

# Tema 1.1

# Sistema Operativos (SSOO)

## Introducción

# Índice

- Definición de sistema operativo
- Funciones del SO
- Entorno virtual
- Un poco de historia

# Definición de SO/OS

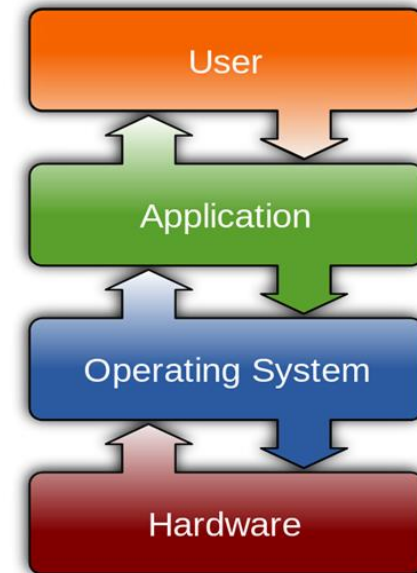
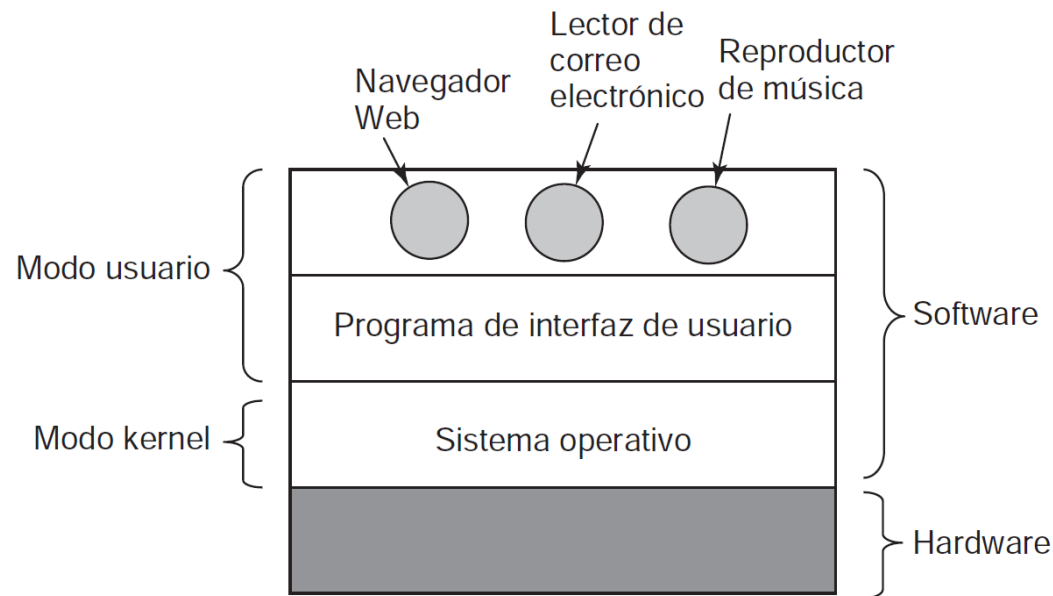
- Un ordenador moderno consta de uno o más procesadores, una memoria principal, discos, impresoras, un teclado, un ratón, una pantalla o monitor, interfaces de red y otros dispositivos de entrada/salida.
- Los ordenadores están equipados con una capa de software llamada **sistema operativo**, cuyo propósito es proporcionar a los programas de usuario acceso a todos los recursos y administrar cada uno de ellos.



# Definición de SO/OS

**Shell:** programa **basado en texto** con el que los usuarios generalmente interactúan

**GUI** (Graphical User Interface; Interfaz gráfica de usuario): cuando está basado elementos gráficos o iconos.



# Definición de SO/OS

- La mayoría de las computadoras tienen dos modos de operación: **modo kernel** y **modo usuario**.
- El SO se ejecuta en **modo kernel** (modo supervisor).
- En el modo kernel, el SO tiene acceso a todo el hardware.
- Las otras aplicaciones se ejecutan en **modo usuario**.
- Shell o GUI, es el nivel más bajo del software en modo usuario.  
**Graphic User Interface**
- No confundir con los **sistemas integrados** (*como los sistemas operativos basados en Java que para separar los componentes utilizan interpretación y no el hardware*).
- El trabajo del SO es **proporcionar una asignación** ordenada y controlada de los procesadores, memorias y dispositivos de E/S, entre los diversos programas que compiten por estos recursos.

