependiendo de que aplicación sea, se puede decidir por una u otra planificación según sus ventajas.
- Los ULT pueden ejecutar en cualquier sistema operativo. La biblioteca de hilos es un conjunto compartido.

Desventajas de los ULT

- En la mayoría de los sistemas operativos las llamadas al sistema (System calls) son bloqueantes. Cuando un hilo realiza una llamada al sistema, se bloquea el mismo y también el resto de los hilos del proceso.
- En una estrategia ULT pura, una aplicación multihilo no puede aprovechar las ventajas de los multiprocesadores. El núcleo asigna un solo proceso a un solo procesador, ya que como el núcleo no interviene, ve al conjunto de hilos como un solo proceso.

Una solución al bloqueo mediante a llamadas al sistema es usando la técnica de jacketing, que es convertir una llamada bloqueante en no bloqueante; esto se consigue comprobando previamente si la llamada al sistema bloqueará el proceso o no. Si es así, se bloquea el hilo y se pasa el control a otro hilo. Más adelante se reactiva el hilo bloqueado y se vuelve a realizar la comprobación, hasta que se pueda realizar la llamada al sistema sin que el proceso completo sea bloqueado.

Hilos a nivel de núcleo (KLT)

En una aplicación KLT pura, todo el trabajo de gestión de hilos lo realiza el núcleo. En el área de la aplicación no hay código de gestión de hilos, únicamente un API (interfaz de programas de aplicación) para la gestión de hilos en el núcleo. Windows 2000, Linux y OS/2 utilizan este método. Linux utiliza un método muy particular en el que no hace diferencia entre procesos e hilos. Para Linux, si varios procesos creados con la llamada al sistema "clone" comparten el mismo espacio de direcciones virtuales, el sistema operativo los trata como hilos, y lógicamente son manejados por el núcleo.

Ventajas de los KLT

- El núcleo puede planificar simultáneamente múltiples hilos del mismo proceso en múltiples procesadores.
- Si se bloquea un hilo, puede planificar otro del mismo proceso.
- Las propias funciones del núcleo pueden ser multihilo.

Desventajas de los KLT

- El paso de control de un hilo a otro precisa de un cambio de modo a modo núcleo.

Combinaciones ULT y KLT

Algunas distribuciones de linux y derivados de UNIX ofrecen la combinación de ULT y KLT, como Solaris, Ubuntu y Fedora.

La creación de hilos, así como la mayor parte de la planificación y sincronización de los hilos de una aplicación se realiza por completo en el espacio de usuario. Los múltiples ULT de una sola aplicación se asocian con varios KLT. El programador puede ajustar el número de KLT para cada aplicación y máquina para obtener el mejor resultado global.

En un método combinado , los múltiples hilos de una aplicación se pueden ejecutar en paralelo en múltiples procesadores y las llamadas al sistema bloqueadoras no necesitan bloquear todo el proceso.

Saber el número de hilos usados por un proceso en Linux
Aquí dejo la lista de comandos que uso normalmente para esto:

para conocer el PID:

`ps -ef | grep myprocess` Ejemplo: `ps -ef | grep /sbin/init`
Monitorizar esos threads en TOP:

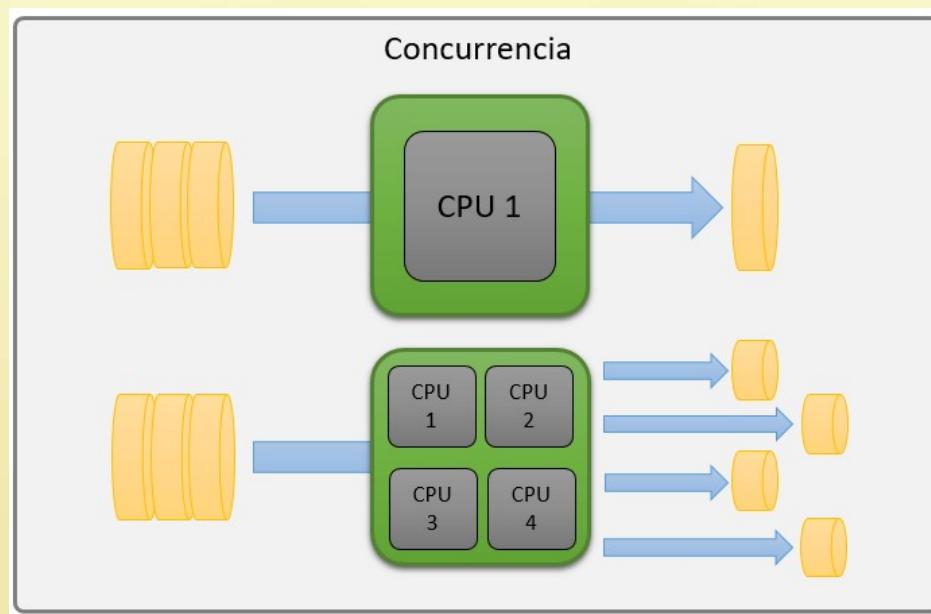
`top -p (PID) -H` Ejemplo: `top 1 -p -H`
Averiguar el número de hilos usados por ese proceso.

