UT01. TEORÍA DE SISTEMAS OPERATIVOS

Contenidos

1.- INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS

Un **sistema operativo** es un programa que controla la ejecución de los programas y actúa como interfaz entre el usuario de un computador y el hardware de este.

El sistema operativo tiene dos funciones principales:

- **Extender la máquina**: si vemos un ordenador a muy bajo nivel podremos apreciar que su arquitectura (conjunto de instrucciones, organización, E/S y estructura de bus) es primitiva y complicada de programar. Como veremos, para permitir al usuario gestionar esta complejidad hará uso de abstracciones que mostrarán una vista más sencilla de la máquina subyacente al usuario.
- **Gestionar los recursos**: cuando hablamos de recursos nos referimos a procesadores, memorias, unidades de almacenamiento, ... Cada programa y usuario necesita acceder a estos recursos y será el sistema operativo quien se encargue de repartirlos de la forma más equilibrada y óptima posible.

De estas dos funciones se pueden inferir tres objetivos que el sistema operativo pretende conseguir:

- **Comodidad**: el sistema operativo pretende que el usuario tenga una interacción más natural o sencilla con la máquina. Para ello abstrae todas las complejidades de la máquina subyacente mediante una interfaz de usuario que sea sencilla y manejable para el usuario.
- **Eficiencia**: el sistema operativo gestionará los recursos de la máquina de la forma más optima posible para dar servicio a los usuarios y aprovecharlos de la manera más eficiente.
- **Capacidad de evolución**: al igual que el hardware evoluciona con el tiempo, el sistema operativo debe evolucionar de forma paralela.

Veamos con más detalle estos tres objetivos:

1.1- El sistema operativo como interfaz usuario/computadora

El primer objetivo del sistema operativo es simplificar el manejo del ordenador para el usuario. A muy bajo nivel, un ordenador es una máquina muy compleja cuyo funcionamiento no tiene que conocer el usuario. Para simplificar el manejo, el sistema operativo crea una serie de capas de **abstracciones**, en las que el sistema operativo enmascara el trabajo a bajo nivel con el equipo mediante conceptos más familiares para el usuario.

Por ejemplo, a bajo nivel, un disco duro magnético es una superficie magnetizable que tiene la capacidad de almacenar bits que son accesibles mediante el movimiento de un cabezal magnético. Pero los sistemas operativos crean una abstracción mediante la que vemos todo ese almacenamiento como una serie de archivos y carpetas. Cuando el usuario abre un archivo, no necesita saber nada del funcionamiento interno del disco duro, ni de movimientos del cabezal, simplemente hace doble click en el icono que representa dicho archivo.

1.2.- Eficiencia en la gestión de recursos

Un ordenador tiene una serie de recursos (CPU, memoria, periféricos, ...) cuyo uso debe compartir entre todos los procesos que están en ejecución en el sistema. Esto es más importante incluso en sistemas multiusuario en los que el sistema operativo debe repartir los recursos no solo entre procesos, sino también entre usuarios. Es su labor realizar un reparto ordenado de dichos recursos, llevando un control de quien está usando cada recurso, concediendo solicitudes de uso de estos y mediando entre diferentes procesos y usuarios que estén en conflicto. La compartición o multiplexaje de recursos se puede realizar de dos formas en función del recurso:

- **Multiplexaje en el tiempo**: el recurso únicamente puede ser utilizado por un proceso en un momento determinado, por lo que se deben turnar para ir utilizándolo. Un ejemplo de este tipo de multiplexaje lo podemos ver en la CPU, que solo puede ser utilizada por un proceso en cada momento.
- **Multiplexaje en el espacio**: en este caso, cada proceso recibe una parte del recurso. El ejemplo más claro es la memoria RAM, donde cada proceso tendrá un espacio asignado por el sistema operativo.