# Definición de SO/OS

## Tres funciones básicas:

- Interfaz con el hardware: gestión de los recursos del ordenador
- Interfaz con las aplicaciones: gestión de los programas en ejecución (aplicaciones)
- Interfaz con el usuario: ejecución de las órdenes de los usuarios
  - Gestión de **procesos**
  - Gestión de **memoria**
  - Gestión del **sistema de ficheros**
  - **Seguridad**
  - Gestión de los dispositivos de **entrada y salida**
  - **Networking** (TCP/IP, UDP)

# Entorno virtual

## Virtualización del SO

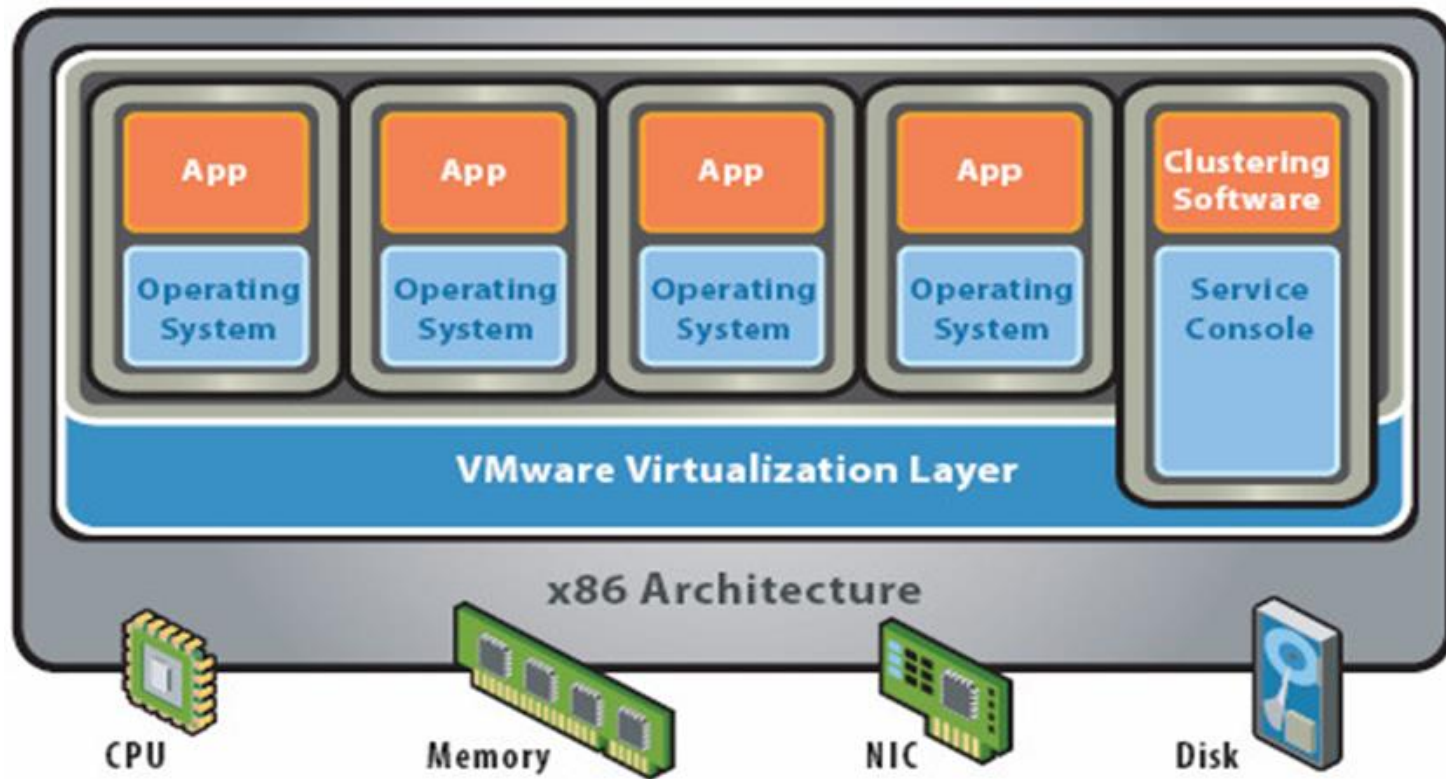
- VirtualBox - Vagrant, VMWare, Parallels, Docker



