Tema 2

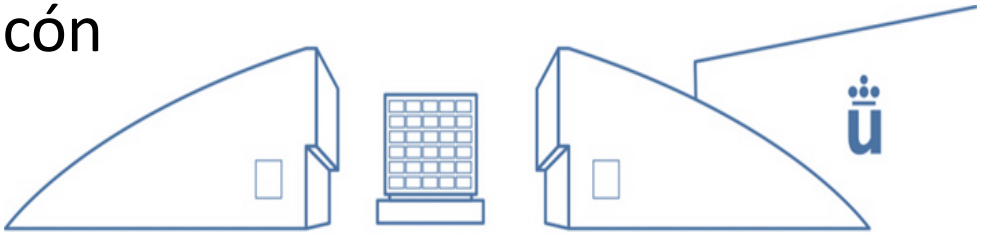
Introducción a los Sistemas Operativos

Alberto Sánchez

Sofia Bayona

Luis Rincón

Universidad Rey Juan Carlos



Índice

- Bibliografía.
- ¿Qué es un sistema operativo?
- Estructura y funcionamiento del computador.
- Estructura de un sistema operativo
- Servicios del sistema operativo
- Diseño e implementación
- Estructuras

Bibliografía

- **“Fundamentos de sistemas operativos”** *Silberschatz, Galvin, Gagne*. Editorial Mc Graw Hill.
  - Capítulo 1: 1.1, 1.2, 1.4, 1.5, 1.6, 1.7, 1.8
  - Capítulo 2: 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7

Índice

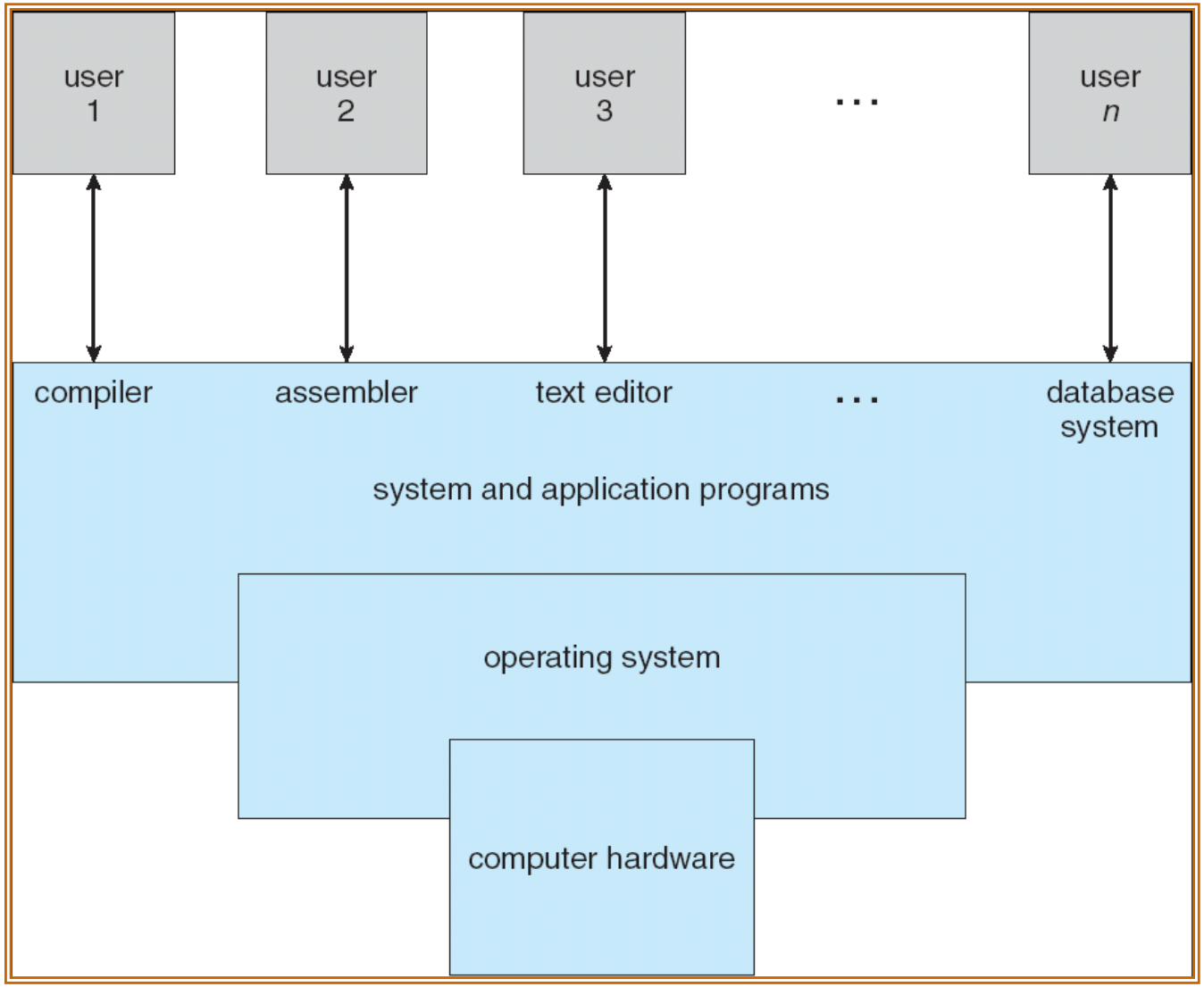
¿Qué es un sistema operativo?

- Es un programa que hace de intermediario entre el usuario y el hardware de un computador.
- Objetivos del sistema operativo:
  - Gestiona la ejecución de los programas del usuario.
  - Hacer que el computador sea cómodo de usar.
  - Utilizar los recursos del computador de forma eficiente.
- Tareas del sistema operativo
  - Comunicación con los periféricos
  - Administración de recursos
  - Coordinación de procesos simultáneos (multiproceso y multitarea)
  - Gestión de memoria
  - Gestión de datos, comunicaciones y redes

Componentes de un sistema informático

- Usuarios
- Programas:
  - Utilizan los recursos del computador para resolver las necesidades de los usuarios.
    - Procesadores de texto, navegadores, juegos . . .
- Sistema operativo:
  - Coordina el uso del hardware entre varios programas y usuarios.
- Hardware:
  - Proporciona los recursos de cómputo.
    - Procesador, memoria, dispositivos de entrada salida . . .

Componentes de un sistema informático



Punto de vista del usuario

- Muchos tipos diferentes de computadores:
  - Computadores personales.
    - Un único usuario. El S.O está diseñado para que sea de fácil uso.
  - Dispositivos móviles.
    - Limitaciones de alimentación, velocidad e interfaz. El S.O está diseñado para proporcionar usabilidad. El consumo es importante.
  - Computadores compartidos (*mainframes*, *servidores, …*).
    - Varios usuarios simultáneos. El S.O está diseñado para maximizar el uso de los recursos y repartirlos equitativamente entre los usuarios.