1.3.- Facilidad de evolución del sistema operativo

Hay varias razones por las que un sistema operativo debe estar en continua evolución:

- **Actualizaciones de hardware y nuevos tipos de hardware**: el hardware está evolucionando constantemente, y, además, pueden aparecer nuevos tipos de hardware (por ejemplo, cuando aparecieron los primeros discos de estado sólido). El sistema operativo debe disponer de mecanismos para adaptarse a este hardware y obtener su máximo rendimiento. Habitualmente, esto se consigue mediante los drivers o controladores.
- **Nuevos servicios**: los sistemas operativos deben ir añadiendo nuevos servicios para facilitar la labor de los usuarios.
- **Correcciones**: un sistema operativo es un programa muy complejo con millones de líneas de código, por lo que es habitual que con el tiempo se detecten fallos y problemas de seguridad. Por tanto, será necesario ir solucionando estos bugs a medida que se detecten mediante parches y actualizaciones.

2.- HISTORIA DE LOS SISTEMAS OPERATIVOS

La historia de los ordenadores se descompone en generaciones, que corresponden con grandes avances tecnológicos que han supuesto un punto de inflexión en su evolución. Estas generaciones han sido:

- **Primera generación (1945-1955)**: primeros ordenadores que utilizaban válvulas de vacío.
- **Segunda generación (1955-1965)**: aparece el transistor, que motiva la aparición de los denominados *mainframes*. Estos ordenadores empiezan a ser accesibles para las grandes multinacionales.
- **Tercera generación (1965-1985)**: con el circuito integrado se reduce aún más el tamaño de los ordenadores, apareciendo los *minicomputadores*, accesibles ya para prácticamente cualquier empresa.
- **Cuarta generación (1985-hoy)**: con la integración a escala muy grande (**VLSI**), los circuitos integrados pueden contener millones o incluso miles de millones de transistores, lo que motiva la aparición de los ordenadores personales. En esta generación, cada persona puede tener su propio ordenador.

De forma paralela a la evolución de los ordenadores así lo han hecho los sistemas operativos que se ejecutan sobre ellos, pudiendo establecer una relación entre las distintas generaciones de ordenadores y los sistemas operativos que han ido surgiendo a lo largo de estos 75 años.

2.1.- La primera generación (1945-1955): tubos de vacío y tableros

A mediados de la década de 1940 *Howard Aiken*, en Harvard; *John von Neumman* en Princeton; *Eckert y Mauchely* en Pensilvania y *Konrad Zuse* en Alemania, entre otros, lograron construir máquinas calculadoras. Las primeras empleaban lentos reveladores mecánicas que rápidamente fueron sustituidos por tubos de vacío, lo que dio lugar a los primeros ordenadores.

En estos primeros tiempos un solo grupo de personas diseñaba, construía, programaba, operaba y mantenía cada máquina. Toda la programación se efectuaba en lenguaje de máquina absoluto, a menudo alambrando tableros de conexiones para controlar las funciones básicas de la máquina. Por lo tanto, **no existían sistemas operativos** y los usuarios debían conocer perfectamente el funcionamiento interno de las máquinas.

2.2.- La segunda generación (1955-1965): transistores y sistemas por lotes

Con la introducción del **transistor** las computadoras se volvieron lo bastante fiables como para fabricarse y venderse a clientes comerciales. Por primera vez hubo una distinción clara entre diseñadores, constructores, operadores, programadores y personal de mantenimiento.

Un concepto de esta época eran los **sistemas por lotes**, que consistía en ejecutar una serie de programas de forma secuencial. Y este es precisamente el cometido de los sistemas operativos surgidos en esta época. Por ejemplo, **GM-NAA I/O**, desarrollado por General Motors para el IBM 704, era un sistema operativo cuya única función era ejecutar programas por lotes, es decir, uno tras otro.

Otro sistema operativo de la época fue el **IBSYS** para el IBM 7094. Este sistema operativo se caracterizaba porque incluía algunas utilidades tales como compiladores de FORTRAN y COBOL, un ensamblador y algunos programas, como, por ejemplo, un programa de ordenación.

Otro sistema operativo es **FMS (FORTRAN Monitoring System)**, un sistema operativo para el IBM 7090 basado en cinta cuyo propósito era únicamente compilar programas en FORTRAN.

2.3.- La tercera generación (1965-80): circuitos integrados y multiprogramación

A principios de los 60 cada ordenador tenía su propio sistema operativo que únicamente funcionaba en dicho ordenador, lo que hacía incompatibles entre sí todos los sistemas operativos, incluso para ordenadores de un mismo fabricante.