VMware Fusion



# Entorno virtual





# SO para la asignatura



debian



MINIX 3



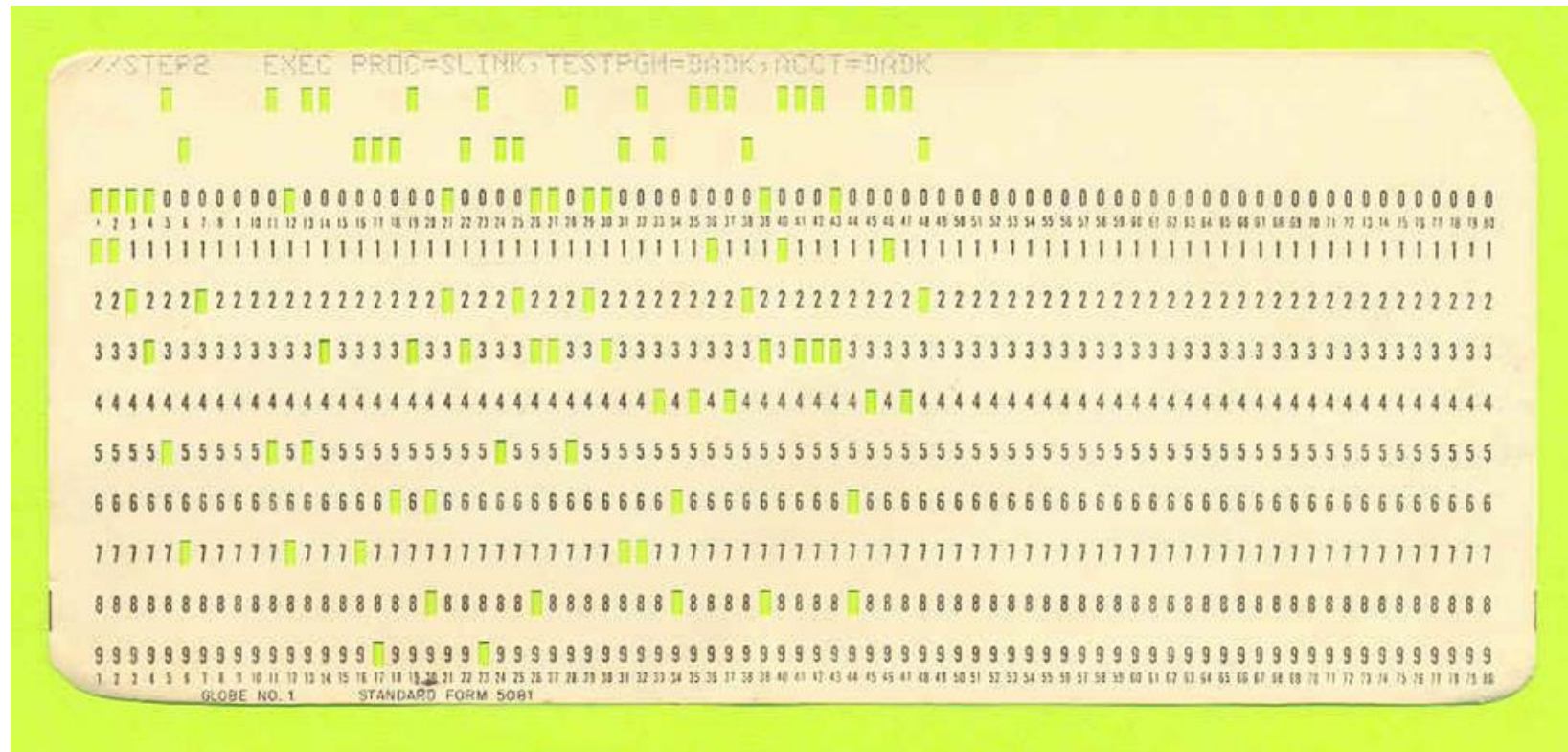
[Descargar SO](#)

# Un poco de historia

- Los primeros ordenadores se llamaban mainframes (IBM 704 de 1964).
- Los usuarios cargaban por turnos sus programas y datos mediante tarjetas perforadas
- El programa se cargaba, se ejecutaba y terminaba cuando el programa había concluido...
- O había fallado antes de terminar.
- Sistemas operativos por lotes (batch).



# Un poco de historia



# Un poco de historia

- Uno de los primeros sistemas operativos modernos con mayor aceptación fue **UNIX**
- Creado en los 70 por **Dennis Ritchie** y **Ken Thompson**
- Fue escrito en **C** (lenguaje creado por Ritchie entre los años 1969-73)
- Se caracterizó por su diseño modular.
- El SO provee un conjunto de herramientas más o menos sencillas y cada una es capaz de llevar a cabo ciertas tareas.
- Además, un sistema de ficheros unificado y jerárquico
- Dispone de una Shell o intérprete de comandos.
- ¿Quién coordina a todas esas herramientas?
  - El núcleo o kernel
- Además, era portable, **multiusuario** y **multitarea**



# Un poco de historia

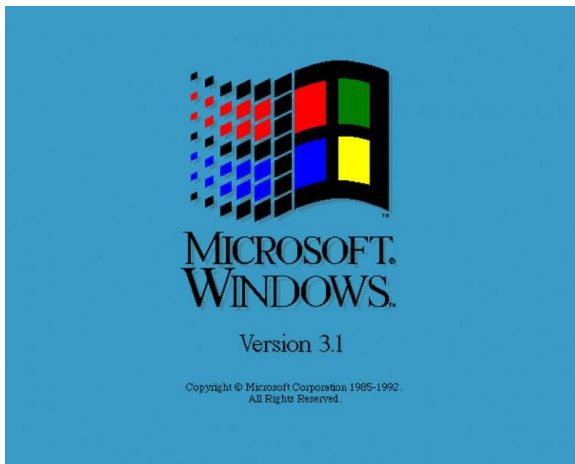
- En los 80 se empezó a pensar que un SO podría usarse en ordenadores de cualquier tamaño.
- Nace así **MS-DOS**: Seattle Computer Products tenía un sistema operativo conocido como **DOS** (Disk Operating System; Sistema Operativo en Disco).
- BGates compró DOS y lo ofreció a IBM como DOS/BASIC.
- IBM pidió modificaciones y BGates proporcionó MS-DOS (Microsoft Disk Operating System; Sistema Operativo en Disco de Micro-Soft)
- Doug Engelbart en el Stanford Research Institute inventó la Interfaz Gráfica de Usuario GUI.
- Xerox PARC copió la idea y la utilizó en sus sistemas.
- El primer intento de Job surgió de lo visto en PARC
- Su segundo intento generó Apple Macintosh.
- La evolución de MS-DOS estuvo influenciada por el éxito Apple, de ahí lo de Windows





# Un poco de historia

- Entre 1985 y 1995, Windows fue sólo un entorno gráfico encima de MS-DOS.
- En 1995 se liberó una versión independiente de Windows, conocida como Windows 95.
- En 1998, se liberó una versión ligeramente modificada de este sistema, conocida como Windows 98.
- Sin embargo, tanto Windows 95 como Windows 98 aún contenían una gran cantidad de lenguaje ensamblador para los procesadores Intel de 16 bits.