`cat /proc/(PID)/status | grep Threads` Ejemplo: `cat /proc/1/status | grep Threads`

SISTEMAS OPERATIVOS 2

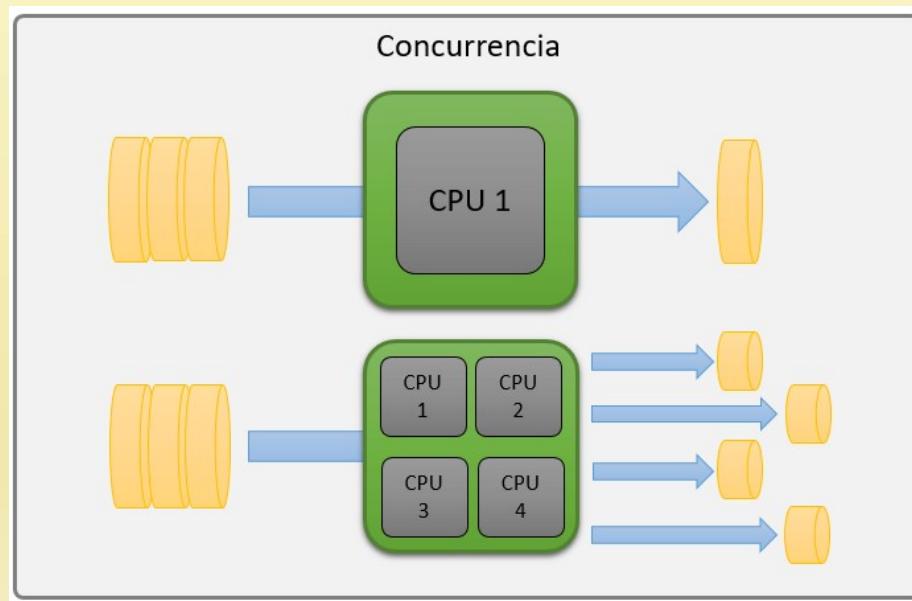
LABORATORIO

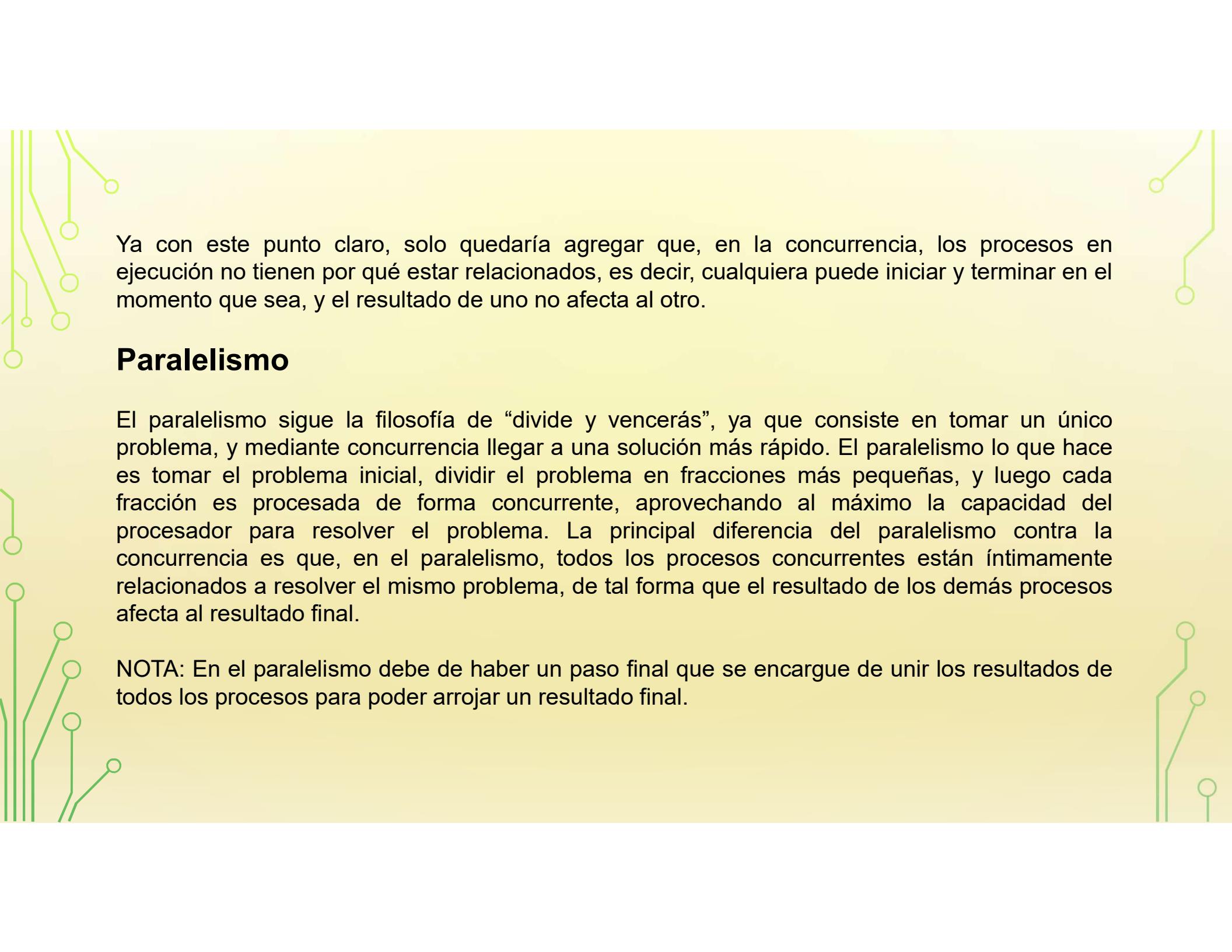
CONCURRENCIA



Concurrencia

La concurrencia es la capacidad del CPU para procesar más de un proceso al mismo tiempo, ¿simple no? Pero que implica esto realmente. Pues bien, para comprender esto es necesario entender cómo funciona un procesador. Un procesador puede procesar al mismo tiempo el mismo número de procesos