Punto de vista del programador

- El S.O proporciona al programador una serie de abstracciones que le permiten acceder al hardware de forma sencilla:
- Ejemplo:
  - Para acceder a un disco duro es necesario controlar:
    - Dirección del bloque.
    - Controlar códigos de estado y error.
    - ….
  - El S.O proporciona funciones para abrir un fichero, leer, escribir en el fichero, cerrar el fichero, etc.

Definición

- No existe una definición universal de “Sistema Operativo”
- “Todo lo que el vendedor te da cuando le pides un sistema operativo” puede ser una aproximación, pero varía mucho de un sistema a otro.
- “El programa que **siempre** está ejecutándose en el computador”. Usualmente se le llama *kernel* o núcleo. Todo lo demás son programas del sistema y programas de aplicación.

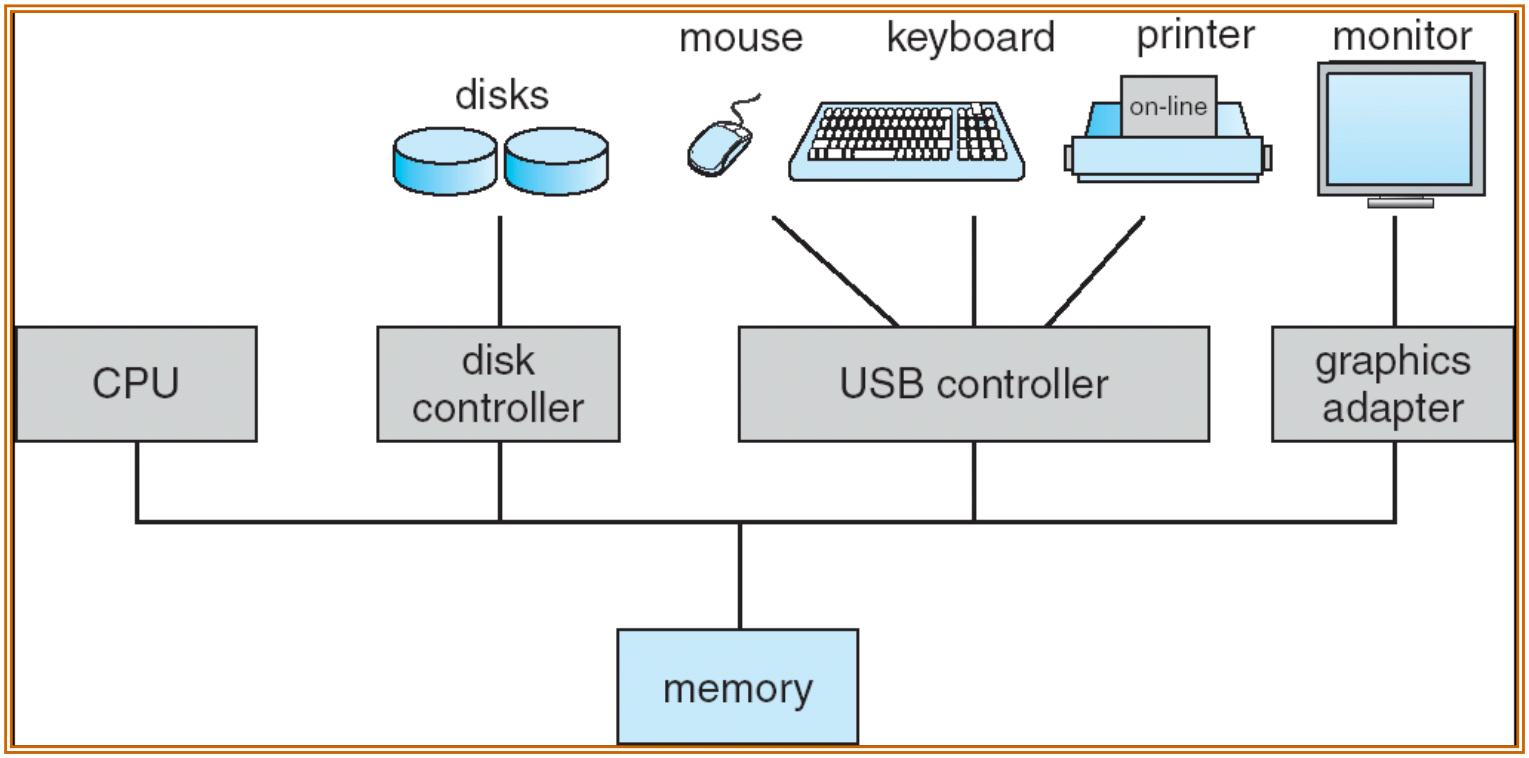
Índice

Arranque del computador

- El programa de arranque se ejecuta al encender o reiniciar el computador:
  - Está almacenado en algún tipo de ROM y se conoce generalmente como *firmware.*
  - Inicializa distintos aspectos del computador (registros, controladoras de dispositivos, memoria, etc)
  - Busca en el disco la imagen del *kernel*, la carga en memoria y comienza su ejecución.

Funcionamiento del computador

- PC de propósito general:
  - Una o varias CPUs y controladoras de dispositivos, todos conectados a un bus común para acceder a la memoria.



Funcionamiento del computador

- Los dispositivos de E/S y las CPUs funcionan de forma paralela.
- Los dispositivos están gestionados mediante una controladora (*controller*)
  - Cada *controller* se encarga de un tipo concreto de dispositivo.
  - Los *controller* tienen un buffer local.
- La CPU se encarga de mover datos entre la memoria principal y los buffers locales de los *controllers*.
- Los *controllers* utilizan interrupciones para informar a la CPU cuando han finalizado su trabajo.

Funcionamiento del computador

- La interrupción transfiere el control a la rutina de atención de la interrupción, generalmente a través del vector de interrupciones.
- Se almacena la dirección de la instrucción interrumpida en algún sitio (depende de la arquitectura).
- Mientras se está atendiendo una interrupción, se deshabilita la llegada de nuevas interrupciones.

Funcionamiento del computador

- **Los sistemas operativos actuales basan su funcionamiento en las interrupciones.**
  - El sistema operativo es el encargado de preservar el estado del procesador, guardando el contenido de los registros y del contador de programa, para poder restaurarlo cuando finalice la interrupción.
  - El kernel puede ser visto como el código de las llamadas de tratamiento de interrupción.
- Además de las interrupciones hardware existen:
  - Interrupciones software: causadas por una petición del usuario
  - Excepciones: causadas por un error

Funcionamiento del computador (E/S)

- Gran parte del S.O. se dedica a gestionar la E/S.
- Los controladores *(drivers)* son el software que permite al S.O comunicarse con el hardware a través de los *controllers*.
- El S.O tiene un *driver* para cada *controller*, que proporciona una interfaz uniforme mediante la cual comunicarse con el dispositivo hardware.