# Un poco de historia

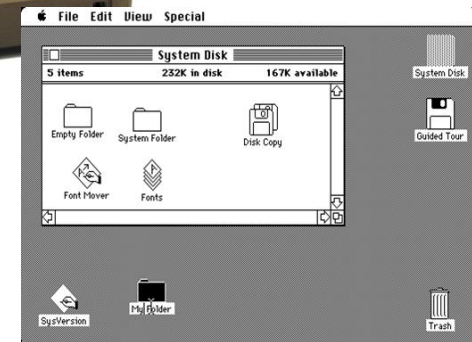
- Otro de los sistemas operativos de Microsoft es Windows NT (NT significa Nueva Tecnología), que es compatible con Windows 95 en cierto nivel, pero fue completamente rediseñado en su interior. Es un sistema **completo de 32 bits**.
- En los 80 aparecen los ordenadores-consolas con sus propios sistemas operativos.



Amiga Commodore



Mac OS



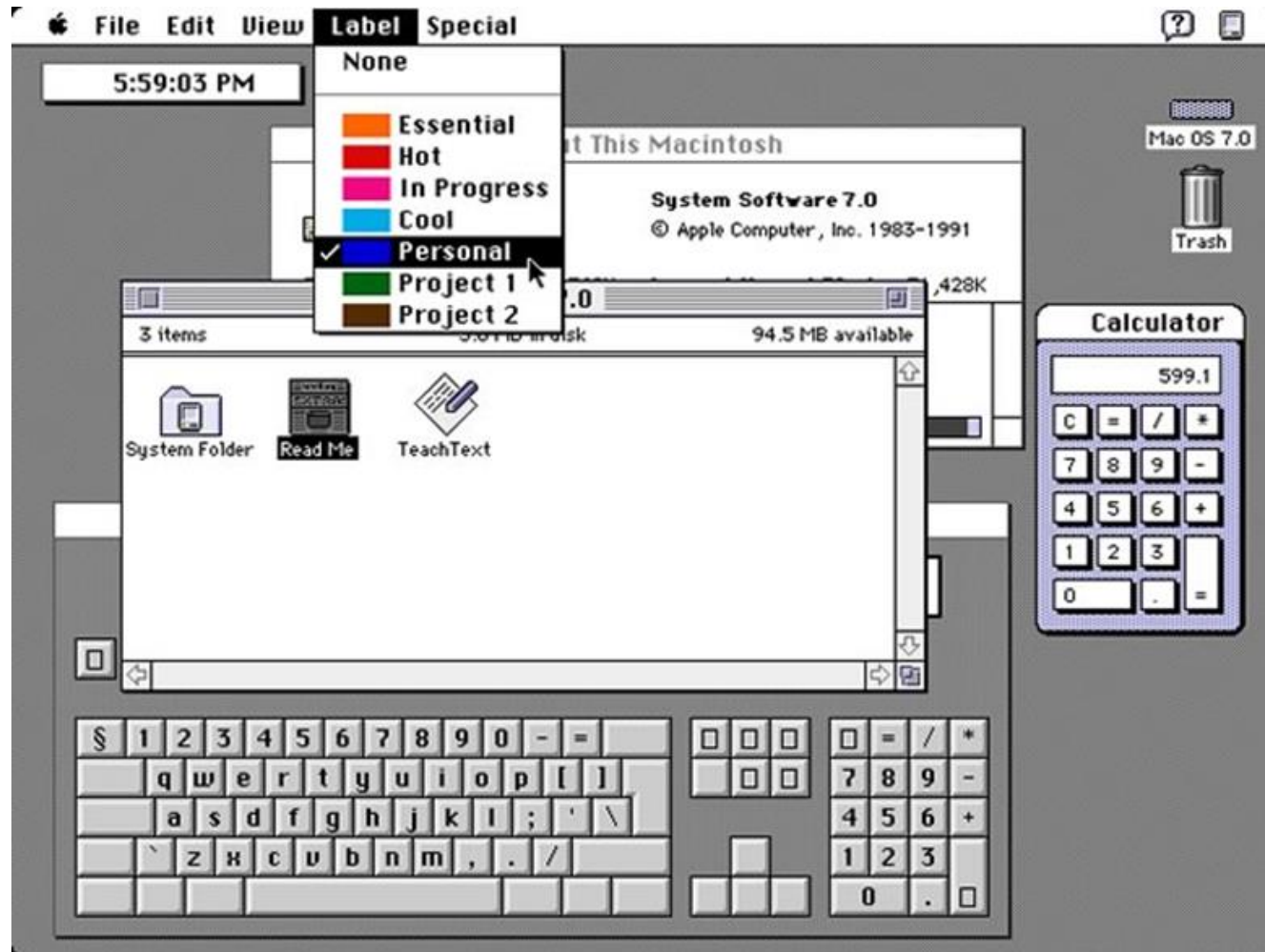
# Un poco de historia

- El OS/2 de IBM (aunque también funcionaban con MSDOS y Windows).
- La clave: El desarrollo de la GUI (Graphical User Interface)
- La del Mac OS 7 fue la primera que soportaba los colores