que el número de CORES que tiene, de esta forma, si un procesador tiene un CORE, entonces solo podrá ejecutar un proceso a la vez, por otro parte, si tenemos 8 CORES, entonces podremos ejecutar hasta 8 procesos al mismo tiempo.





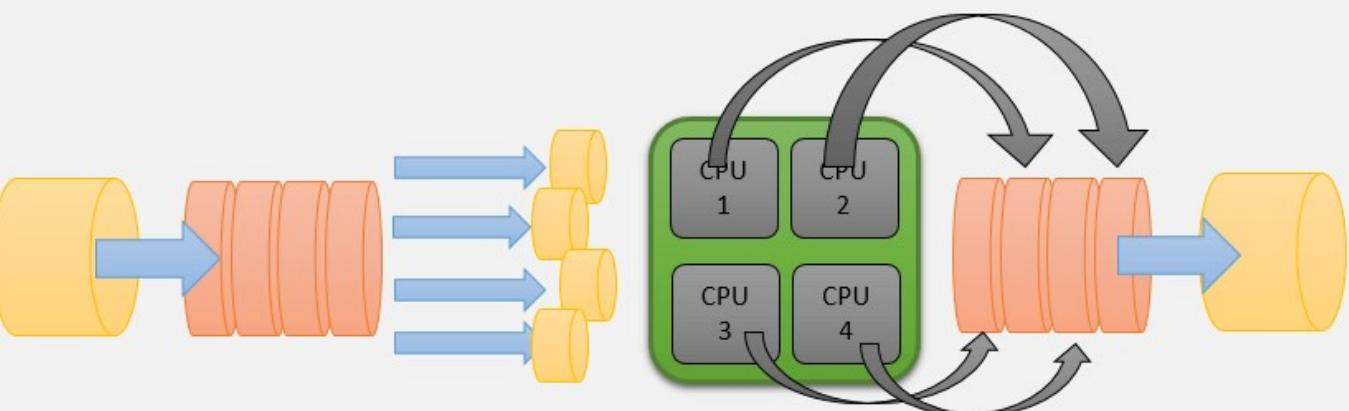
Ya con este punto claro, solo quedaría agregar que, en la concurrencia, los procesos en ejecución no tienen por qué estar relacionados, es decir, cualquiera puede iniciar y terminar en el momento que sea, y el resultado de uno no afecta al otro.