Estructura de almacenamiento

- El sistema operativo se encarga de administrar la memoria principal y la secundaria.
- **Memoria principal**:
  - Es el único medio de almacenamiento grande que puede ser accedido directamente desde el procesador.
  - Los programas deben estar en memoria principal para poder ser ejecutados.
- **Memoria secundaria:**
  - Proporciona un medio de almacenamiento no volátil de gran capacidad.

Estructura de almacenamiento

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

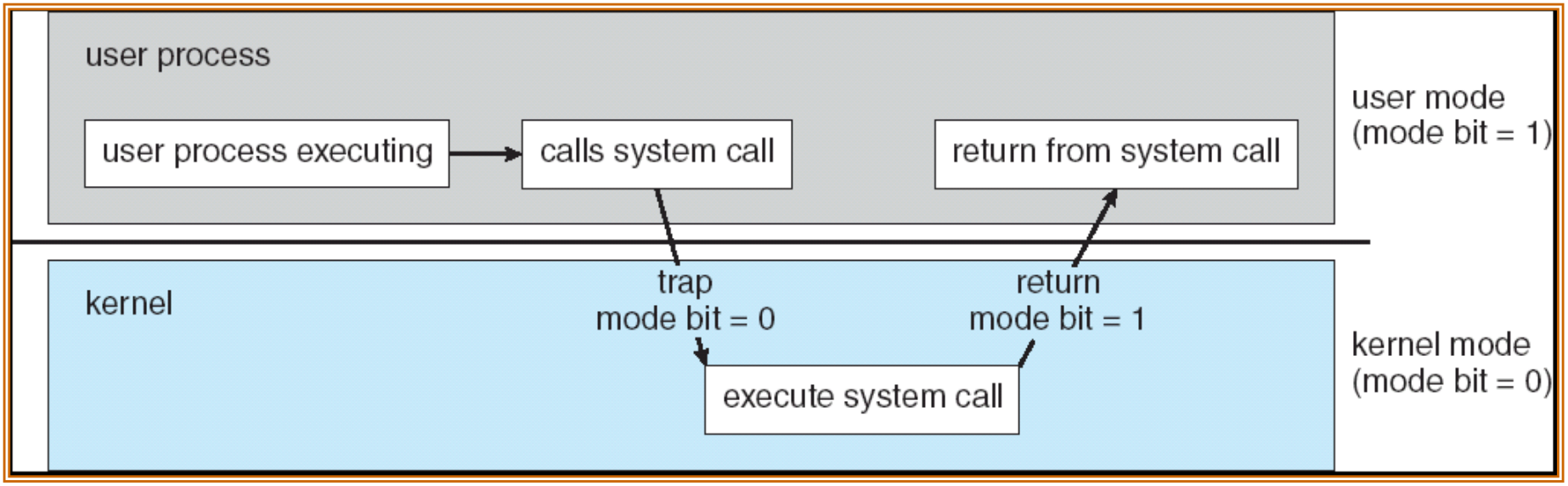
Operaciones del sistema operativo

- El S.O operativo y los usuarios comparten los recursos hardware y software del sistema informático.
- Los errores de un programa pueden afectar a otros programas o al propio S.O.
  - Un programa tarda demasiado tiempo en terminar o no termina nunca (por ejemplo debido a un bucle infinito) => Poner temporizador
  - Algunas instrucciones “peligrosas” no deberían poder ser ejecutadas directamente por programas de usuario => Modo dual

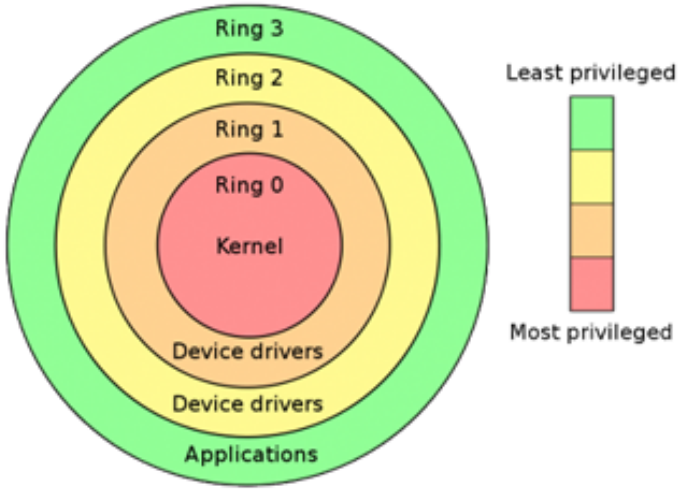
Modo dual

- El modo dual es una característica hardware que permite al S.O operativo protegerse:
  - El procesador tiene dos modos de funcionamiento indicados mediante un bit:
    - Modo usuario (1)
    - Modo kernel (0, también llamado modo supervisor o modo privilegiado).
  - Las instrucciones del procesador más delicadas solo pueden ejecutarse si el procesador está en modo kernel.
  - Las llamadas al sistema son la interfaz que proporciona el S.O. para que los programas de usuario puedan hacer uso de forma segura de determinadas partes del sistema informático.

Modo dual


user process
user process executing
calls system call
return from system call
user mode
(mode bit = 1)
kernel
trap
mode bit = 0
return
mode bit = 1
execute system call
kernel mode
(mode bit = 0)

Modo dual



- CPUs x86 proveen un rango mayor de niveles de protección que el modo dual, llamados rings
- Ring 0 tiene el más alto nivel de privilegio (modo kernel)
- Ring 3 correspondería al modo usuario
- Ring 1 y 2 no se suelen usar habitualmente, aunque se pueden utilizar para temas de virtualización

- La multiprogramación aumenta la eficiencia de un sistema informático:
  - Un solo usuario no mantiene ocupado el procesador y los dispositivos de E/S continuamente.
  - La multiprogramación organiza los trabajos de tal forma que el procesador se mantiene trabajando el máximo tiempo posible.
  - Cuando un trabajo tiene que esperar (por una operación de E/S por ejemplo), el sistema operativo pone a ejecutar otro trabajo en el procesador.
    - **Si no hay nadie queriendo ejecutar, el SO ejecuta el proceso nulo**

Tiempo compartido

- El tiempo compartido es una extensión de la multiprogramación.
  - El sistema asigna un tiempo máximo de ejecución continuada a cada proceso.
    - Temporizador.
  - Eso permite que el sistema operativo cambie tan rápidamente de trabajo que el usuario puede interaccionar con cada uno de ellos mientras se ejecuta, sin tener sensación de pausa o parada.
    - Se denomina planificación de la CPU
  - El tiempo compartido permite también que haya más de un usuario utilizando el sistema.

