CAPÍTULO I: INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS

ÍNDICE DE CONTENIDO

1. ¿Qué es un sistema operativo?

El *software* hace que un ordenador "haga algo".

Un ordenador es un sistema complejo ⇒ Un programador no puede tener en cuenta todos los aspectos de una computadora cada vez que hace un programa ⇒es necesario **ocultar la complejidad del *hardware* al programador.**

Se opta por colocar un nivel *software* por encima de la capa *hardware* con el fin de controlar todas las partes del sistema y presentar al usuario un interfaz o máquina virtual:

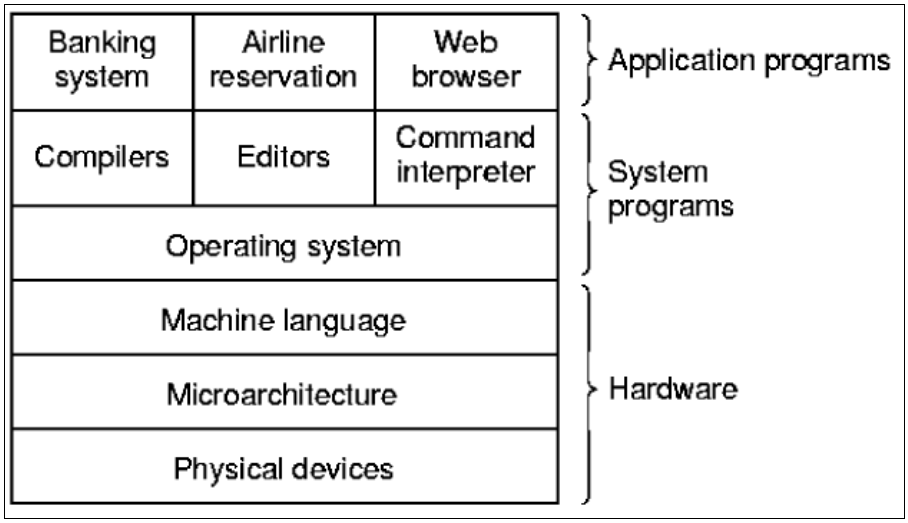

Figura 1: Estructura de capas hardware, software de sistema y programas de usuario.

Esta capa esta formada por los programas de sistema.

El sistema operativo (SO) es el **programa de sistemas** en el que se basan otros programas de este tipo.

Varias **definiciones** de SO:

- Es el *software* que controla al *hardware*.
- Son los programas que hacen utilizable el *hardware*.
- Se trata del *software* que hace funcionar un computador y proporciona un entorno para la ejecución de los programas.
- El *hardware* proporciona la capacidad de cómputo. Los SO ponen dicha capacidad al alcance de los usuarios y administradores de recursos. El principal recurso que administran es el *hardware* del computador: procesadores, memoria, archivos y dispositivos de E/S.
- Un ordenador consta de multitud de componentes que hay que controlar. Para que un programador pueda desarrollar su tarea debe poder aislarse de toda esa complejidad. Para esto, añadimos una capa de *software* sobre el *hardware* puro, que se va a encargar de gestionar todos los elementos del sistema y que presenta al usuario un interfaz o máquina virtual (también denominada extendida) más fácil de entender y programar.

1.1. Funciones

Se agrupan en tres categorías:

1. Administración de los recursos.
2. Ejecución de servicios para los programas.
3. Ejecución de los mandatos de los usuarios.

Estas tres funciones han dado lugar a la siguiente visión de capas del SO:

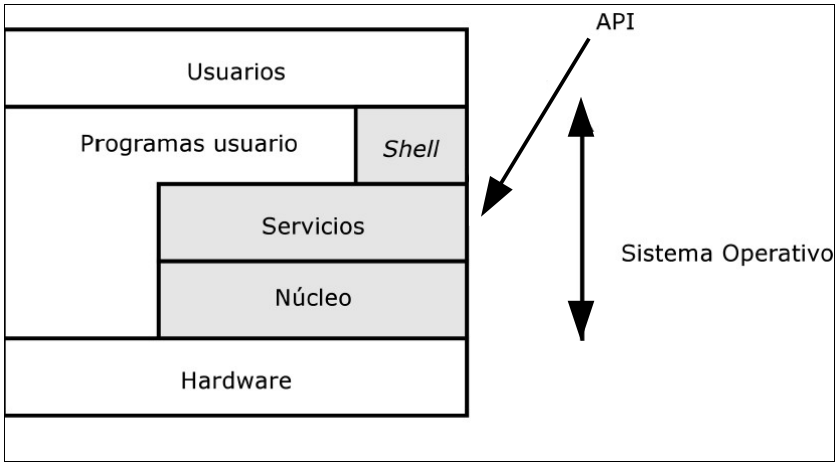

Figura 2: Capas del sistema operativo.

1.1.1. Administración de recursos

Tanto *hardware* (memoria, procesador) como *software* (archivos o puertos de comunicaciones) de la computadora. Los distintos procesos compiten por los recursos.

El SO debe:

1. Asignar y planificar los recursos entre procesos ⇒ debe tener estructuras para controlar el estado de los recursos.
2. Proteger los recursos ⇒ impedir que unos procesos accedan a los recursos de otro.
3. Contabilidad (estadísticas), permite medir las cantidades de recursos necesarios por un proceso. Nos ayuda a hacer un buen equilibrado del sistema.

Algunos sistemas operativos, como es el caso del MSDOS, permitían que los programas de usuario accedieran directamente a los recursos hardware del ordenador sin intervención del SO.

1.1.2. Ejecución de servicios para programas

El SO ofrece un conjunto de servicios (llamadas al sistema o *system calls*), proporcionando una visión del computador de máquina extendida.

Las llamadas al sistema sirven de interfaz entre el sistema operativo y los programas de usuario.

Estos servicios estén disponibles como funciones de librería, en Linux la biblioteca que contiene todas estas funciones es la libc.

El compilador es el encargado de ofrecer los mecanismos necesarios para el acceso al SO.

El programa escrito en un lenguaje de programación podrían funcionar en diferentes máquinas siempre que lo compilemos con el compilador correspondiente de esa máquina.

Estos servicios se agrupan en:

- **Ejecución de programas**, incluye servicios para:
    1. Lanzar, parar o abortar la ejecución.
    2. Conocer y modificar las condiciones de ejecución.
    3. Comunicar y sincronizar.
- **Órdenes de E/S,** proveen a los programas de operaciones de lectura, escritura y modificación del estado de periféricos.
- **Operaciones sobre archivos**, similar a las órdenes de E/S pero ofrecen un nivel de abstracción mayor.
- **Detección y tratamiento de errores**, tratan todas las condiciones de error del *hardware* (desbordamientos, violaciones de memoria, etc...).

1.1.3. Ejecución de los mandatos de los usuarios (interfaz de usuario)

El usuario realiza las operaciones que requiere la máquina mediante órdenes del intérprete de comandos (IC) o *shell*.

El IC es una capa de traducción entre el usuario y el computador.

El IC tiene una serie de comandos que el usuario debe conocer para poder llevar a cabo sus acciones.

1.2. Objetivos

- En grandes computadoras, incrementar la productividad del *hardware*.
- En pequeñas computadoras, incrementar la productividad del usuario.

2. Historia de los SO

La evolución de los sistemas operativos est?relacionada con las generaciones de computadoras.

La “máquina analítica”, primera computadora digital diseñada por Charles Babbage, no tenía un sistema operativo.