Paralelismo

El paralelismo sigue la filosofía de “divide y vencerás”, ya que consiste en tomar un único problema, y mediante concurrencia llegar a una solución más rápido. El paralelismo lo que hace es tomar el problema inicial, dividir el problema en fracciones más pequeñas, y luego cada fracción es procesada de forma concurrente, aprovechando al máximo la capacidad del procesador para resolver el problema. La principal diferencia del paralelismo contra la concurrencia es que, en el paralelismo, todos los procesos concurrentes están íntimamente relacionados a resolver el mismo problema, de tal forma que el resultado de los demás procesos afecta al resultado final.

NOTA: En el paralelismo debe de haber un paso final que se encargue de unir los resultados de todos los procesos para poder arrojar un resultado final.

Paralelismo



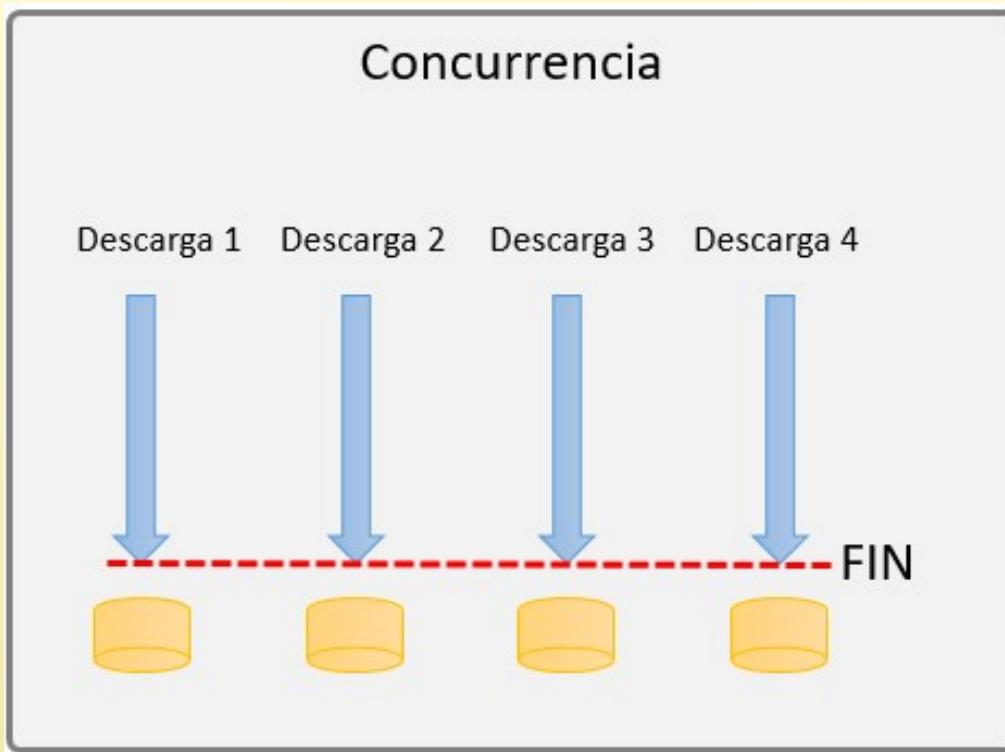
Puede resultas complicado entenderlo a primera vista, pero observemos la imagen anterior. Primero que nada, llega un proceso grande (Amarillo), el cual es divididos en cuatro partes(Naranja), las cuales coinciden con el número de CORES del procesador(Verde). Ahora bien, por cada CORE disponible se lanza un proceso concurrente para resolver una fracción del problema. Cuando el proceso de un CORE termina, deja su resultado como una parte del resultado final (Naranja), así al terminar todos los procesos concurrentes, tenemos un resultado final (Amarillo).