IBM intentó resolver ambos problemas con la introducción del **System/360**, una familia de ordenadores que incluía desde pequeñas computadoras hasta grandes mainframes. Lo relevante de esta familia de ordenadores es que compartían la misma **arquitectura**, difiriendo únicamente en el desempeño y el precio. El utilizar la misma arquitectura permitió usar el mismo sistema operativo en todos ellos, denominado **OS/360**, un sistema enorme y complejo, pero que permitía algún tan común actualmente como intercambiar programas y datos entre diferentes ordenadores, es decir, la **compatibilidad** entre sistemas y la **portabilidad** de aplicaciones.

Además, este sistema introdujo técnicas tan comunes en la actualidad como:

- **Multiprogramación**: hasta este momento, los programas solo se podían ejecutar de uno en uno. Cuando se estaba ejecutando un programa había que esperar a que finalizara su ejecución antes de dar paso a otro programa. Esto era muy ineficiente porque los programas habitualmente tienen largos tiempos de espera en los que no hacen nada (por ejemplo, cuando envían una solicitud de lectura a un disco duro o cuando esperan algún tipo de entrada del usuario por teclado). La multiprogramación soluciona este problema permitiendo que varios programas se alojen en memoria y ejecutarlos alternativamente aprovechando los tiempos en que estén esperando algún evento.
- **Spooling (*Simultaneous Paripheral Operation Online*)**: que permitía almacenar los trabajos en disco de forma que nada más finalizar uno se pudiera comenzar a ejecutar el siguiente.

En esta generación también apareció el **tiempo compartido**, una variante de la multiprogramación, en la que cada usuario tiene una terminal en línea desde la que pueden trabajar simultáneamente, incluso mientras la computadora trabajaba en segundo plano con trabajos de lotes grandes aprovechando los periodos inactivos de la CPU. El primer sistema de tiempo compartido fue el **CTSS (Compatible Time Sharing System)**, desarrollado en el MIT para la máquina 7094.

Después del éxito del CTSS, MIT, Bell Labs y General Electric decidieron emprender el desarrollo de un "*servicio de computadora*", una máquina que atendiera a cientos de usuarios por tiempo compartido en un sistema que se podría considerar análogo al sistema de telefonía fija. Este sistema se llamó **MULTICS (MULTiplexed Information and Computing System)**.

Desde el punto de vista comercial, MULTICS tuvo un éxito muy limitado, aunque es cierto que sus usuarios eran muy fieles al sistema, existiendo ordenadores con este sistema operativo instalado hasta mediados de los 90. Sin embargo, este sistema tuvo una gran relevancia en la historia de los sistemas operativos ya que algunas de sus características influyeron enormemente en posteriores sistemas operativos que han acabado desembocando en los sistemas operativos actuales.

Otro adelanto durante la tercera generación fue el gran crecimiento de las **minicomputadoras**, como la serie **DEC-PDP**. En una de estas minicomputadoras, junto con MULTICS, podemos marcar el germen que ha acabado desembocado en los sistemas operativos actuales. Esto hecho vino de la mano de dos científicos de computación de Bell Labs que había trabajado en el proyecto MULTICS, **Ken Thompson** y **Dennis Ritchie**. Ambos usaron una vieja DEC PDP-7 que nadie estaba usando para crear un lenguaje de programación denominado **C** (precursor de prácticamente todos los lenguajes de programación que hay en la actualidad) y un sistema operativo que pretendía ser una versión austera de MULTICS para un único usuario. A este sistema lo llamaron **UNIX**.

El sistema operativo UNIX tuvo una rápida expansión entre universidades, donde gran número de colaboradores contribuyeron a ampliarlo y mejorarlo. Con el tiempo se escindió en dos versiones: el **System V de AT&T** y el **BSD de la Universidad de Berkeley**.

El System V ha dado lugar a las versiones comerciales de UNIX de la actualidad, como el IBM AIX, HP-UX o Solaris de Oracle. En cambio, BSD evolución en otro sentido, hacia versiones de software libre, como NetBSD, FreeBSD y OpenBSD; y también en un sistema operativo ampliamente conocido en la actualidad: el Mac OS X de Apple.