Índice

Servicios del sistema operativo

- Una parte de los servicios que ofrecen los sistemas operativos proporcionan funciones que son útiles para los usuarios:
  - Interfaz de usuario (UI, *user interface*): existen de varios tipos: línea de comandos (CLI), interfaz gráfica (GUI), proceso por lotes (*batch*)
  - Ejecución de programas.
  - Operaciones de E/S.
  - Manipulación de archivos.
  - Comunicaciones.
  - Detección de errores.
- Otras funciones del sistema operativo están pensadas para garantizar la eficiencia del propio sistema:
  - Asignación de recursos.
  - Contabilidad.
  - Protección y seguridad

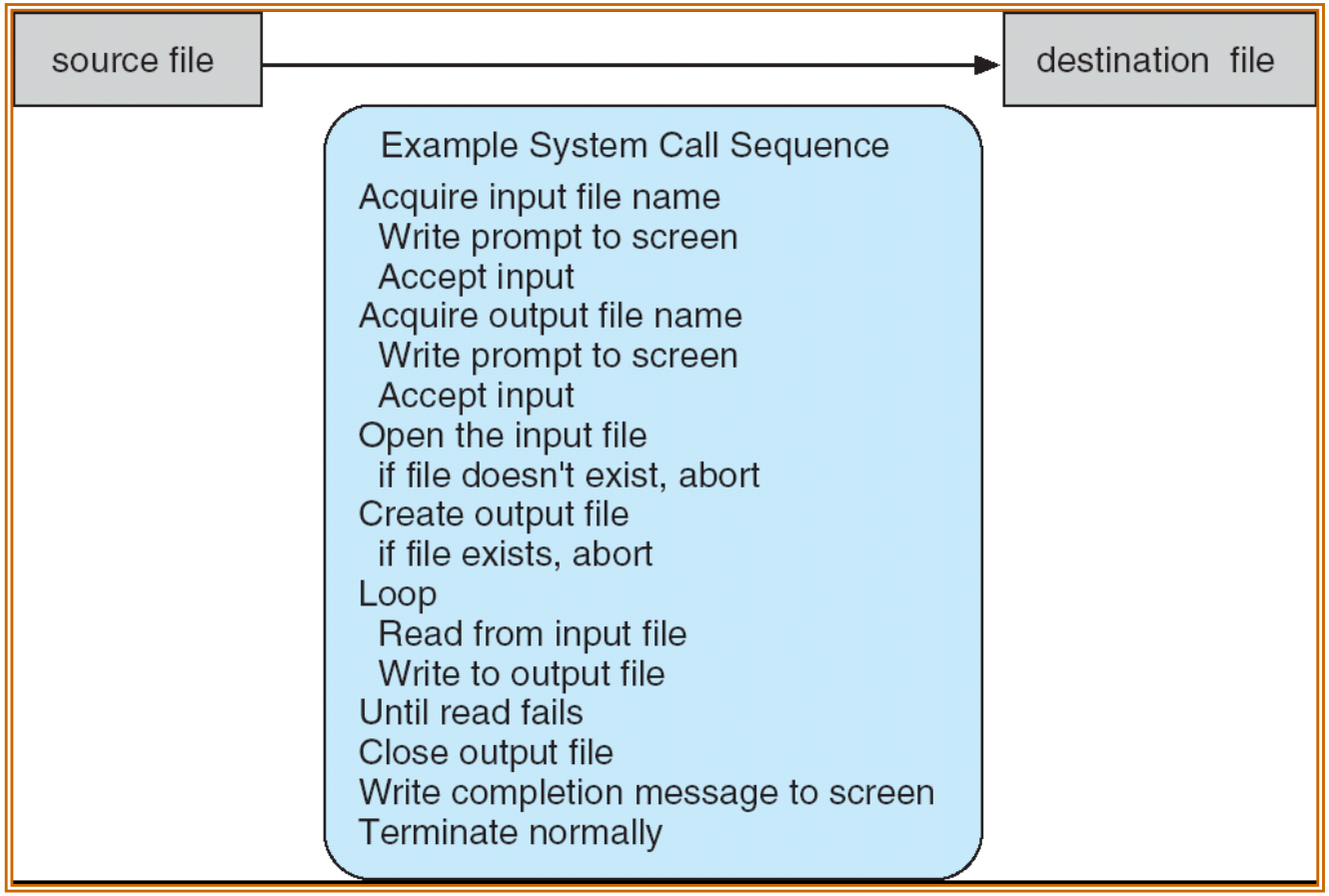
Interfaz de usuario

- Permite al usuario interactuar con el sistema operativo.
- Interfaz modo texto (shell) -> Interprete de mandatos:
  - Su función principal es interpretar mandatos introducidos por el usuario y ejecutarlos.
  - Implementación:
    - Mandatos como parte del propio shell
    - Mandatos como programas del sistema. Unicámente se identifica el archivo a cargar
- Interfaz gráfica (GUI):
  - Sistema de ventanas que permiten al usuario interactuar con el sistema.
  - Distintos tipos de periféricos para interactuar con el sistema:
    - Los de toda la vida: teclado y ratón.
    - Más modernos: pantallas multitáctiles.
    - Lúdicos: hápticos, tabletas digitalizadoras, alfombras de baile . . .