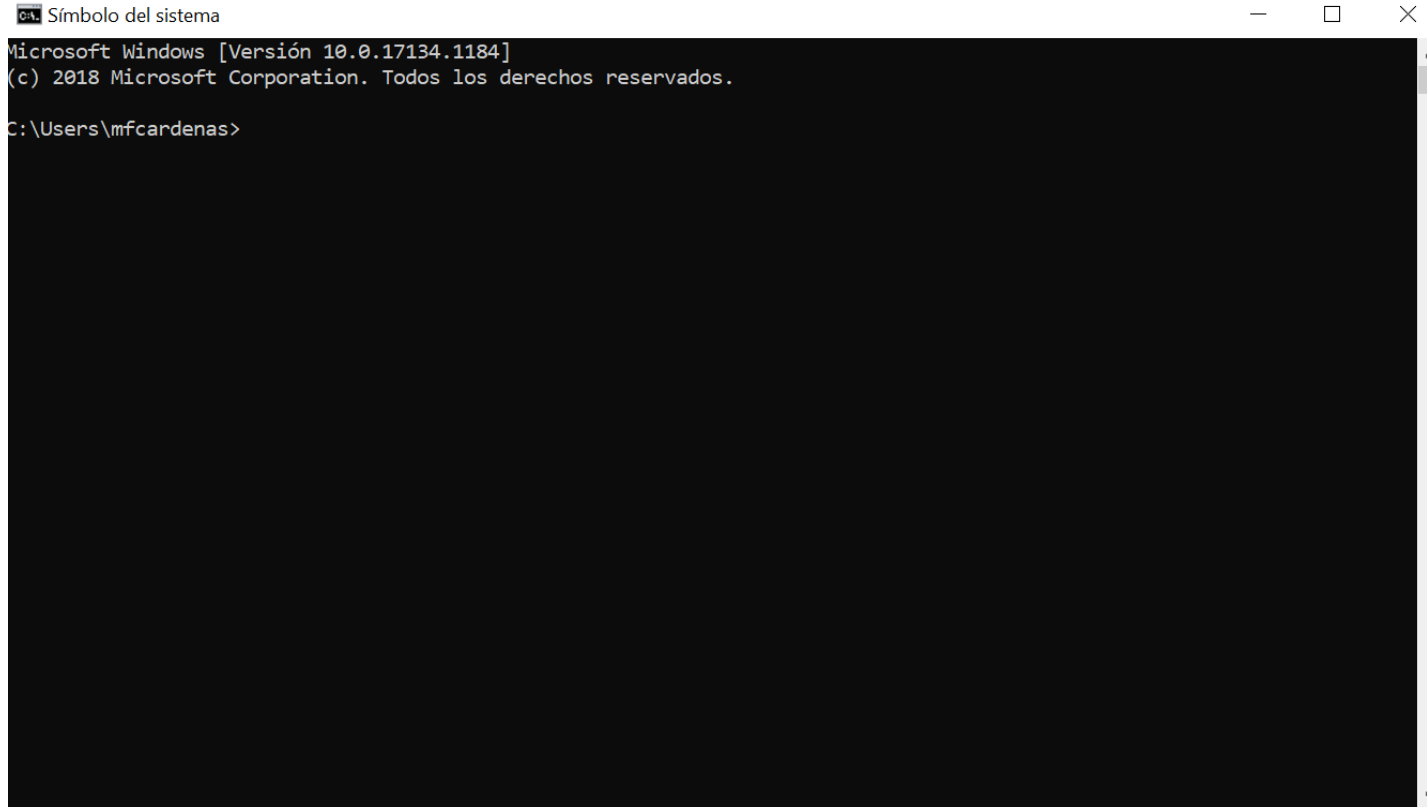




# Un poco de historia



# Recordemos MS-DOS



A screenshot of a Windows Command Prompt window. The title bar at the top reads 'Símbolo del sistema' with standard Windows window controls (minimize, maximize, close) on the right. The main area is black with white text. The text displayed is: 'Microsoft Windows [Versión 10.0.17134.1184]', '(c) 2018 Microsoft Corporation. Todos los derechos reservados.', and 'C:\Users\mfcardenas>'.