2.4.- La cuarta generación (1980-1995): computadoras personales

Esta generación está caracterizada por los ordenadores personales y, con ellos, una gran variedad de sistemas operativos que, en muchos casos, establecieron las bases para los sistemas operativos actuales.

2.4.1.- Primeros sistemas operativos orientados a computadoras personales

El primer sistema operativo para microcomputadoras fue **CP/M (Control Program for Microcomputers)**, desarrollado por Gary Kildall para el procesador Intel 8080. Posteriormente fue reescrito para otros procesadores, como el Zilog Z80 o el Motorola 68000, lo que le permitió dominar el mundo de la microcomputación durante cinco años. Algunos ordenadores que utilizaron este sistema operativo son el ZX Spectrum, Atari ST, MSX o el Apple II.

El otro gran sistema operativo de la época fue el **MS-DOS**. Este sistema tiene su origen en el sistema DOS (Disk Operating System) de Seatle Computer Products. Este fue adquirido por Bill Gates, que lo renombró y le añadió el intérprete BASIC de Microsoft para su uso en el PC original de IBM. Este sistema pronto dominó el mercado de PCs. Un factor clave fue la decisión de vender el MS-DOS a compañías de computadoras para incluirlo con el hardware, en lugar de venderlo directamente al usuario como se hacía con CP/M.

Hasta este momento, la interfaz de usuario de todos los sistemas operativos consistía en una terminal de texto en la que el usuario introducía comandos u órdenes. Pero eso cambió gracias a las investigaciones hechas por **Doug Engelbart** en el Stanford Research Institute donde inventó la **GUI (Graphical User Interface)**, provista de ventanas, iconos, menús y ratón. Los investigadores de Xerox PARC adoptaron estas ideas y las incorporaron en las máquinas que fabricaban, aunque no les llegaron a dar aplicación comercial ya que no fueron capaces de ver su potencial. Quien sí lo vio fue Steve Jobs después de una visita a Xerox PARC, lo que llevó a la construcción de Lisa, la primera computadora con GUI, que fracasó por ser demasiado cara. El segundo intento, la **Apple Macintosh**, fue un enorme éxito, no sólo porque era más económica, sino porque era amigable para el usuario, lo cual la hacía apta para usuarios sin conocimientos de informática.

2.4.2.- MS-DOS y Windows

Cuando Microsoft buscó un sucesor para MS-DOS en 1985 se basó en el éxito de la Macintosh, por lo que produjo un sistema basado en GUI que llamó **Windows** y que originalmente se ejecutaba encima de MS-DOS, es decir, era un **shell** y no un verdadero sistema operativo. En 1995 Microsoft sacó **Windows 95**, que ya se podía considerar un sistema operativo que integraba Windows y MS-DOS. Este evolucionó con Windows 98 y posteriormente con el malogrado **Windows Me (Millenium Edition)**.

Pero estos sistemas seguían estando construidos sobre MS-DOS, lo que acarreaba múltiples deficiencias y fallos, lo que se puso de manifiesto sobre todo con Windows Me. Por ello, Microsoft reescribió Windows desde cero y creó un sistema operativo gráfico de 32 bits que denominó **Windows NT** y que posteriormente evolucionó hacia **Windows 2000**.

El objetivo de Microsoft era que todos los usuarios de Windows 95, 98 y Me se pasaran a este sistema, ya que era un sistema muy superior y mucho más estable, pero no llegó a conseguirlo, quedando relegado a

entornos corporativos. Microsoft siguió intentándolo y llegó al que sería su gran éxito, el sistema operativo que combinaba la estabilidad de Windows 2000 con la familiaridad para el usuario de Windows Me. Este sistema se denominó **Windows XP** y llegó a vender más de 1000 millones de copias hasta su desaparición en 2014.

En 2005, cuando se empezó a hacer patente que Windows XP necesitaba ceder el paso a otro sistema operativo más evolucionado, Microsoft anunció que estaba desarrollando el sucesor de XP, denominado con el nombre clave Longhorn. Este sistema operativo prometía un gran número de funcionalidades totalmente revolucionarias, pero, a medida que fue avanzando su desarrollo, diversos problemas hicieron que estas fantásticas funcionalidades prometidas se esfumaran de forma que el producto que finalmente apareció, denominado **Windows Vista**, fue un auténtico fracaso.