Llamadas al sistema

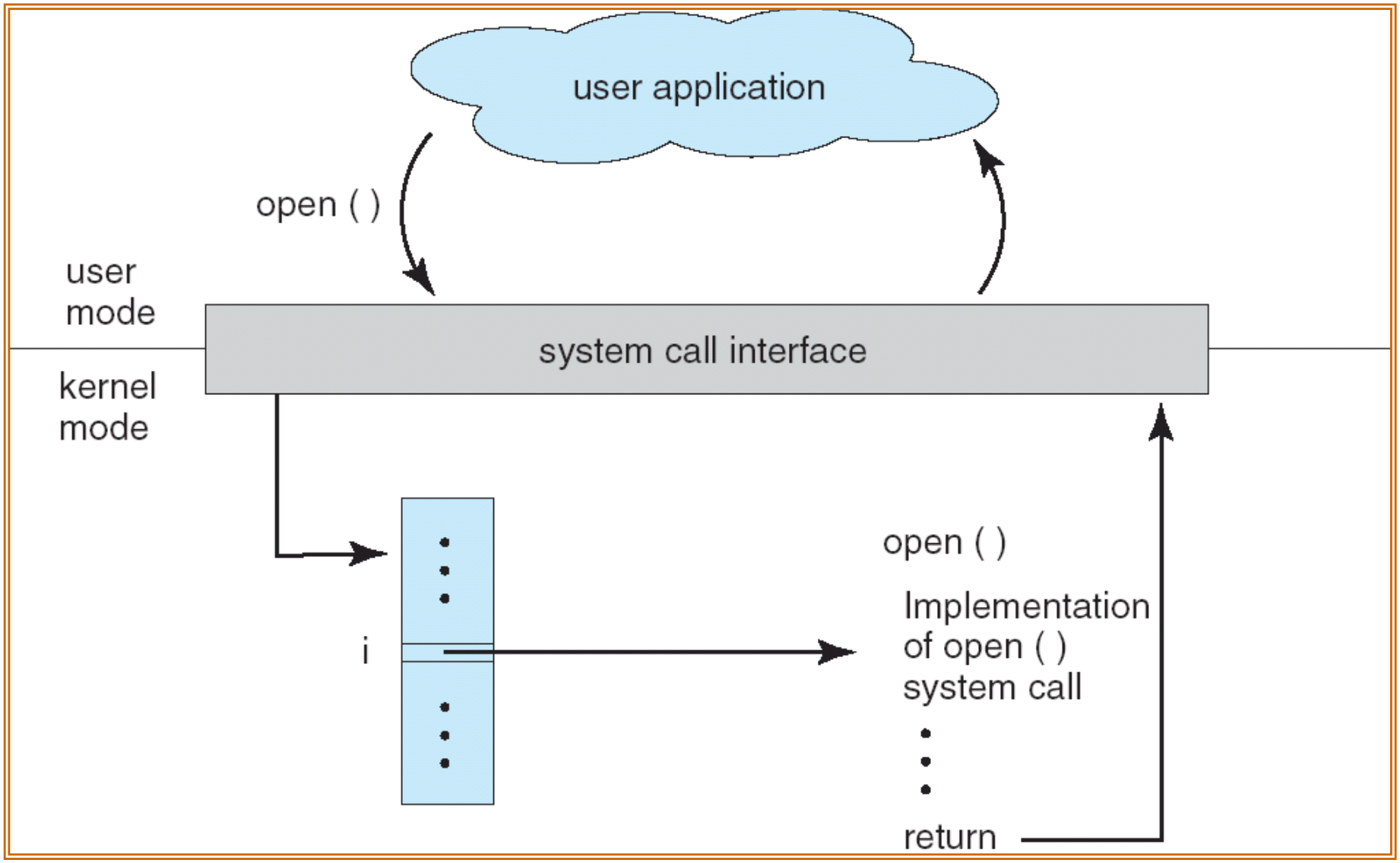
- Proporcionan una interfaz de programación para invocar los servicios que ofrece el S.O.
- Normalmente están escritas en un lenguaje de programación cercano a la máquina (C o C++).
- El programador no suele utilizar las llamadas al sistema directamente, sino que utiliza una API de más alto nivel.
- Las APIs más comunes son:
  - Win32 API en sistemas Windows
  - POSIX en sistemas UNIX, GNU Linux y macOS.

- Ejemplo: secuencia de llamadas al sistema para copiar un archivo.



- Implementación:
  - Lo normal es tener asociada cada llamada al sistema a un número, y se mantiene una tabla indexada con las direcciones de las llamadas.
  - El interfaz de llamadas al sistema invoca a la llamada correspondiente del sistema operativo, y devuelve el estado y los valores de retorno.
  - El programador no necesita saber como está implementada la llamada al sistema, sólo necesita utilizar la API y entender que es lo que hará el S.O como resultado de la llamada.

Llamadas al sistema



Llamadas al sistema

- Cinco categorías principales:
  - Control de procesos.
  - Administración de archivos.
  - Administración de dispositivos.
  - Mantenimiento de información.
  - Comunicaciones (memoria compartida, paso de mensajes)

Programas del sistema

- Los sistemas operativos proporcionan una colección de programas que permiten de forma más cómoda desarrollar y ejecutar programas:
  - Administración de archivos
  - Información de estado
  - Modificación de archivos
  - Soporte de lenguajes de programación
  - Carga y ejecución de programas
  - Comunicaciones
- Por debajo lanzan las llamadas al sistema necesarias para realizar la operación

Índice

Objetivos del diseño

- La estructura interna del sistema operativo varía mucho de un sistema a otro.
- Depende de la elección del hardware y del tipo de sistema.
- Objetivos:
  - Objetivos del usuario: cómodo de usar, fácil de aprender, fiable, seguro, rápido.
  - Objetivos del sistema: fácil de diseñar e implementar, flexible, fiable, libre de errores y eficiente.

Implementación

- Tradicionalmente la implementación se hacía en lenguaje ensamblador => poca portabilidad.
- Actualmente se escriben en lenguajes de programación cercanos a la arquitectura del ordenador como C o C++, aunque algunas partes es necesario escribirlas en ensamblador
  - Cambios de contexto: guardar/restaurar estado de los registros
  - *Drivers*
  - Etc…
- Utilizar estos lenguajes proporciona muchas ventajas: más rápido de escribir, más fácil de entender y depurar, cuando mejora el compilador se mejora todo el S.O

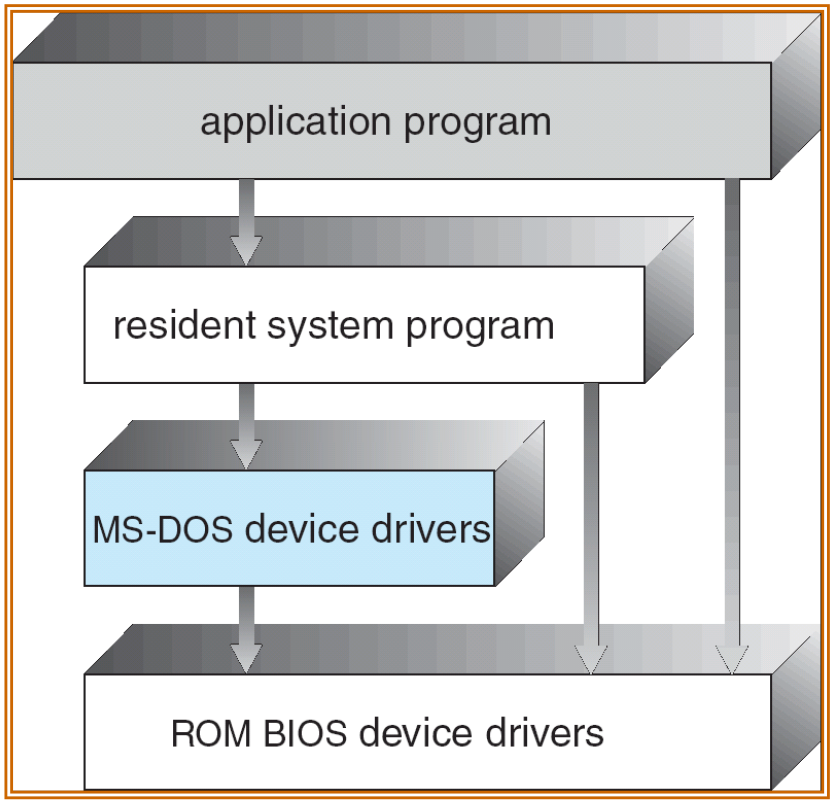
Índice

Estructura de un sistema operativo

- Existen distintas alternativas:
  - Sistemas monolíticos
  - Sistemas modulares
  - Sistemas por capas
  - Sistemas basados en micronúcleos
  - Sistemas híbridos
  - Exokernel
- Máquinas virtuales
  - Contenedores