Generaciones:

1. La primera generación (1945-1955): Tubos de vacío y conexiones.
2. La segunda generación (1955-1965): Transistores y sistemas de procesamiento por lotes.
3. La tercera generación (1965-1980): Circuitos integrados y multiprogramación.
4. La cuarta generación (1980-...): Computadoras personales.

2.1. La primera generación (1945-1955): Tubos de vacío y conexiones

Después de un largo periodo de inactividad, desde los intentos de Babbage, aparecen las máquinas de cálculo mediante tubos de vacío.

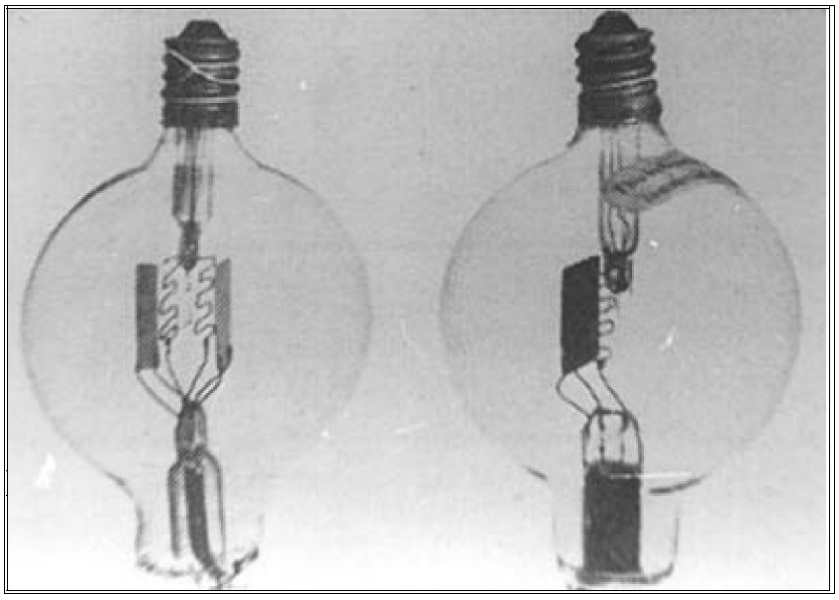

Figura 3: Tubos de vacío.

Eran máquinas muy grandes y muy poco potentes comparadas con los PC actuales.

Un solo grupo de personas diseñaba, construía, programaba, operaba y daba mantenimiento a cada máquina.

Se programaba en lenguaje máquina:

Figura 4: Lenguaje máquina.

No existían los SO.

El programador reservaba la máquina, introducía el programa (conexiones en los paneles) y esperaba los resultados. Los tubos de vacío eran muy poco fiables.

En los 50 la técnica mejoró un poco con la aparición de las tarjetas perforadas.

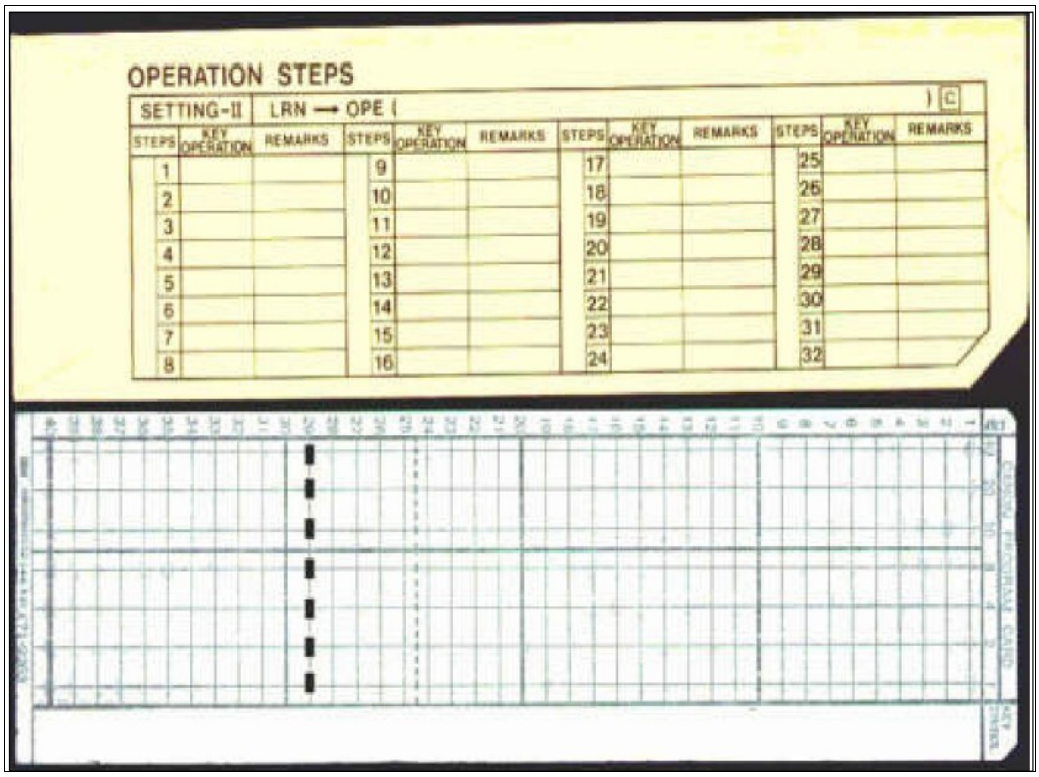

Figura 5: Tarjetas perforadas.

2.2. La segunda generación (1955-1965): Transistores y sistemas de procesamiento por lotes

Aparece el transistor ⇒ordenadores más fiables ⇒se venden a clientes.

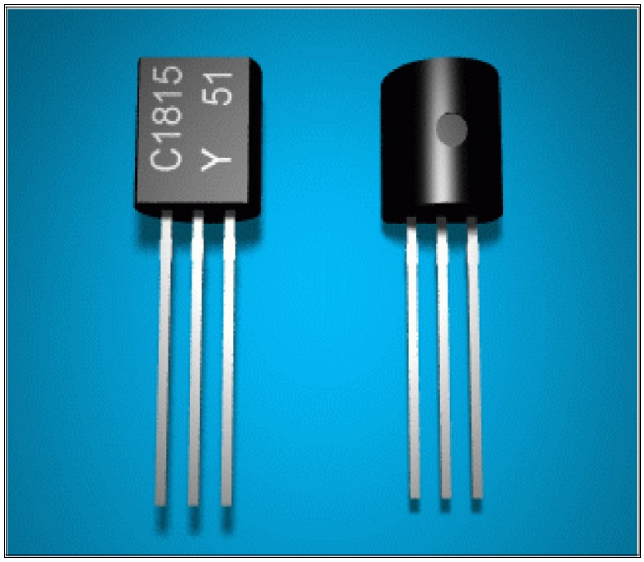

Figura 6: Transistores.

Clara separación entre diseñadores, constructores, operadores, programadores y personal de mantenimiento.

Los ordenadores se confinan en los cuartos de cómputo, sólo algunas entidades se los podían permitir.

Figura 7: Cuartos de cómputo.

¿Cómo se ejecuta un trabajo (programa o conjunto de ellos)?

- Se escribía el programa (Fortran, ensamblador) en papel y después se pasaba a tarjetas perforadas.
- Se reserva el ordenador en múltiplos de 30 minutos.
- El conjunto de tarjetas se llevaba al cuarto de lectura. También podía ser necesario cargar algún compilador.
- Si se producía algún error se indicaba mediante indicadores luminosos. Para buscar la causa del error se examinaban los registros y la memoria.
- Si el programa finaliza correctamente se iba al cuarto de salida a recoger los resultados en papel.