Microsoft rápidamente sacó un nuevo sistema operativo, denominado **Windows 7** que es el que consiguió que los usuarios abandonaran Windows XP. Tras un intento fallido con Windows 8 llegamos al sistema operativo más utilizado en la actualidad, **Windows 10**.

Paralelamente a todos estos sistemas orientados al mercado doméstico, Microsoft han mantenido una familia de sistemas operativos orientados al mercado de servidores, denominados **Windows Server**.

2.4.3.- Linux

El desarrollo de **Linux** es, en cierto modo, paralelo al de Windows, y su origen tiene un nombre propio y una fecha: **Richard Stallman**, un programador que, en 1983, ante un entorno tecnológico cada vez más comercializado, añoraba los viejos tiempos en los que el software era creado en universidades y no tenía propietario, los programas se distribuían libremente y cualquiera con conocimientos podía contribuir a su desarrollo. Por ello, en ese año decidió acuñar el concepto de **software libre**, el cual se resume en cuatro libertades esenciales:

- Libertad de ejecutar un programa para cualquier propósito.
- Libertad para estudiar como funciona un programa, para lo que es necesario, por tanto, el acceso al código fuente.
- Libertad de redistribuir copias de un programa.
- Libertad de distribuir copias de sus versiones modificadas a terceros.

Además, al tiempo que acuñó este concepto, inició el **proyecto GNU** (GNU is not Unix), con el que pretendía crear un sistema operativo similar y compatible con Unix, pero de software libre. En su camino hacia este objetivo, en 1985 creó la Fundación del Software Libre (FSF) y desarrolló la Licencia General de GNU (GNU GPL)

A mediados de los 90 había avanzado mucho en su objetivo de construir un sistema operativo libre. Tenía un gran número de utilidades, pero se había encontrado con grandes problemas al diseñar el componente más importante de un sistema operativo: el kernel, que es el componente encargado de comunicarse directamente con el hardware.

Y aquí dejamos a Richard Stallman para viajar a Holanda, allí, un profesor de Universidad llamado **Andrew Tannenbaum** había escrito un libro (*Operating Systems: Design and Implementation*) orientado a sus alumnos de sistemas operativos donde desarrollaba la construcción de un sistema operativo, al que denominaba Minix por ser una versión muy limitada de Unix. Este libro tuvo una gran repercusión entre estudiantes de todo el mundo. Uno de estos estudiantes fue **Linux Torvald**¸ un estudiante finés que decidió partir de Minix para

crear un sistema operativo más funcional. El 25 de agosto de 1991 publicó un mensaje en Usenet anunciando la creación de este núcleo y dejando el proyecto a la colaboración de cualquiera que quiera aportar algo. Este llamamiento tuvo un éxito enorme y muy pronto había un gran número de colaboradores contribuyendo a este sistema, al cual decidió llamar **Linux**.

Aunque al principio tenía una licencia propia que restringía la actividad comercial, en 1992 lo vinculó a la licencia GNU GPL que había desarrollado Richard Stallman, lo que permitió completar aquello que faltaba en el proyecto GNU: el kernel. El resultado es el sistema operativo que conocemos actualmente como **GNU/Linux**.

3.- COMPONENTES DE LOS SISTEMAS OPERATIVOS

Por norma general, se suele considerar que un sistema operativo está formado por tres capas: el núcleo o kernel, los servicios del sistema operativo y el intérprete de mandatos o Shell.

3.1.- El núcleo o kernel

En el nivel más bajo se encuentra el **núcleo** o **kernel**, que es la parte del sistema operativo encargada de *interactuar con el hardware*. Por tanto, servirá de intermediario entre el resto de los componentes del sistema operativo y el hardware. Ningún otro componente necesita conocer las particularidades del hardware del ordenador, simplemente le dice al kernel qué es lo que quiere y es el este quien se encargará de comunicarlo al hardware. Sus funciones se centran en la gestión de recursos como repartir el procesador entre los procesos, gestionar las interrupciones del sistema o realizar las funciones básicas de manipulación de memoria.