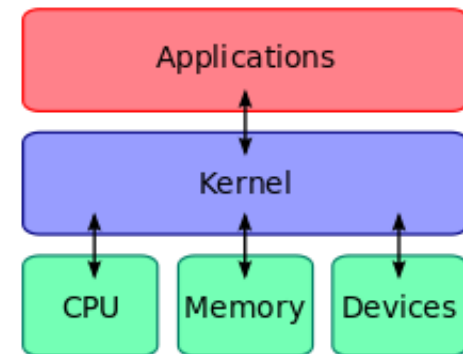
```
Microsoft Windows [Versión 10.0.17134.1184]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\mfcardenas>
```

Escribir "CMD" en aplicaciones  
de Windows

# Kernel

- La capa más cercana al hardware se denomina **kernel** (o núcleo).
- Es la parte fundamental del SO.
- Gestiona los recursos hardware del sistema
- Provee acceso seguro al hardware para los diferentes programas que se ejecutan.
- Y es responsable de decidir **cuándo y por cuánto tiempo** un programa puede acceder a cierto hardware.
- Se encarga de:
  - Gestión de procesos
  - Memoria principal
  - Memoria de disco
  - E/S, Sistema de ficheros
  - Seguridad
  - Red



# Arquitecturas del SO

Tipos de arquitecturas del SO según la definición del kernel:

- Arquitectura **kernel monolítico**
- Arquitectura **microkernel**
- Arquitecturas **híbridas**

# Arquitecturas del SO: kernel monolítico

- Gran parte de la funcionalidad reside en el kernel
- Incluyendo planificación, sistema de ficheros, redes, drivers de dispositivos, gestión de memoria,...
- Suele estar implementado como un único proceso que se ejecuta en modo kernel.
- El diseño es modular, como en UNIX
- Cuando el sistema arranca se cargan todos los servicios del SO y se quedan en memoria.
- Proceso init.

# Arquitecturas del SO: kernel monolítico

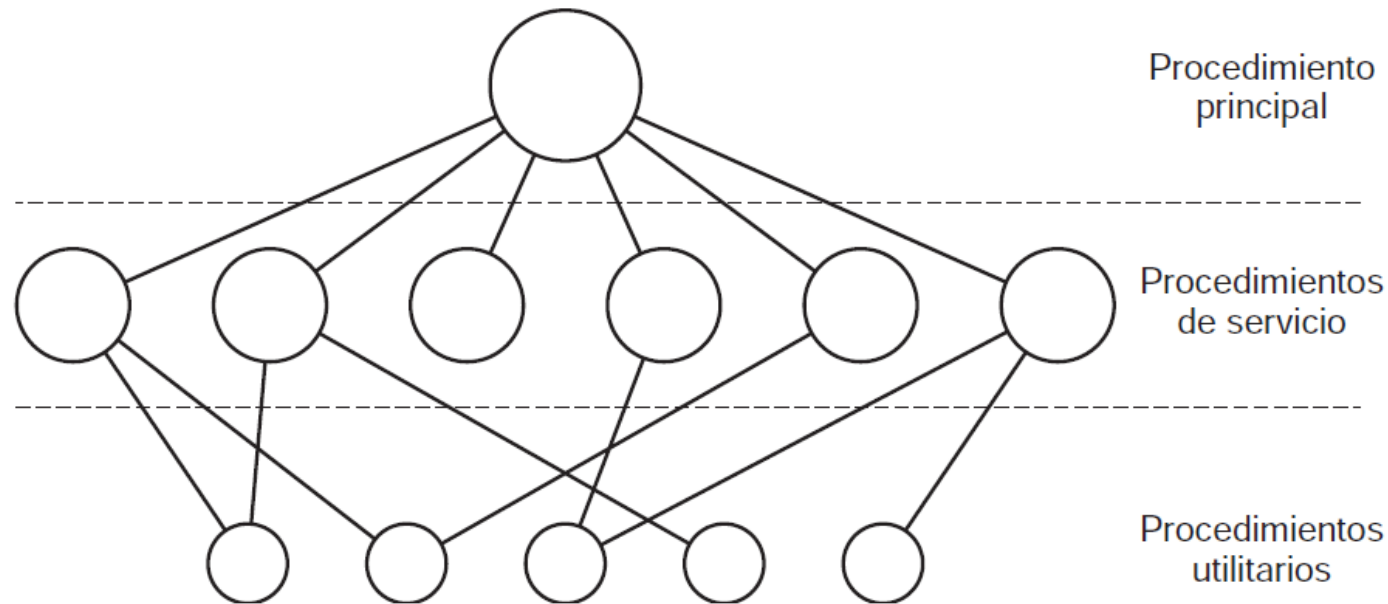
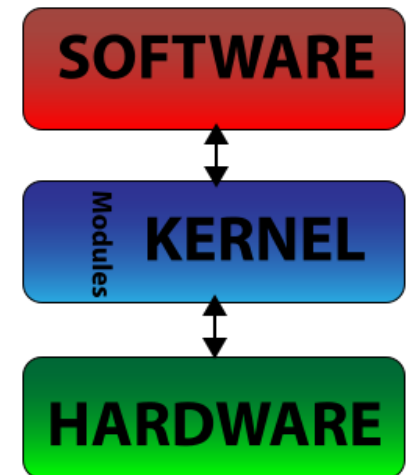


Figura 1-24. Un modelo de estructuración simple para un sistema monolítico.



# Arquitecturas del SO: kernel monolítico

## Desventajas

- Podemos tener kernels demasiado grandes.
- Los errores en el kernel pueden paralizar el sistema.
- Si un driver falla puede hacer caer todo el sistema.
- Poco flexible.
- Pero es muy rápido.
- Cada procedimiento en el sistema tiene la libertad de llamar a cualquier otro, si éste proporciona cierto cómputo útil que el primero necesita.
  - Al tener miles de procedimientos que se pueden llamar entre sí sin restricción, con frecuencia se produce un sistema poco manejable y difícil de comprender.