Con esta forma de trabajo se producía mucha pérdida de tiempo. La solución fue el **sistema de procesamiento por lotes (*Batch*)**. Se recolectan varios trabajos antes de proceder a ejecutarlos:

- Los programadores llevan sus trabajos a una computadora de propósito específico pequeña y relativamente barata (IBM 1401) que los almacena en una unidad de cinta.
- Después de transcurrir un tiempo para recolectar trabajos, el operador lleva la cinta de trabajos a una computadora m? cara (IBM 7094) que los ejecuta.
- La cinta con los resultados van al 1401 y se imprimen.

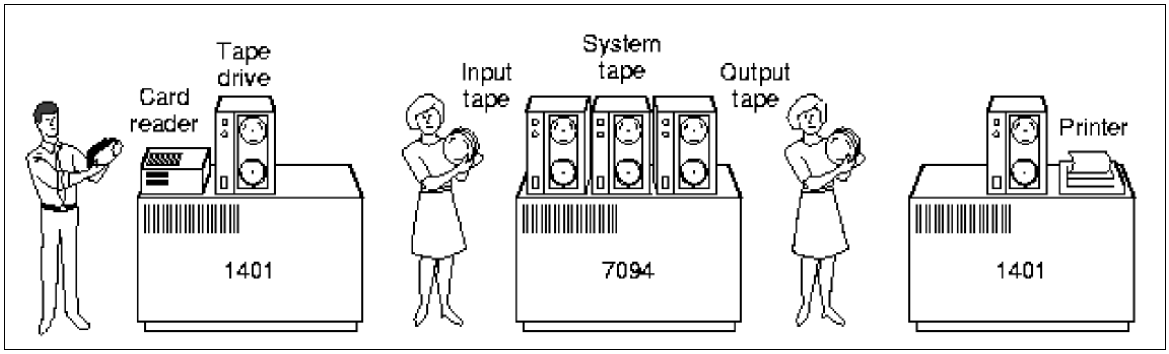

Figura 8: Esquema de tratamiento por lotes.

Para ir leyendo los trabajos, enviarlos a ejecución y almacenar los resultados, es necesario un programa que es el predecesor de los SO actuales.

A estos SO se les llamaba FMS (*Fortran Monitor System*) o monitor.

El monitor da el control al procesador y este lo devuelve al finalizar la ejecuci? del programa.

Para indicarle al SO lo que debe de hacer aparece el lenguaje JCL (*Job Control Language*).

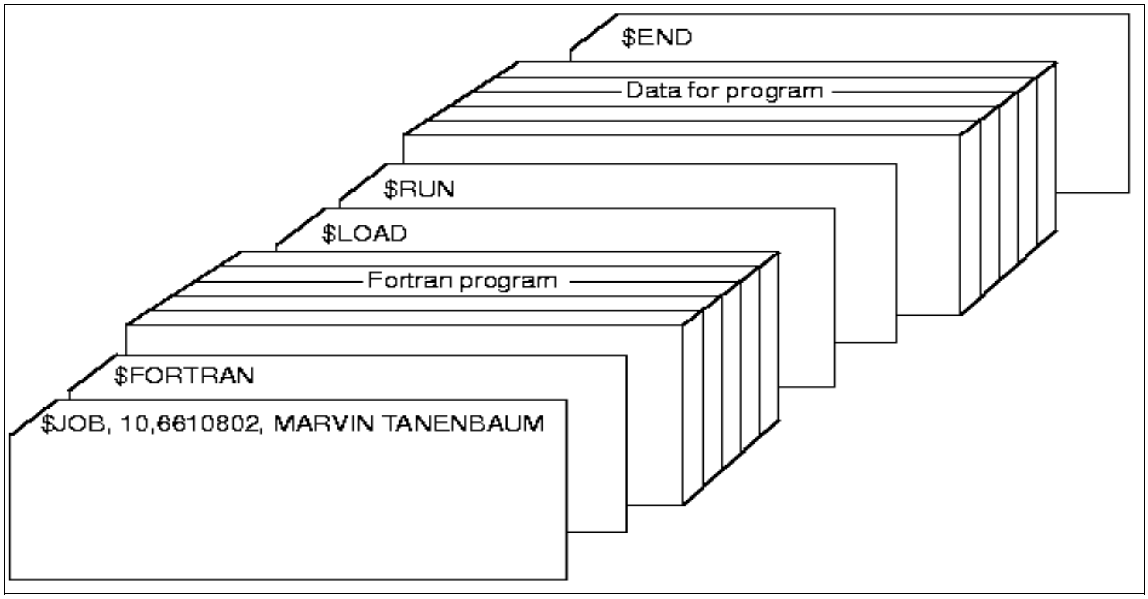

Figura 9: Representación esquemática de un trabajo por lotes.

El monitor no realizaba acciones de protección de memoria, temporización, instrucciones privilegiadas, interrupciones...

Dos nuevos recursos:

- Memoria para el monitor.
- Tiempo de máquina consumido por el recurso.

Inconvenientes:

- Aumenta el tiempo de retorno.
- Depuración indirecta.
- Cuello de botella en las operaciones E/S.

Soluciones al problema de E/S:

- *Buffering*: un buffer para leer el siguiente trabajo.
- *Spooling (Simultaneous Periphereal Operations On Line)*: un disco para almacenar temporalmente las salidas.

2.3. La tercera generación (1965-1980): Circuitos integrados y multiprogramación

Aparece el concepto de **mainframe** o **macrocomputadora** con la introducción de la serie 360 de IBM, computadores compatibles entre sí.

Características: 5 pies de altura x 6 de ancho, memoria de 64Kb y un precio de 200000 dólares.

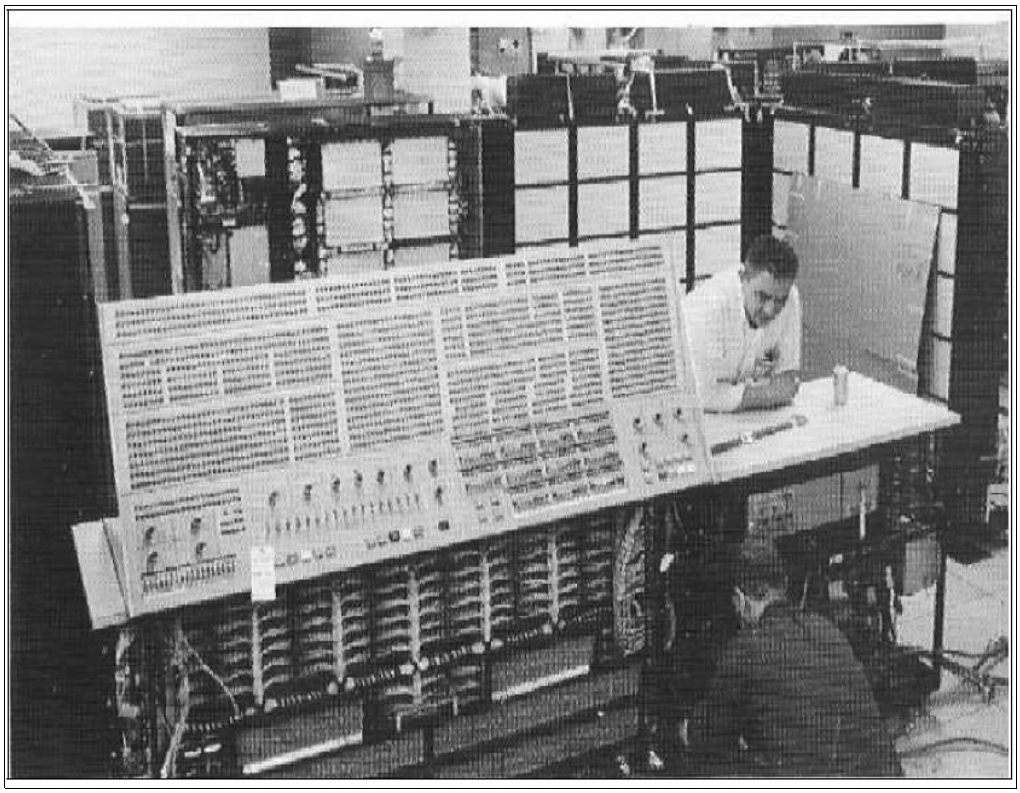

Figura 10: Mainframe IBM 360.