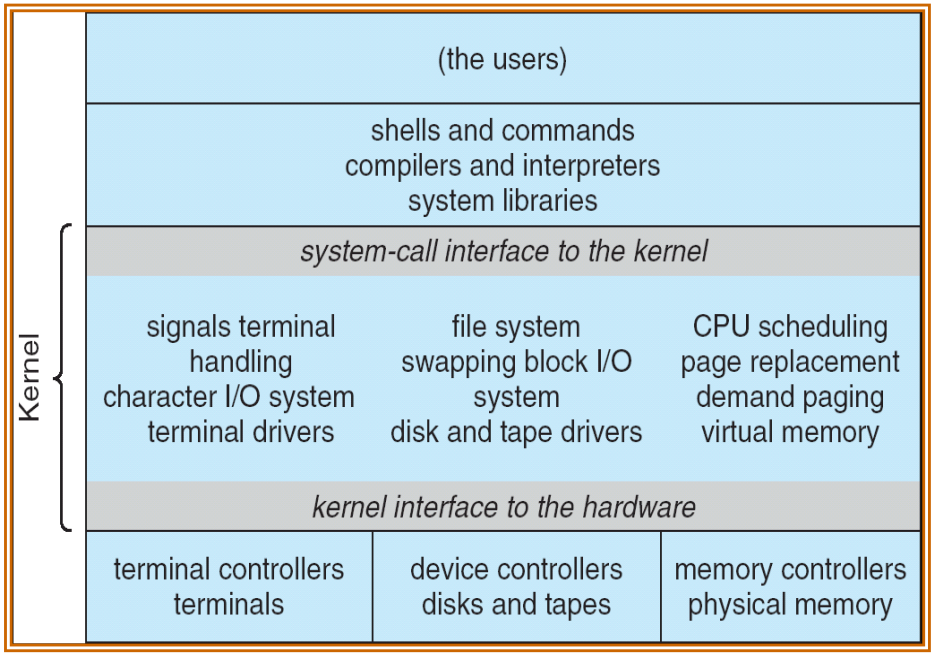
Estructura simple

- Inicialmente sin estructura definida
- Ejemplo: MS-DOS
  - Fue escrito para proporcionar la máxima funcionalidad en el menor espacio posible.
  - Aunque está algo estructurado, las interfaces y los niveles de funcionalidad no están bien separados.
    - Los programas pueden acceder directamente a las rutinas básicas de E/S
  - Fue escrito para el 8088 de Intel, que no tenía modo dual.



Monolíticos

- En el origen, debido a limitaciones del hardware, se tenía una estructura limitada.
- Dos partes separadas:
  - Programas del sistema.
  - K*ernel*:
    - TODO entre el hardware y el interfaz de llamadas al sistema.
    - Sistema de archivos, gestión de procesos, gestión de memoria, etc todo en una única capa.



Monolíticos

- Arquitectura más habitual. Característica de la familia UNIX (Linux)
- SO = núcleo (kernel) → programa que ejecuta en modo sistema
  - Todo el código del SO enlazado en un único programa que ejecuta en un mismo espacio de direcciones
- Aplicaciones y programas de sistema ejecutan en modo usuario
- Ventaja: Eficiencia
  - Ejecución de servicio de SO: Sólo requiere cambio de usuario a sistema y viceversa
- Desventaja: Difícil depuración y extensibilidad
  - Error en cualquier parte del SO afecta al resto

Módulos

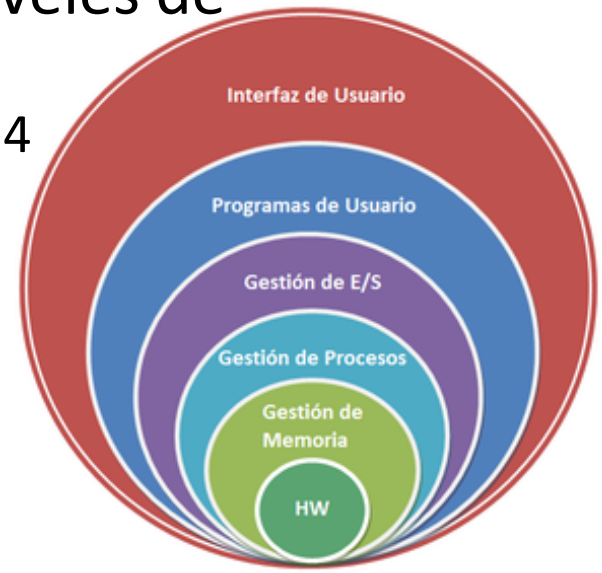
- Mayoría de sistemas monolíticos actuales no están “cerrados”
  - Usa las técnicas de programación orientada a objetos para crear un *kernel* modular.
  - Cada módulo tiene interfaces bien definidas para comunicarse con los otros módulos.
  - Permiten cargar módulos en tiempo de ejecución
- Ejecutable del SO contiene funcionalidad básica. La parte restante se encuentra en módulos (manejadores, s. ficheros, protocolos, etc.)
  - Similar a biblioteca dinámica pero para el núcleo. Se incorpora a espacio de SO y comparte símbolos
  - Se mantienen los mismos problemas de fiabilidad
- Ventajas:
  - Facilita extensibilidad del SO
  - Adaptación kernel a características de la plataforma.
    - Sistemas empotrados
  - Posibilita técnicas como hot-plugging

Módulos

	Sistema Monolítico	Sistema Modular
Arquitectura	Cerrada, es decir, inmodificable a los programadores ajenos a la compañía propietaria del código fuente.	Abierta. Todos los componentes del sistema del computador son compatibles con los de la compañía que lo haya producido.
Vulnerabilidad frente a ataques	Muy vulnerable	Vulnerabilidad controlada
Extensibilidad	No	Extensión de capacidades hacia operaciones con agentes externos mediante API's en un modelo cliente-servidor.
Cambios dinámicos en el software	No	Posibles, ya que cada módulo independientemente puede ser dado de alta o baja de manera independiente.