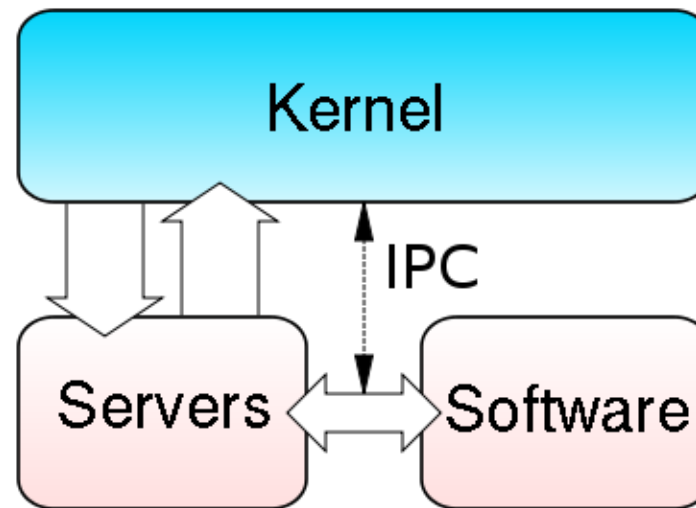
# Arquitecturas del SO: Microkernel

- Los procesos de usuario se pueden configurar para que tengan menos poder, por lo que un error en ellos tal vez no sería fatal.
- Asigna solamente unas pocas funciones al kernel (modo privilegiado o modo kernel).
- Planificación básica, comunicación entre procesos (IPC),...
- El resto sigue una arquitectura cliente – servidor.
- Estos se ejecutan en modo usuario o no privilegiado.



# Arquitecturas del SO: Microkernel

- Los servicios del SO los proporcionan determinados procesos, algunas veces llamados **servidores**.
- Los servicios del SO se ejecutan en modo usuario, y el microkernel los trata como a cualquier otra aplicación: e.g. *Servidor X (GUI), drivers...*



# Cultura general

- La densidad de los errores depende del tamaño del módulo, su tiempo de vida...
- Aproximada en los sistemas industriales formales existen **10 errores** por cada **1.000 líneas** de código.
- Es probable que un SO monolítico de **cinco millones de líneas de código** contenga cerca de **50,000 errores** en el kernel.
- No todos son graves, algunos errores pueden ser: emitir un mensaje de error incorrecto en una situación poco común.
- Los SO tienen tantos errores que los fabricantes de ordenadores colocan botones de reinicio “reset” en ellas (a menudo en el panel frontal).
  - No pasa lo mismo en equipos como televisiones, estéreos, coches, etc. pese a la gran cantidad de software que hay en estos dispositivos.

# Arquitecturas del SO: Microkernel

- Sólo el microkernel se ejecuta en modo kernel.
- El resto se ejecuta como procesos de usuario ordinarios, sin poder relativamente.
- Al ejecutar cada driver de dispositivo y sistema de archivos como un proceso de usuario separado, un error en alguno de estos procesos puede hacer que falle ese componente, pero no puede hacer que falle todo el sistema.
- Un error en el driver del dispositivo del audio hará que el sonido sea confuso o se detenga, pero el ordenador seguirá funcionando.
- En un **sistema monolítico** con todos los drivers en el kernel, un driver de audio con errores puede hacer uso de una dirección de memoria inválida y paralizar todo el sistema.
  - Algunos de los microkernel mejor conocidos son Integrity, K42, L4, PikeOS, QNX, Symbian y MINIX 3.

# Cultura general: Minix 3

- MINIX 3 es un sistema de código fuente abierto en conformidad con POSIX.
  - Sergio: ¿Qué es POSIX?
- El microkernel MINIX 3 sólo tiene cerca de 3200 líneas de C y 800 líneas de ensamblador para las funciones de muy bajo nivel (se usan para atrapar interrupciones y conmutar proceso).
- El código de C administra y planifica los procesos, se encarga de la comunicación entre procesos (al pasar mensajes entre procesos) y ofrece un conjunto de aproximadamente **35 llamadas** al kernel para permitir que el resto del SO realice su trabajo.
- Estas llamadas realizan funciones como: asociar los drivers a las interrupciones, desplazar datos entre espacios de direcciones e instalar nuevos mapas de memoria para los procesos recién creados.