Fue la primera que usó los circuitos integrados haciendo las computadoras más pequeñas y menos costosas.

Todo el *software* debía de funcionar en todas las máquinas ⇒un SO complejo y con muchos fallos.

Para evitar las pérdidas de tiempo por las operaciones de E/S aparece el concepto de **multiprogramación** o **multitarea**:

- Se cargan varios trabajos en memoria.
- Cuando uno está esperando a que finalice una operación de E/S, se sigue con otro trabajo.

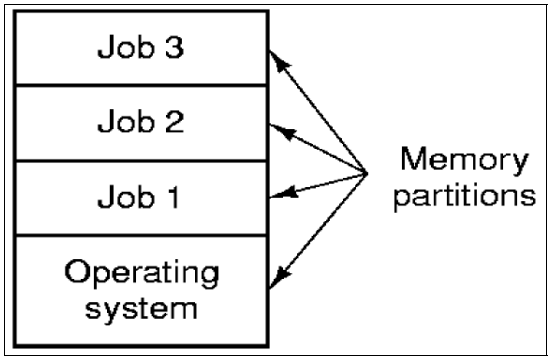

Figura 11: Esquema de memoria de un sistema multiprogramado.

Para solucionar los problema de los SO de procesamiento por lotes, aparece el concepto de tiempo compartido (***timesharing***) que es una variante de la multiprogramación que:

- Soporta múltiples usuarios interactivos.
- Los usuarios disponen de todos los recursos de la máquina como si estuviera solo.

Comienzan a fabricar los intérpretes de comandos para satisfacer la necesidad de interacción de los usuarios.

Aparecen los minicomputadores:

Figura 12: Minicomputador Comodore C64.

2.4. La cuarta generación (1980-...): Computadoras personales

La evolución de las tecnología de circuitos, LSI y VLSI (Very Large Scale Integration) hizo posible crear computadoras personales, máquinas muy potentes y baratas que los macrocomputadores de los 70.

Se empieza a desarrolla una floreciente industria de *software*.

Comienza a proliferar la transferencia de información entre computadores. Aparecen los sistemas operativos de red (es necesario saber donde están los recursos) y distribuidos (no es necesario saber donde están los recursos).

A Continuación se muestra un cuadro cronológico con los hitos más importantes dentro de la historia reciente de los últimos Sistemas Operativos más importantes.

Año	Linux	Microsoft	Apple
1979			Comienzo del proyecto Macintosh para crear una computadora barata y fácil de usar para el consumidor medio.
1981		Lanzamiento de PC-DOS, el sistema operativo que equipaba a los primeros IBM-PC.	Vista de los ingenieros de Apple al centro Xerox Alto donde se desarrollaba una nueva interfaz gráfica de usuario (GUI) con barras de menú, menús emergentes y tecnología click-and-drag.
1983	Richard Stallman anuncia la intención de crear un sistema operativo completamente libre dentro del proyecto GNU.	MS-DOS 2.0, con soporte para subdirectorios, operaciones con ficheros, redirección de comandos y pipes.	Aparición de Lisa OS, el S.O. de los Apple Lisa.
1984		MS-DOS 3.0, soporte para PC-AT, disquetes de 1.2 MB y particiones de 32 MB.	Se lanza la primera version del Mac OS (System 0.0), primer S.O. con GUI.
1985	Se crea la FSF (Free Software Foundation), una fundación no lucrativa para promover el desarrollo de software libre dentro del proyecto GNU.	Lanzamiento de Windows 1.0, una interfaz gráfica de usuario que funciona bajo MS-DOS y que imita la interfaz del Mac OS.	Mac Os System 2.1 introduce el sistema de ficheros HFS.
1987	El profesor Andrew Tannenbaum publica su libro texto “Sistemas Operativos: Diseño e implementación” que incluye una versión de aprendizaje de UNIX denominada Minix.	Aparece Windows 2.0, con un límite de 1MB de memoria alcanzó gran popularidad gracias a los paquetes Excel y Word.	Mac Os System 5 incorpora MultiFinder que posibilita la ejecución de varios programas a la vez mediante multitarea cooperativa.
1988		MS-DOS 4.0, se añade el DOS Shell y soporte para discos de 32 MB. Esta versión contiene muchos errores que deben corregirse con una nueva versión 4.01	
1989	Se lanza la primera versión de la GNU General Public License (GPL) para dar cobertura legal a los proyectos de software libre.		
1990		Windows 3.0 proporciona multitarea en modo gráfico bajo MS-DOS.	
1991	Linus Torvald publica la versión 0.01 de un sistema operativo similar a Minix denominado Linux. Richard Stallman muestra	MS-DOS 5.0, mejora en el soporte de memoria.	Mac Os System 7. Incorpora cambios sustanciales en la interfaz de usuario. Introduce el concepto de “acceso directo”.

Año	Linux	Microsoft	Apple
	su interés en que el kernel de Linux se distribuya bajo GPL.		Acceso a memoria en 32 bits. Soporte para memoria virtual.
1992	Primera distribución de Linux (MCC Interim Linux). Soporte para X-Windows en Linux. En noviembre se funda SuSE.		
1993	Se crea la distribución Debian. Se lanza la Slackware 1.0.	Soporte de red con la versión 3.11 de Windows. Lanzamiento de Windows NT 3.1, la apuesta de Microsoft para servidores.	
1994	Versión 1.0 del kernel. Se lanzan las distribuciones 1.0 de Red Hat, SuSE y Caldera. Primera edición de la Linux Journal.	Última versión de MS-DOS (6.22) que se distribuye en solitario. A partir de esta fecha se incluye embebida en los productos Windows. Desaparece como tal con Windows Me (año 2000).	
1995	Se comienza el proyecto para crear el servidor web Apache.	Windows NT 3.51, se diferencian dos versiones Workstation y Server. Soporte para los procesadores Pentium y OpenGL. Windows 95, aunque basado en el MS-DOS de 16 bits, aporta una nueva API Win32 para aplicaciones de 32 bits. Nueva interfaz gráfica. Incorpora la tecnología Plug&Play. Numerosos bloqueos del sistema por la utilización de código en 16 bits.	
1996	Versión 2.0 del kernel, se añade soporte SMP (varios procesadores). Se anuncia el inicio del proyecto KDE (Kool Desktop Environment).	Windows 95 OSR2, se incorpora el nuevo sistema de ficheros FAT32. Windows NT 4.0, nueva interfaz de usuario estilo Windows 95. Nuevas versiones Terminal Server y Advanced Server para grandes infraestructuras.	
1997	Comienza el proyecto GNOME, otro entorno de escritorio para Linux.		Mac OS 8, introduce la ejecución multi-hilo y el sistema de ficheros HFS+. En sucesivas versiones 8.X se introduce el soporte para USB y Firewire.
1998	Aparece el servidor Google utilizando servidores Linux.	Windows 98, soporte para nuevas tecnologías: DVD, firewire, USB y AGP. La principal diferencia con W95 es la modificación del núcleo para permitir el uso de controladores de Windows NT.	
1999	Versión 2.2 del kernel, soporte para la carga dinámica de módulos IDE, SCSI, USB, ...		Mac OS 9, se cambia la GUI y la API para el manejo de ficheros.
2000	IBM anuncia fuertes inversiones en el desarrollo de Linux.	Windows 2000, evolución importante dentro de los sitemas Microsoft ya que incorporaba bastantes mejoras técnicas: EFS, DSF, soporte RAID y Directorio Activo. Se lanza Windows Me, una copia de W98 con más aplicaciones añadidas. Famoso por sus continuos errores.	Mac OS X 10 supone un cambio revolucionario en Apple ya que basa su nuevo núcleo en una variante de un UNIX BSD denominado Darwin.