3.2.- Los servicios

El sistema operativo proporciona al usuario diversos **servicios**. Algunos de los más importantes son:

- **Gestión de procesos**: más adelante veremos en detalle los procesos, pero podemos avanzar que un proceso es un programa en ejecución, es decir, cada vez que ejecutamos un programa se crea el proceso correspondiente. La creación de un proceso no es trivial, ya que requiere asignarle tiempo de procesador, un espacio en memoria, acceso a las unidades de almacenamiento y dispositivos, ... El sistema operativo proporciona el servicio de creación, suspensión, reanudación y eliminación de procesos al usuario.
- **Gestión de memoria**: un recurso clave en el sistema es la memoria ya que todos los procesos en ejecución del sistema deben estar en memoria. El problema es que hay diversas tecnologías, pero ninguna óptima: las memorias más rápidas tienen precios muy elevados, mientras que las memorias más baratas que permiten grandes capacidades son excesivamente lentas. La solución a esta situación son las **jerarquías de memoria**, que combinan diferentes tipos de memoria buscando un equilibrio entre capacidad, velocidad y coste.
- **Gestión de ficheros y directorios**: dado que la memoria principal es volátil y su tamaño limitado, los ordenadores disponen de dispositivos de almacenamiento secundario como discos duros, tarjetas de memoria, ... Para almacenar la información en estos dispositivos el sistema operativo dispone de un sistema de ficheros que ayuda al usuario a organizar la información en archivos y carpetas.
- **Seguridad y protección**: este componente se encarga de garantizar la identidad de los usuarios y de definir lo que pueden hacer cada uno de ellos con los recursos del sistema.

3.3.- Intérprete de comandos o Shell

Es el componente que se encuentra al más alto nivel, en contacto directo con el usuario y por tanto, el que será responsable de interpretar las órdenes de este.

El intérprete de comandos puede ser textual, en el que el usuario introduce los comandos en modo texto siguiendo una sintaxis determinada, o gráfico, en el que el usuario interacciona a través de una interfaz gráfica mediante un ratón o dispositivo similar.

4.- ARQUITECTURA DE LOS SISTEMAS OPERATIVOS

Según su arquitectura los sistemas operativos pueden ser:

- **Sistemas monolíticos**: se caracteriza por una ausencia de estructura. El sistema operativo está formado por una serie de procedimientos que se pueden invocar entre sí sin ninguna restricción. No hay ningún tipo de ocultamiento de la información. Ejemplos: FreeBSD, MS-DOS, familia Windows 9x
- **Sistemas en capas**: el sistema operativo se estructura en una jerarquía de capas. El primer sistema operativo con esta arquitectura fue THE, desarrollado por E. W. Djistra y sus alumnos, el cual tenía las siguientes capas:
    - Capa 5: interfaz de usuario
    - Capa 4: aloja los programas de usuario
    - Capa 3: control de los dispositivos de E/S
    - Capa 2: administración de la comunicación inter-proceso y la consola de operador
    - Capa 1: administración de memoria y discos
    - Capa 0: asignación del procesador haciendo transparente la multiprogramación a las capas superiores.
- Otros ejemplos son Venus o MULTICS.
- **Máquinas virtuales**: fue implementada por primera vez en el VM/370 y consiste en separar los dos componentes principales de un sistema de tiempo compartido: multiprogramación y una máquina extendida con una interfaz más cómoda que el hardware desnudo. El corazón del sistema, llamado **monitor de máquina virtual**, se ejecuta directamente sobre el hardware y realiza la multiprogramación. Este proporciona varias máquinas virtuales a la capa superior que son copias exactas del hardware desnudo y que incluyen modo de kernel/usuario, E/S, interrupciones, … Cada máquina es idéntica al hardware verdadero, por lo que puede soportar cualquier sistema operativo soportado por la máquina real. Diferentes máquinas virtuales pueden ejecutar diferentes sistemas operativos.
- Otro ejemplo es el modo 8086 virtual disponible en los procesadores de 32 bits de Intel que permite ejecutar antiguos programas de MS-DOS.
- **Exokernels**: va un paso más allá de los sistemas operativos de máquina virtual proporcionando a cada usuario un clon de la computadora real, pero con un subconjunto de los recursos. En la capa más baja se encuentra un programa llamado exokernel que asigna recursos a las máquinas virtuales y que garantiza que ninguna máquina utilice los recursos de otra. Su ventaja es que ahorra una capa de mapeo o correspondencia. Algunos sistemas operativos que siguen esta filosofía son Nemesis, de la Universidad de Cambridge, y ExOS, del MIT.
- **Micronúcleo o microkernel**: una tendencia reciente es quitar lo posible del modo kernel, dejando un microkernel mínimo que provee un conjunto de primitivas o llamadas mínimas para implementar los servicios básicos. Sus ventajas son una reducción de la complejidad y la descentralización de los fallos. Algunos ejemplos son Amoeba, Minix, Hurd o Symbian.