Escenario del mundo real:

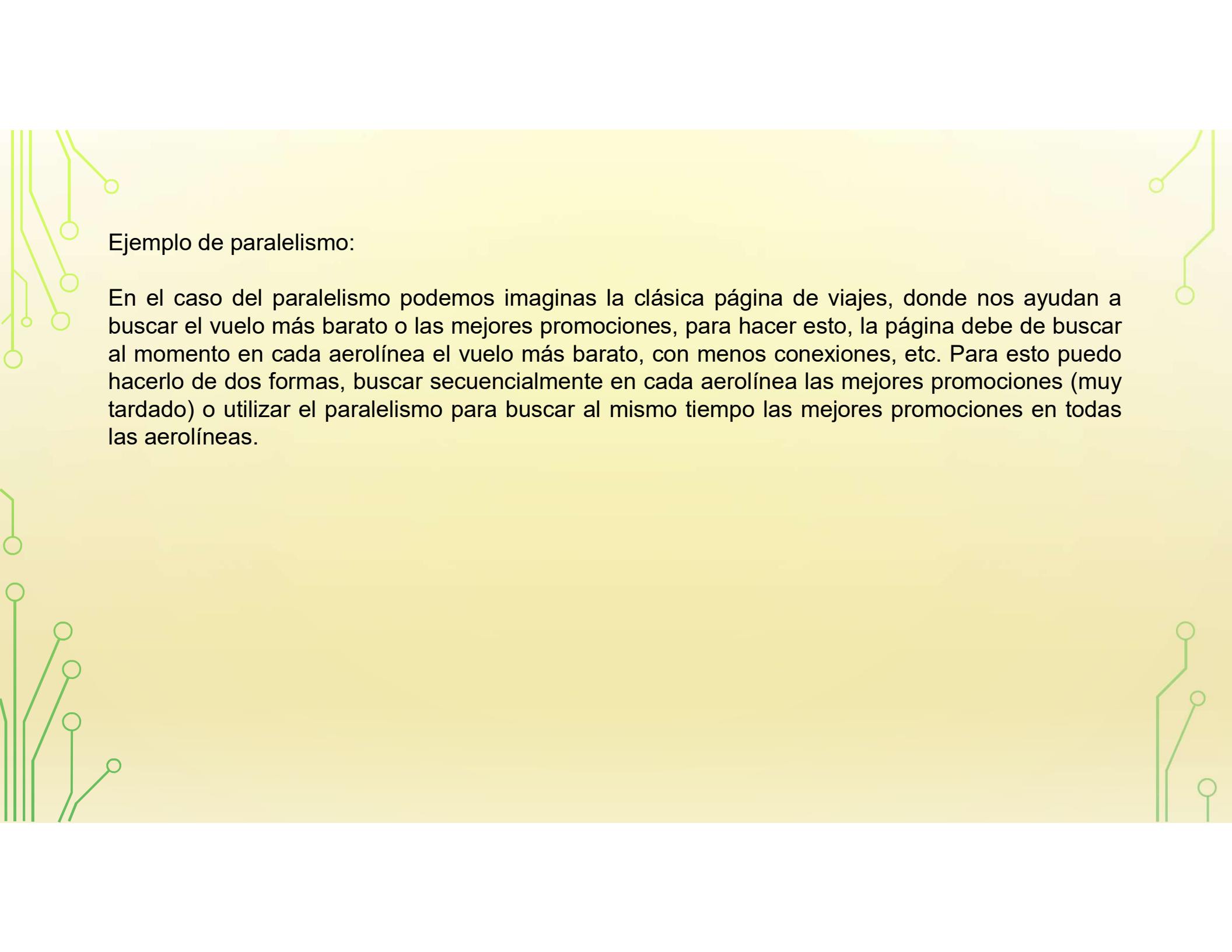
Para comprender mejor como es que funcionan estos dos conceptos veremos un ejemplo simple de cada uno.

Ejemplo de concurrencia:

Empecemos con la concurrencia, imagina una aplicación de descarga de música, en la cual puedes descargar un número determinado de canciones al mismo tiempo, cada canción es independiente de la otra, por lo que la velocidad y el tiempo que tarde en descargarse cada una no afectara al resto de canciones. Esto lo podemos ver como un proceso concurrente, ya que cada descarga es un proceso totalmente independiente del resto.



Observa la imagen, cada descarga se procesa de forma separada, y al final a una descarga no le importa el estado de las demás. Ya que cada descarga es una tarea completamente diferente.



Ejemplo de paralelismo:

En el caso del paralelismo podemos imaginar la clásica página de viajes, donde nos ayudan a buscar el vuelo más barato o las mejores promociones, para hacer esto, la página debe de buscar al momento en cada aerolínea el vuelo más barato, con menos conexiones, etc. Para esto puedo hacerlo de dos formas, buscar secuencialmente en cada aerolínea las mejores promociones (muy tardado) o utilizar el paralelismo para buscar al mismo tiempo las mejores promociones en todas las aerolíneas.