Año	Linux	Microsoft	Apple
2001	Versión 2.4 del kernel, soporte para ISA pnp, Bluetooth, LVM y RAID.	La unión entre las familias de S.O. de servidores (NT) y de usuario (Win 9.X) se logró Windows XP. Presenta una gestión de memoria y de la multitarea mejorada. Mejoras notables en el tratamiento de la multimedia (versión Media Center).	Mac OS X 10.1 "Puma" es una nueva versión que soluciones algunos errores de la versión 10.0.
2002			Mac OS X 10.2 "Jaguar" última versión en incorporar Internet Explorer como navegador por defecto.
2003	Versión 2.6 del kernel. Mejoras en el sistema de ficheros, capaz de almacenar 16 Terabytes. Aparece Molinux,distribución linux de la Junta de Extremadura, precursora de Guadalinex.	Windows Server 2003. Basado en el núcleo de XP es la versión para servidores de éste con la incorporación de nuevos servicios y mejora en el rendimiento (por la supresión de algunas características de XP).	
2004	Aparece la primera versión de Ubuntu Linux.		Mac OS X 10.3 "Panther" última versión en incorporar Microsoft Internet Explorer.
2005		Lanzamiento Windows Vista Beta 1 futuro reemplazo de Windows XP. Entre otras mejoras, destaca su novedosa interfaz de usuario (sorprendentemente muy parecida a prototipos existentes de Sun y Linux).	Mac OS X 10.4 "Tiger" introduce el soporte de Apple para arquitecturas Intel x86.
2006	Oracle anuncia su intención de poseer su propia distribución de Linux. Se lanza el primer prototipo de un portátil a 100$ basado en Linux. Bill Gates ridiculiza el proyecto.		
2007		Previsión de lanzamiento de Windows Vista. Previsión de lanzamiento de Windows "Longhorn" Server (con el nombre probable de Windows Server 2007)	
2011		Previsión de lanzamiento de Windows "Vienna" sucesor de Windows Vista.	

2.4.1. Nuevas tendencias en Sistemas Operativos

La reciente aparición de diferentes dispositivos con una potencia de cálculo cada vez mayores (terminales de telefonía móviles, agendas electrónicas, vídeo consolas,...) han hecho que las tradicionales aplicaciones informáticas que las soportaban no se construyan ad-hoc sino que tomen como base sistemas operativos específicos para dichos dispositivos.

En cuanto a estos tipos de sistemas operativos cabe destacar los siguientes:

- **Palm OS**, es el sistema operativo diseñado para los Palm Pilot. Este tipo de dispositivos son denominados ayudantes personales digitales, agendas electrónicas o PDA (Personal Digital Assistant).

- **Windows Mobile**, es un sistema operativo compacto para dispositivos móbiles basados en el API Win32, o lo que es lo mismo, esta basada en las llamadas al sistema que implementan los sistemas operativos de Microsoft para aplicaciones de 32 bits.

- **Linux**, debido a su escalabilidad y su facilidad de adaptación también existen distribuciones específicamente adaptadas para su utilización en PDAs, móviles y tablet PC (distribución maebo de Nokia) e incluso podemos instalar Linux en una X-Box (www.linux-xbox.org).

3. Componentes del sistema operativo

Los componentes que forman parte de un SO son:

1. Administrador de procesos.
2. Gestión de memoria.
3. Administración de E/S.
4. Sistema de ficheros.
5. Comunicación y sincronización.

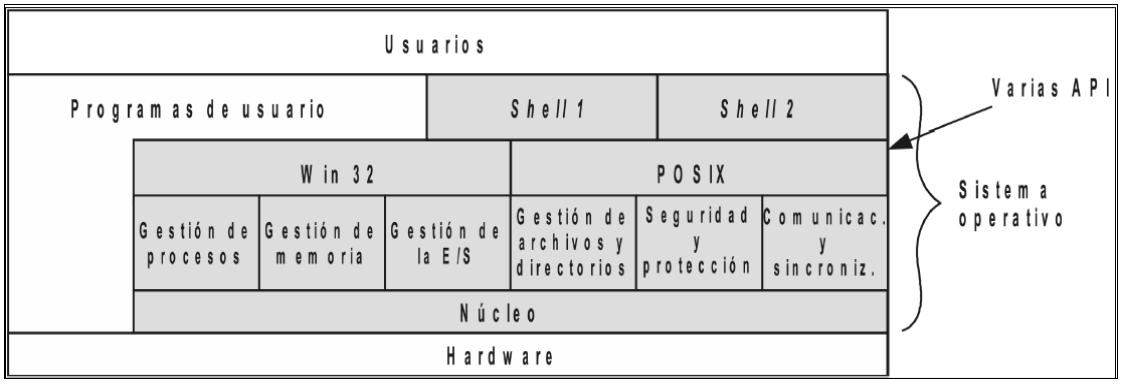

Figura 13: Componentes de un sistema operativo.

3.1. Administrador de procesos

Un proceso es un concepto dinámico a diferencia del programa que será un elemento estático.

Un proceso es básicamente un programa en ejecución.

Los servicios que de el SO depende de si el sistema es monotarea, multiproceso, monousuario, multiusuario.

En un sistema tendremos en general más de un proceso. Para el manejo simultáneo de dichos procesos necesitaremos una estructura de datos denominada tabla de procesos.

El proceso durante su vida (desde su creación hasta su destrucción) puede pasar por varios estados: preparado, ejecución, bloqueado.

Para estos estados tendremos una cola de procesos preparados para ejecutarse, varias de procesos bloqueados, pero un solo proceso en ejecución.

Habrá múltiples eventos que pueden producir un cambio de estado:

- Despacho de un proceso.
- Expiración del tiempo de ejecución.
- Petición de operación de E/S.

Toda la información referente a un proceso se guarda en el Bloque de Control de Proceso (PCB). Tendremos uno por cada proceso, y será una estructura de datos que contendrá:

- Identificador de proceso.
- Prioridad.
- Estado en el que se encuentra.
- Contexto (registros, *flags*,....).

Además podremos tener una serie de operaciones asociadas a un proceso:

- Crear.
- Destruir.
- Despertar.
- Despachar, ...

En todo esto hay una operación fundamental que debe ser llevada a cabo por el SO como es la conmutación de procesos (sacar al proceso que tiene la CPU y poner en ejecución al primero de la cola de preparados).

De la efectividad de esta operación (muy frecuentemente usada) depende mucho el rendimiento del SO en multiprogramación. Toda esta información, junto con las operaciones de los procesos y los cambios de estado variar? dependiendo del SO en el que estemos trabajando.