- **Sistemas por módulos**: es el enfoque de la mayoría de los sistemas modernos. El núcleo está formado por módulos independientes entre sí. Los módulos se cargan en el núcleo cuando se necesiten, ya sea en tiempo de ejecución o durante el arranque. Ejemplos: Linux (con el comando lsmod se pueden ver los módulos cargados) o Solaris.

5.- CLASIFICACIÓN DE LOS SISTEMAS OPERATIVOS

Hay diversos métodos de clasificación de sistemas operativos en función del aspecto en que nos fijemos. Hay que tener en cuenta también que muchas clasificaciones se han ido diluyendo a lo largo del tiempo habiendo una separación muy difusa entre las diferentes categorías.

5.1.- En función del número de usuarios

- **Monousuario**: solo soportan el trabajo de un usuario que dispondrá de todos los recursos del sistema en exclusiva, evitándose los problemas típicos de contienda. Ejemplos: MS/DOS, Windows XP
- **Multiusuario**: son capaces de dar servicio a varios usuarios de forma simultánea de manera que compartan los recursos del sistema: CPU, memoria, … Ejemplos: UNIX, Linux, Windows Server

5.2.- En función del número de procesos simultáneos

- **Monotarea o monoprogramación**: solo pueden ejecutar una tarea cada vez y cuando terminan pasan a la siguiente.
- **Multitarea o multiprogramación**: se pueden ejecutar varias tareas de forma simultánea. Esta concurrencia es únicamente aparente ya que las tareas se van turnando en el uso del procesador.

5.3.- En función del número de procesadores simultáneos

- **Monoproceso**: trabajan con un único procesador sobre el que se van alternando todos los trabajos.
- **Multiproceso**: trabajan con varios procesadores y, por tanto, pueden desarrollar varias tareas simultáneamente.

5.4.- En función de los requerimientos temporales

- **Procesos por lotes (batch)**: cada trabajo ocupa completamente la CPU y cuando finaliza se pasa al siguiente. Esto permite la ejecución de varios programas sin supervisión directa del usuario.
- **Procesos con respuesta en tiempo**: de alguna manera es relevante el tiempo en que se deben ejecutar las aplicaciones. Según el tipo de restricción temporal pueden ser:
    - **Tiempo real**: su parámetro clave es el tiempo dado que muchas operaciones deben cumplirse en plazos estrictos. A su vez pueden ser de **tiempo real riguroso**, si el plazo es ineludible, o de **tiempo real no riguroso**, si es aceptable no cumplir de vez en cuando con un plazo. Ejemplos: VXWorks o QNX
    - **Sistemas interactivos**: también deben responder en tiempo a las peticiones del usuario, pero el condicionante temporal no es tan vital. Utilizan técnicas de multiprogramación para conseguir atender varias peticiones de forma simultánea. Ejemplos: Linux, Windows, MacOS, …

5.5.- En función del modo de ofrecer los servicios

- **Sistemas centralizados**: un ordenador se encarga de todo el trabajo de procesamiento y a él se conectan terminales desde los que el usuario envía sus trabajos. Ejemplos: MULTICS, OS/390

- **Sistemas en red**: mantienen varios equipos conectados entre sí para compartir recursos e información, pero manteniendo cada una autonomía en cuando a capacidad de proceso. Ejemplos: Novell Netware, Windows Server, Banyan VINES, ...
- **Sistemas distribuidos**: conjunto de equipos interconectados que se reparten los diferentes trabajos de forma transparente para el usuario. Ejemplos: Amoeba, Sprite, ...