Sistemas por capas

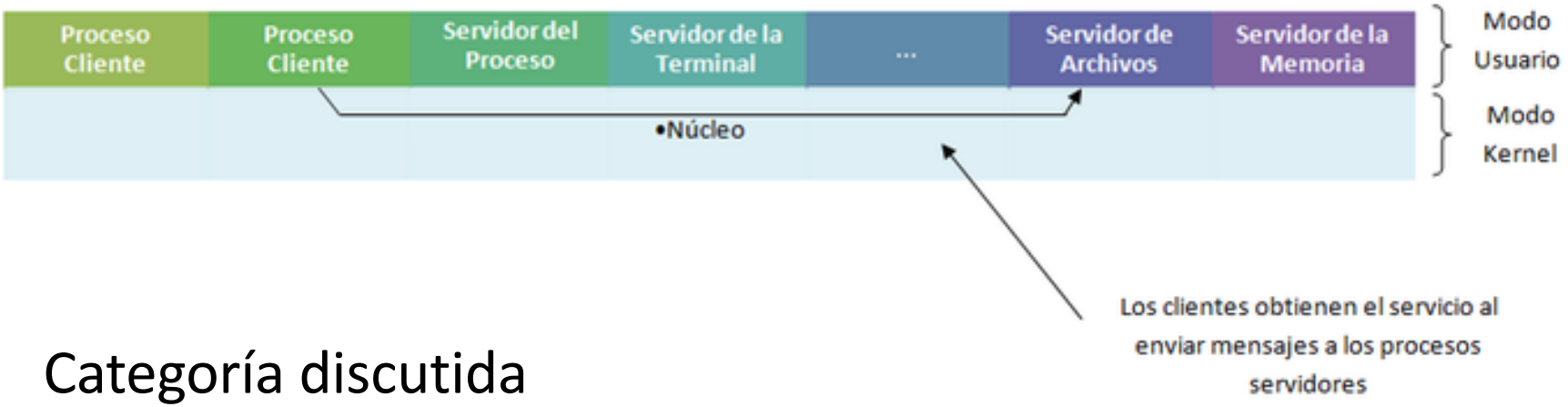
- Organizar funcionalidad del SO en capas independientes
  - Cada capa es un ejecutable independiente que ejecuta en su propio espacio de direcciones
  - Organizadas en niveles de mayor a menor privilegio
  - Capa sólo se comunica con adyacentes usando llam. al sistema
- Menos eficientes que otros tipos de implementaciones.
- Facilita depuración y controla propagación de errores
- Requiere que procesador provea de varios niveles de privilegio
  - Actualmente muchos procesadores proporcionan 4
  - S. monolíticos sólo requieren dos modos
    - Más transportables
- Primer SO por capas: THE (Dijkstra, 1968)
  - MULTICS también usó esta arquitectura



Microkernel

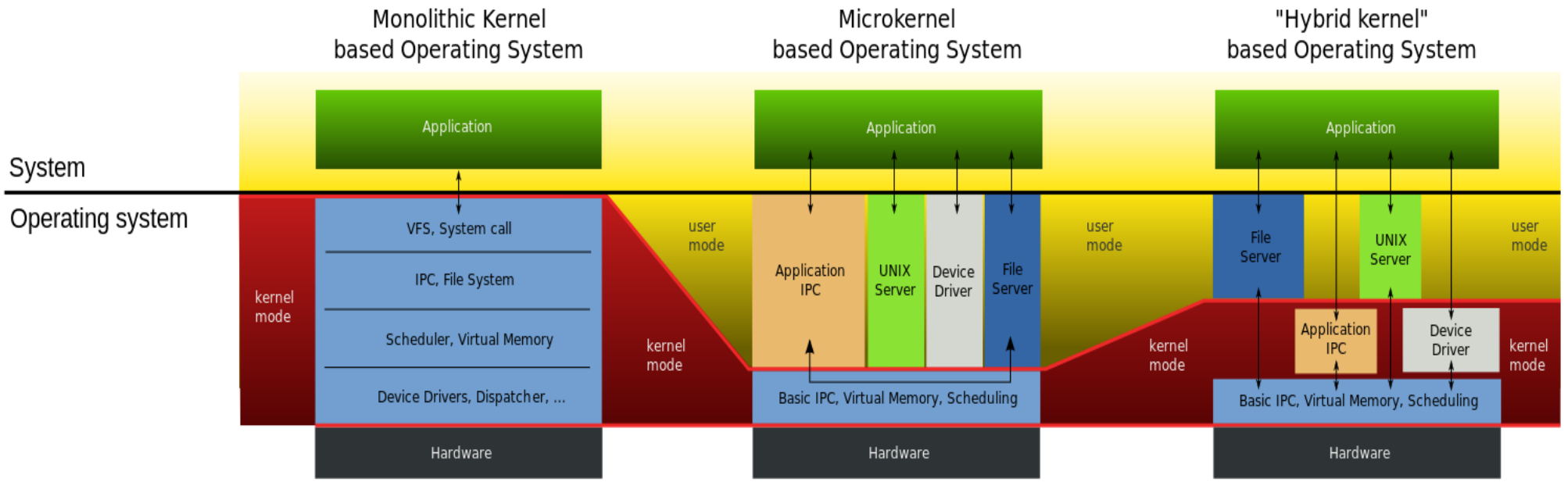
- Deja en el *kernel* lo mínimo imprescindible y todo lo demás se implementa como programas del sistema.
  - Gestión de procesos y de memoria de bajo nivel
  - Adicionalmente controla la comunicación, que se realiza mediante mensajes
- Funcionalidad del SO en servidores en modo usuario
  - Sistema de ficheros, gestor de memoria, manejadores, ...
  - "Llamada al sistema" de aplicación: que se realiza mediante mensajes, entre clientes y servidores o servidores y hardware.
- Ventaja: extensibilidad y facilidad de depuración
  - Más fácil de ampliar y de portar a nuevas arquitecturas.
  - Más confiable (hay menos código ejecutándose en modo kernel).
- Desventaja: Eficiencia. Coste de llamada:
  - Sobrecarga de mensajes y de cambios de proceso

Sistemas híbridos



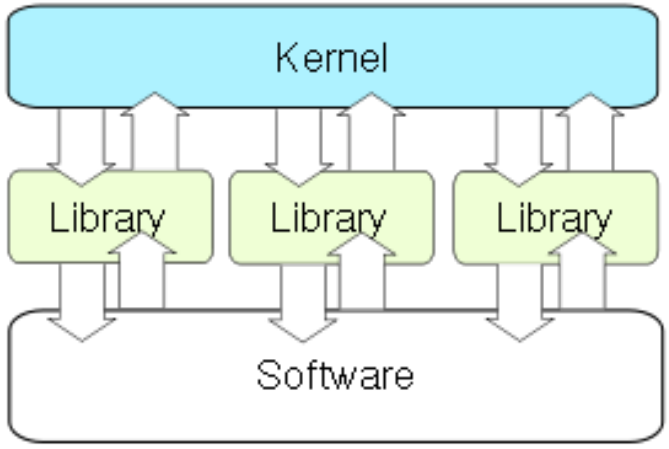
- Categoría discutida
  - Puristas consideran que son monolíticos
- Algunos microkernel permiten servidores en modo sistema
  - Más eficiente pero rompe la filosofía microkernel
  - Servidores son programas independientes pero ejecutan en mismo espacio de direcciones del microkernel
    - Y no usan IPCs para comunicarse
- Ejemplos:
  - macOS basado en el Microkernel Mach con servidores en m. sistema
  - Microsoft Windows