Una de las principales tareas del gestor de procesos es la planificación de procesos (decidir el orden de ejecución).

Veremos dos tipos de planificadores:

- A corto plazo: Decide qué proceso de los preparados se ejecuta.
- A largo plazo: Decide cuál es el siguiente proceso por lotes a ejecutar.

Tendremos diferentes mecanismos de planificación: *FIFO*, *SRTN* (Menor Tiempo Restante), *Round Robin*, etc. Estos métodos pueden ser apropiativos o no apropiativos.

La gestión de múltiples procesos plantea una serie de problemas, como son: concurrencia, sincronización (ambos estudiados en otra asignatura) e interbloqueos.

En un sistema multiusuario hay que mantener un registro que indique a qué usuario pertenece cada proceso.

El SO debe proporcionar los mecanismos necesarios para permitir la comunicaci? entre procesos.

3.2. Gestión de memoria

Para que un programa se ejecute debe estar cargado en memoria (imagen de memoria).

Las funciones del SO son asignación, liberación y compartición de memoria.

En el código fuente de cualquier programa las direcciones serán simbólicas (representadas por variables), tras la compilación obtendremos direcciones relativas y cuando se cargue en memoria tendremos las verdaderas direcciones absolutas (no conocidas durante la compilación).

Los SO disponen de diferentes formas de traducir direcciones, desde modelos muy básicos a otros muy complejos.

Los modelos más sencillos reunirán las siguientes características:

- Monoprogramados (uno solo en memoria).
- Residentes (no abandonan la memoria en toda la ejecución).
- Inmóviles (no cambian de posición en memoria a lo largo de la ejecución).
- Contiguos (direcciones de memoria contiguas para cada programa).
- Enteros (todo el programa debe estar en memoria).

Estos primeros modelos, muy sencillos, y poco eficientes consiguen una mejor administración de la memoria según añadimos nuevas técnicas:

- Reubicación estática: retrasar la traducción de direcciones hasta el momento de la carga, en vez de hacerlo en el momento de la compilación.
- Multiprogramación: dividir la memoria en particiones, ya sean fijas o variables. Surgen con esta técnica dos conceptos clave para valorar la eficacia: fragmentación interna y fragmentación externa.
- Reubicación dinámica: retrasar la traducción de direcciones hasta el momento de la ejecución, con lo cual conseguimos movilidad, al no cargarse el programa siempre en las mismas posiciones.
- Intercambio (Swapping): usar un dispositivo de almacenamiento secundario para sacar de memoria los programas que no están en ejecución y volver a traerlos cuando se necesiten ejecutar de nuevo.

Además de estas nuevas técnicas sobre los primeros modelos, existen otros modelos más complejos de administración de memoria, cuya idea principal sería no obligar a los programas a situarse en posiciones contiguas de memoria. Estos modelos serán:

- la paginación.
- la segmentación.

- Modelos combinación de paginación y segmentación.

La idea fundamental para estos modelos consiste en no tener todo el programa cargado en memoria, sino llevar sólo aquella parte del programa que se requiere para su ejecución, fundamento este de la Memoria Virtual.

Se deben analizar nuevos problemas asociados con estos modelos: el reemplazo de páginas, la asignación de un número de páginas a cada proceso, etc.

3.3. Sistema de ficheros

Es la parte del SO encargada de la gestión de los datos que se encuentran en los dispositivos de almacenamiento.

Proporciona una visión lógica compuesta por una serie de objetos (archivos y directorios) identificables con un nombre lógico.

El SO debería tener en cuenta los siguientes aspectos para una correcta administración:

- Gestión del disco.
- Gestión del espacio libre del disco.
- Gestión del espacio ocupado.
- Estructura de los directorios.
- Compartición de ficheros.
- Protección de ficheros.

La petición de datos se realiza de forma más rápida de lo que puede ser atendida, produciéndose colas de espera.

Para atender esta cola se necesita una buena política de gestión del disco.

Para decidir qué políticas son más adecuadas nos fijaremos en una serie de criterios: justicia, productividad, mínimo tiempo de respuesta, etc..

Los algoritmos se centran en reducir el tiempo de búsqueda (tiempo en posicionar el cabezal en la pista correspondiente), ya que estos tiempos son mucho m? grandes que los tiempos de latencia (tiempo en posicionarnos en el sector correspondiente).

Los principales algoritmos de planificación de disco son: SSTF, FCFS, SCAN, C-SCAN, etc...

La gestión del espacio libre no es complicada, pero si un poco m? compleja que la del espacio ocupado. Entre las que se estudiaran destacan la gestión de Ms-Dos y la de Unix.

Se verán distintas formas de compartición de ficheros: *Hard Link* y *Soft Link*.

3.4. Administración de E/S

Proporciona un interfaz entre los dispositivos de E/S, y el resto del sistema.

Las características de los periféricos son muy dispares, y pueden complicar el objetivo de independizar al usuario de las particularidades del dispositivo: velocidad de transferencia, unidad de transferencia, representación de datos, operaciones permitidas, ...

A la hora del diseño de la E/S en un SO deberemos tener en cuenta: la eficiencia, la seguridad y protección, la independencia de los dispositivos, ...

Para conseguir esta independencia se trabaja con operaciones uniformes y dispositivos virtuales, lo que nos proporciona además la posibilidad de redireccionamiento y la comodidad de la portabilidad.

Se estudiarán las interrupciones y el camino que sigue una operación de E/S desde que es lanzada por un proceso hasta que retornan los resultados de dicha operación.

4. Estructura de los SO

Los distintos enfoques que se han considerado para hacer SO han dado lugar a diferentes arquitecturas o estructuras que se pueden englobar en:

1. Estructura monolítica.
2. Estructura jerárquica.
3. Estructura cliente-servidor.
4. Estructura orientada a objetos.

4.1. Estructura monolítica

Es la estructura de los primeros sistemas operativos constituidos por un sólo programa y que se caracterizan por no tener estructura.

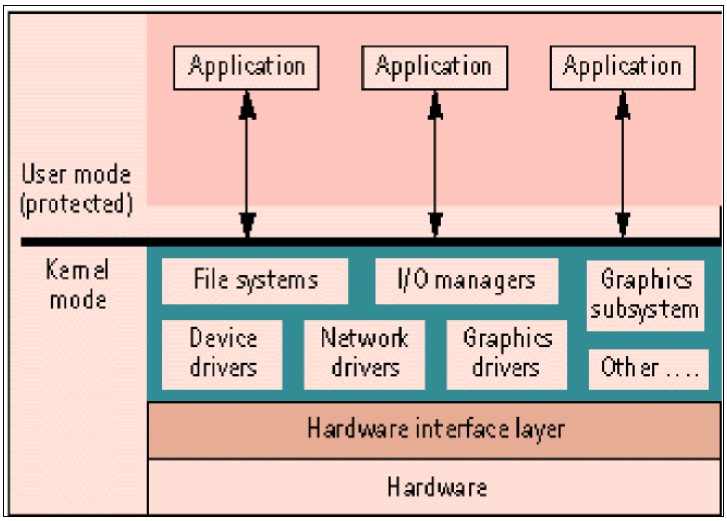

Figura 14: Esquema de un sistema operativo con estructura monolítica.

Esta formada por muchas rutinas enlazadas de tal forma que cualquier rutina puede llamar a otra.

Este conjunto de rutinas se compilaban y daban lugar a un programa que se ejecutaba en el ordenador.