# Arquitecturas del SO: Microkernel

## Ventajas

- Buen diseño
- Protección entre los distintos componentes del sistema
- Kernel pequeño y flexible
- Pero más lento

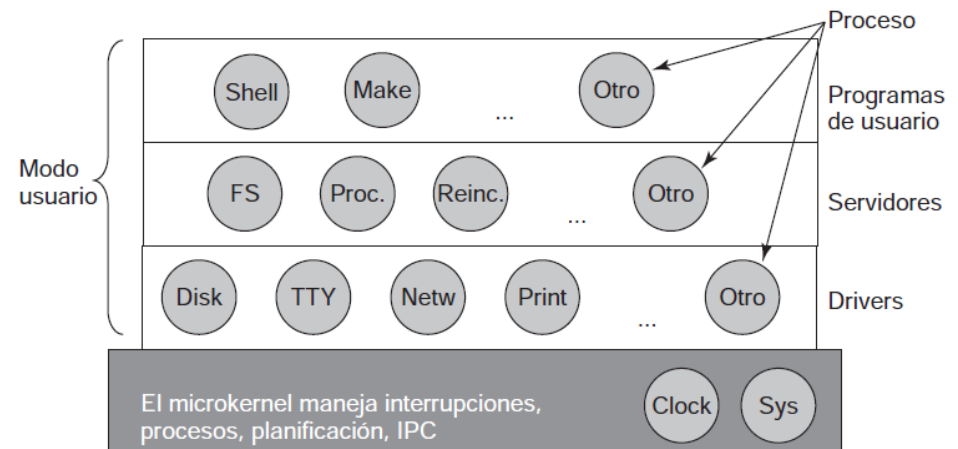


Figura 1-26. Estructura del sistema MINIX 3.

# Arquitecturas del SO: Microkernel

## Desventajas

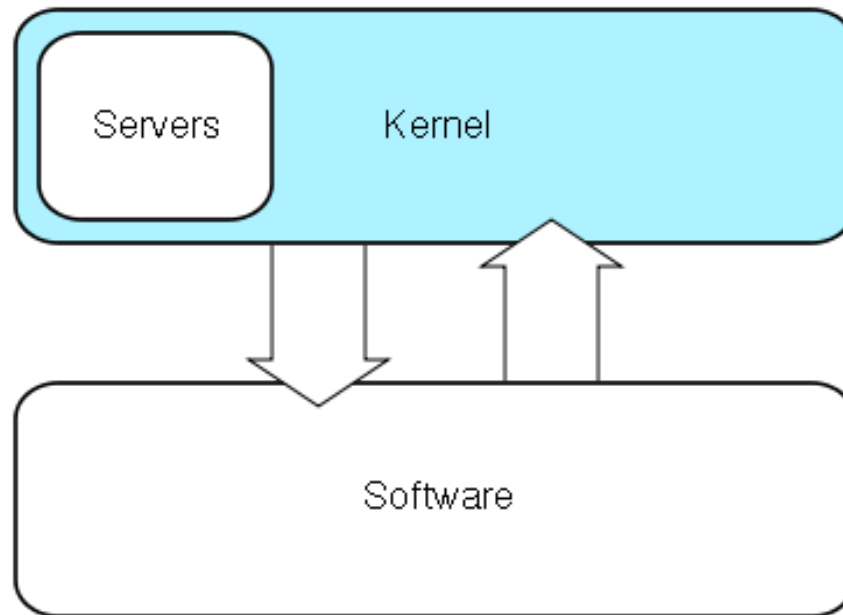
- Se producen muchísimas llamadas al sistema (más carga en el sistema), más pesado.
- Es difícil la gestión entre procesos
  - **IPC – Inter-Process Communication**
- Y muchos cambios entre los dos modos de ejecución (modo kernel – modo usuario), que penalizan el rendimiento
- Ejemplo: Amiga OS

# Arquitecturas del SO: híbrida

- Los **kernels híbridos** son los usados en la mayoría de los sistemas operativos comerciales.
- Windows (desde Windows NT), Mac OS X.
- Es similar al microkernel y al diseño cliente-servidor, pero con algunas mejoras.
- Se incluye parte del código en modo kernel para que mejore el rendimiento.
- Estas optimizaciones se llevaron a cabo a mediados de los noventa.

# Arquitecturas del SO: híbrida

- Se mejoró mucho el rendimiento
- Y por ello esta arquitectura fue la adoptada por la mayoría de los sistemas operativos.





# Arquitecturas del SO: otros

- Según el número de procesos simultáneos que pueden ejecutar:
  - **Monotarea** (MS-DOS)
  - **Multitarea** (el SO debe repartir el tiempo del procesador entre los diferentes procesos)
- Según el número de usuarios simultáneos:
  - **Monousuario** (Windows)
  - **Multiusuario** (Linux)

# Arquitecturas del SO: otros

- Según el número de procesadores que puede atender:
  - **Monoprocesador**
  - **Multiprocesador**
- Según el tipo de arquitectura del procesador que puede gestionar:
  - 16 bits
  - **32 bits**
  - **64 bits**

# Arquitecturas del SO: otros

- Según el uso:
  - Cliente
  - Servidor
  - Empotrado o
  - de tiempo real
- Según la movilidad:
  - Fijos
  - Móviles.

# Bibliografía

- **CARRETERO**, Jesús, **GARCÍA**, Félix, **DE MIGUEL**, Pedro, **PÉREZ**, Fernando. Sistemas Operativos: una visión aplicada. McGraw-Hill, 2001.
- **STALLINGS**, William. Sistemas operativos: aspectos internos y principios de diseño. 5ª Edición. Editorial Pearson Educación. 2005. ISBN: 978-84-205-4462-5.
- **TANENBAUM**, Andrew S. Sistemas operativos modernos. 3ª Edición. Editorial Prentice Hall. 2009. ISBN: 978-607- 442-046-3.