Cada rutina debe tener muy buena definición en el interfaz con otras rutinas (paso de parámetros y devolución de los resultados).

Su falta de estructura hace que se pueda diseñar de forma que se ajuste escrupulosamente a los fines deseados.

Podemos decir que es un gran lío de rutinas entrelazadas donde cualquier rutina puede llamar a otra.

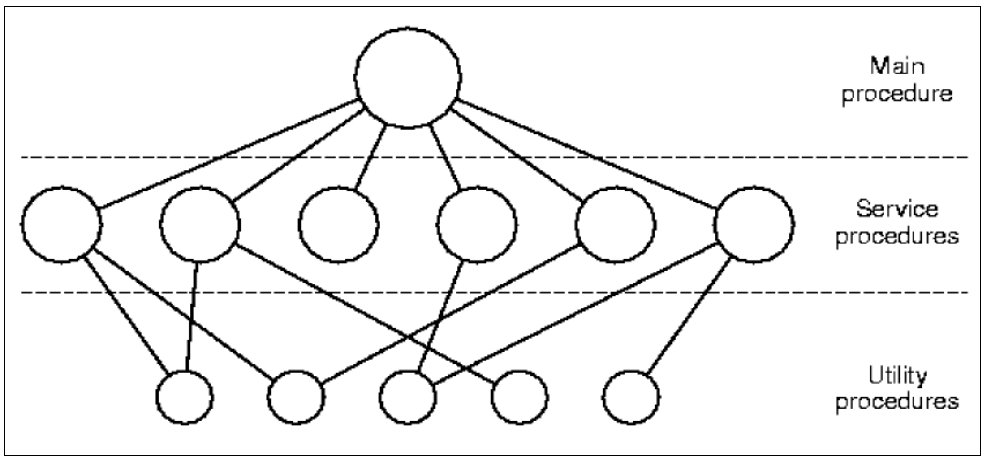

Figura 15: Llamadas a procedimientos en un sistema operativo con estructura monolítica.

Unix es un ejemplo de kernel monolítico. En las últimas versiones se ha mejorado su mantenimiento mediante el uso de módulos.

Los módulos son partes de código que se enlazan o descargan del núcleo bajo demanda.

Las ventajas que proporciona el uso de módulos son:

- Modularización.
- Independencia de la plataforma.
- Mejor uso de la memoria.
- No hay penalización una vez enlazado el módulo.

  Ejem: MS-DOS, Linux clásico

4.2. Estructura jerárquica

A medida que crecía la complejidad de los SO y las necesidades de los usuarios se necesita una mejor organización del software debido a la frecuencia de los cambios.

Se divide el SO en módulos, perfectamente definidos y con un interfaz claro con cada uno de los módulos.

Se consigue así un diseño modular, que permite el mantenimiento del SO durante todo su ciclo de vida.

Cada módulo del SO es una capa, y cada una de ellas lleva a cabo unas funciones muy concretas y específicas.

Esto provoca una jerarquía entre funciones (de ahí el nombre de esta estructura), donde cada capa puede invocar funciones de las capas inferiores, pero nunca de las superiores.

Cada capa tiene una función de entrada conocida como puerta (trap) por donde se pueden hacer llamadas a capas inferiores.

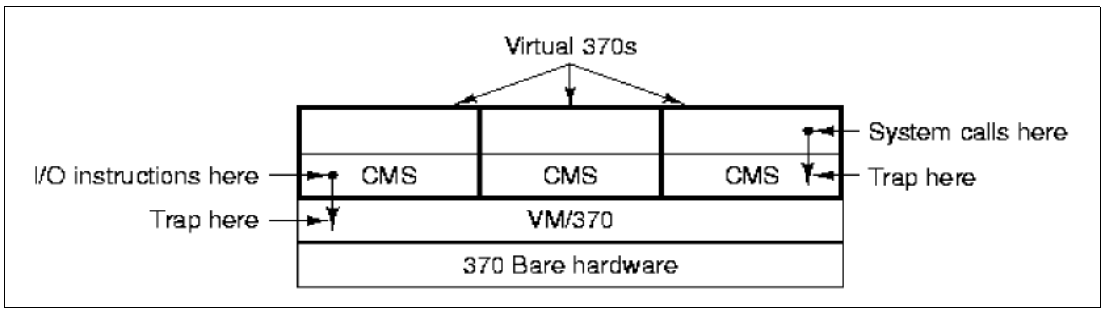

Figura 16: Estructura de un sistema operativo jerárquico.

Las capas más internas estarían más protegidas de accesos indeseados que las externas.

Las capas en las que se estructuran normalmente este tipo de SO son las siguientes:

1. Planificación del procesador: Gestión del procesador y como acceden a él los programas

2. Gestión de memoria: Gestión de memoria y su distribución.

3. Gestión de la E/S: Conjunto de rutinas que gestionan los dispositivos conectados al ordenador.

4. Subsistema de ficheros: Gestionan la información de los usuarios y garantizan su confidencialidad.

5. Programas de usuario.

   Ejem: OS/2, THE (Dijstra 68)

4.3. Estructura cliente-servidor

Es bastante reciente y se está haciendo cada vez más popular, debido en gran parte a que puede ser ejecutado en casi todo tipo de ordenadores y para todo tipo de aplicaciones.

También se le denomina microkernel.

La gran diferencia es la forma en que se va a distribuir el trabajo.

El principal objetivo de diseño es crear un núcleo lo más pequeño posible, creando sólo aquellas funciones que:

- son críticas en tiempo (respuesta rápida a eventos externos).
- son indispensables para la correcta administración del procesador.
- son de uso común y general de todas las aplicaciones.

El núcleo sólo ofrece los mecanismos para:

- Gestión de procesos.
- Gestión de memoria.
- Comunicación entre programas.

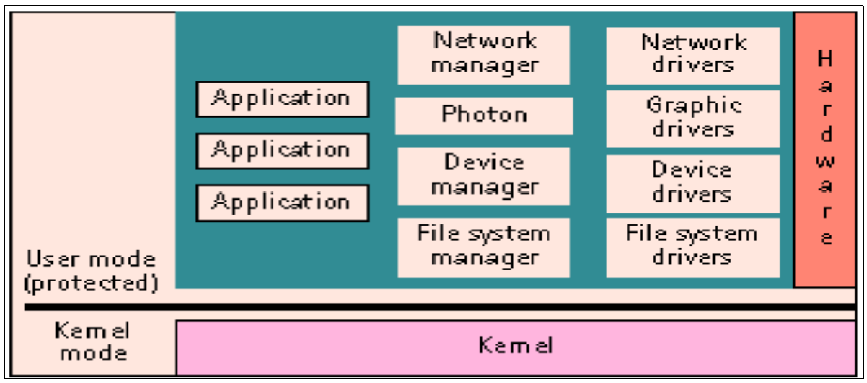

Figura 17: Esquema de un sistema operativo microkernel.

Los programas tendrán ahora que llevar a cabo algunas de las funciones del SO.

Cualquier proceso podría ser Cliente o Servidor.

La principal misión del núcleo es establecer la comunicación entre los clientes y los servidores mediante mensajes:

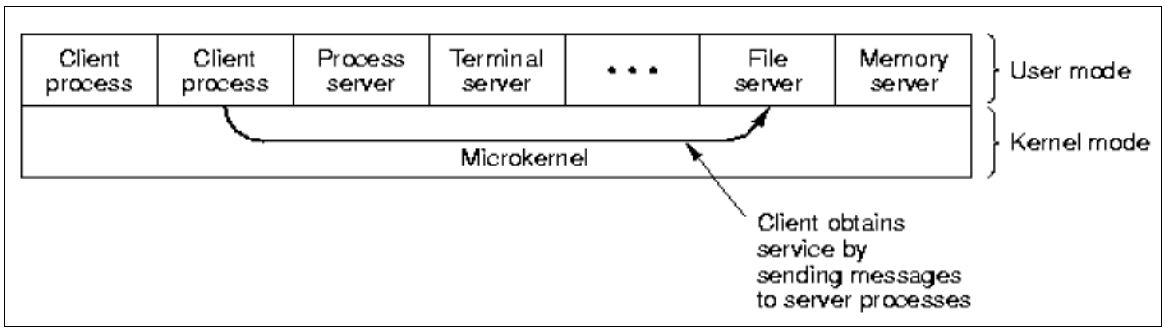

Figura 18: Esquema se provisión de servicios en sistema operativo cliente-servidor.

Estos SO se suelen dividir en partes especializadas para ciertos aspectos que proporcionan las siguientes ventajas:

- Mayor modularidad.

- Sistema más rápido.
- Sistema más manejable.
- Si falla una parte, el resto puede seguir funcionando.
- Como es un SO orientado a comunicaciones es fácil usarlo para diseñar Sistemas Distribuidos (Hay que gestionar los mensajes de forma que el cliente no deba conocer la dirección u ordenador del servidor):

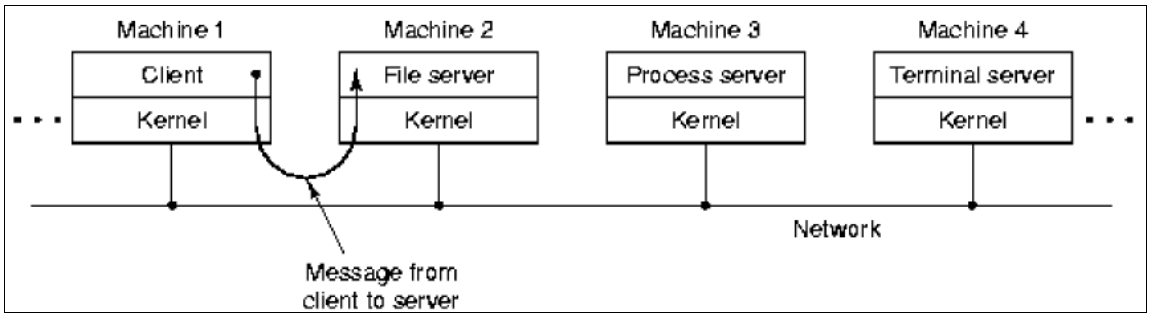

Figura 19: Esquema se provisión de servicios en sistema operativo cliente-servidor distribuido.

Las ventajas que presenta este tipo de estructura son, mas que nada, teóricas.

Ejem: Mach 3.0 (Universidad de Carnegie-Mellon), MINIX, UNIX actual, Windows NT/2000/XP (mezcla entre monolítico y cliente-servidor).

4.4. Estructura orientada a objetos

Existen diseños basados en la metodologías orientada a objetos.

El SO se ve como una colección de objetos (programas, periféricos, ficheros, funciones del SO , etc.), con interfaz y propiedades bien definidas.

El núcleo del SO será el responsable:

- Del mantenimiento de las definiciones de los tipos de objetos.
- Controlar los privilegios de acceso.

La interacción entre ellos viene definida por la capacidad de interactuar que tengan los unos con los otros.

Se puede representar como una red de objetos interconectados entre si por medio de capacidades de acceso a si mismos.

Hay una gran abstracción.

Ejem: Windows NT/XP

5. Tipos de SO

Los sistemas operativos se pueden clasificar según distintos puntos de vista:

1. Utilización de recursos.
2. Interactividad.
3. Número de usuarios.
4. Tipo de aplicación.

5.1. Según la utilización de recursos

Sistemas monoprogramados:

- Sólo se ejecuta un programa a la vez (el resto espera hasta la finalización **completa** del primero).
- Se carga inicialmente en memoria y permanece en ella hasta finaliza.

Sistemas multiprogramados o multitarea:

- Basados en técnicas de multiprogramación.
- Admiten varios procesos simultáneamente.
- Son los más extendidos en la actualidad.
- Según realicen la gestión del procesador, se clasifican en:
  - **Apropiativos**: el SO puede quitar el control del procesador al programa que se esté ejecutando.
  - **No Apropiativa**: el SO no puede tomar el control voluntariamente del procesador para poder decidir el programa que se debe ejecutar.

Sistemas de multiprocesamiento: varios procesadores interconectados.

5.2. Según su Interactividad

Procesamiento por lotes (*Batch*):

- Cada trabajo realiza un conjunto de pasos secuenciales relacionados entre sí (paquete de instrucciones).
- Todos los paquetes de un mismo trabajo se juntan para formar un lote (compilador + linkador + ejecución).
- Características:
  - No existe intervención del usuario durante la ejecución de los trabajos.
  - Procesamiento de trabajos largos.
  - No existe restricción de tiempo.

Tiempo compartido:

- Sistemas con **multiprogramación interactiva**, de forma que se permite al usuario, durante la ejecución de los programas, pedir los datos y aceptar sus respuestas.
- La entidad básica a controlar por el sistema son las **sesiones**. Una sesión comprende el intervalo de tiempo transcurrido desde el momento en que un usuario se identifica en el ordenador hasta que lo abandona despidiéndose del mismo.
- Al dar comienzo una sesión el sistema operativo establece un diálogo con el usuario a través de un intérprete de comandos.
- El usuario **cree** que todos los recursos del ordenador los tiene asignados y disponibles.
- Características:
  - Son muy conversacionales.
  - Permiten el acceso de varios usuarios al mismo tiempo, permitiéndoles compartir el ordenador.
  - Presentan un corto tiempo de respuesta.

Es bastante común la convivencia en los sistemas actuales de las dos técnicas, tiempo compartido y *batch*.

Tiempo real:

- Sistemas multiprogramados e interactivos de mayores exigencias.
- Se suelen emplear en aplicaciones dedicadas al control de instalaciones.
- Obtienen los datos de entrada de sensores los procesan y actúan rápidamente sobre el sistema a controlar, con objeto de producir los efectos necesarios para que se comporte correctamente.
- Un sistema trabaja en tiempo real si su tiempo de respuesta permite afectar (controlar y regular) al medio en el cual opera.
- Características:
  - Fuertes restricciones en el tiempo de respuesta.
  - La información tiene que estar permanentemente actualizada.
  - Los programas deben ejecutarse dentro de fuertes restricciones o, en otro caso, fallaríatodo el sistema.
  - Infrautilización: el sistema deberá permanecer prácticamente inactivo para atender con la mayor rapidez posible cualquier evento de entrada.

5.3. Número de usuarios

Sistemas monousuario:

- Sólo aceptan la conexión de un solo usuario en un momento dado.
- Pueden ser monoprogramados o multiprogramados.

Sistemas multiusuario:

- Aceptan más de una conexió simultánea.
- Basados en técnicas de multiprogramación.
- Normalmente serán sistemas de tiempo compartido aunque también podrían ser de tiempo real.

5.4. Tipo de aplicaciones

Sistemas de propósito general:

- Casi todos los SO son de este tipo.
- Dan servicio a un gran número de usuarios con una amplia variedad de tareas.

Sistemas de propósito especial:

- Construidos específicamente para una determinada aplicación.
- No suelen ser sistemas comerciales ni de gran difusión.