Sistemas híbridos



Exokernel

- Motivación:
  - No todas las aplicaciones necesitan ver las mismas abstracciones
  - Uso de abstracciones inadecuadas es ineficiente
  - Ej: Gestor base de datos maneja mejor bloques de disco que ficheros.
- Propuesta: Exokernel (MIT, 1995)
  - Kernel provee abstracciones básicas (página, bloque, ...)
  - Funcionalidades de tipo SO en bibliotecas en modo usuario
  - Cada aplicación se enlaza con las bibliotecas que requiera
    - Gestor base de datos no requiere de sistema de ficheros
- Todavía en investigación. No usado en ningún SO comercial

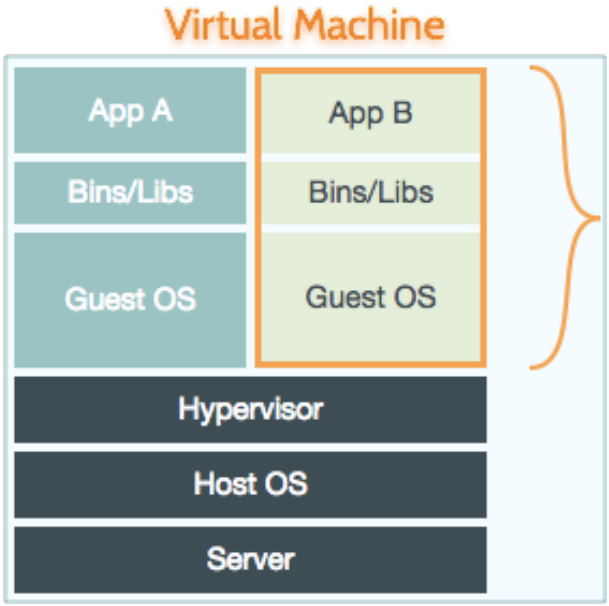


Máquinas virtuales (MV)

- Software que implementa una máquina (= o ≠ máquina real)
  - SO crea máquina virtual pero extendida (abstracción del HW)
- Técnica de nuevo en auge en los últimos años
  - Capacidad de procesamiento actual palia ineficiencia de MV
  - Procesadores incluyen soporte para la misma
- Tipos de MV
  - De sistema: Crea MV como soporte de ejecución de un SO
  - De proceso: Crea MV como soporte de ejecución de 1 proceso

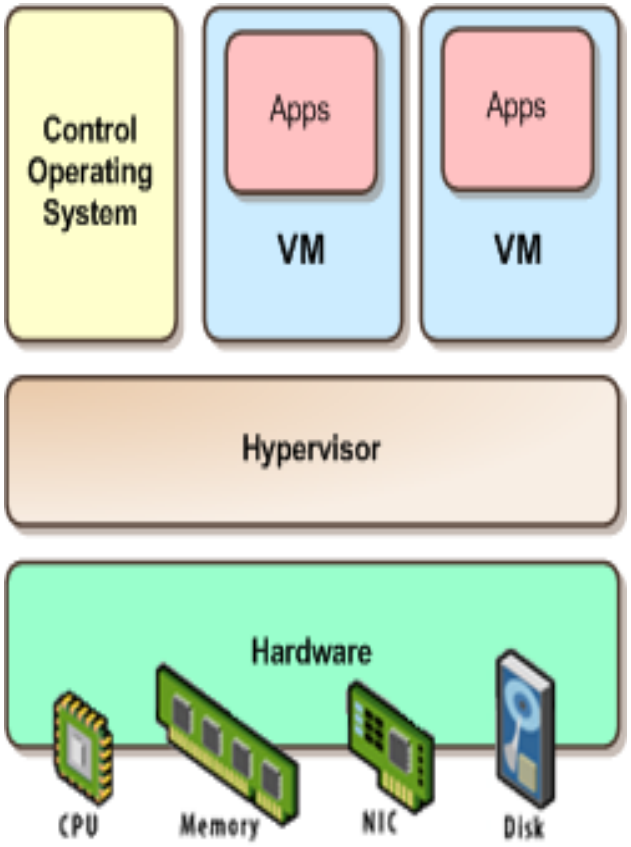
Máquinas virtuales (MV)

- La virtualización se permite gracias al **hypervisor**, también llamado gestor de máquinas virtuales, *virtual machine manager* (VMM).
- Un **hypervisor** es un software (apoyado por el soporte hardware) que permite la ejecución de múltiples sistemas operativos invitados sobre el sistema operativo anfitrión
  - Tipo 1 (nativo, *bare metal*): Directamente sobre el hardware.
    - Ej: Amazon EC2 (Xen), VMWare ESX/ESXi
  - Tipo 2 (hosted): Aplicación que ejecuta dentro del SO.
    - Ej: VirtualBox, VMWare WorkStation, Parallels

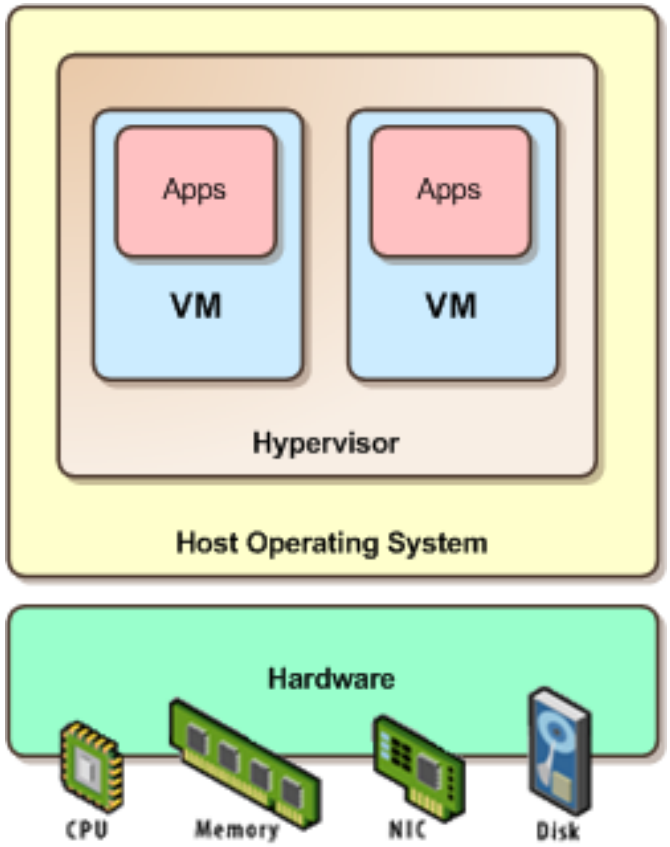


Hipervisor/VMM

Tipo 1 (nativo)



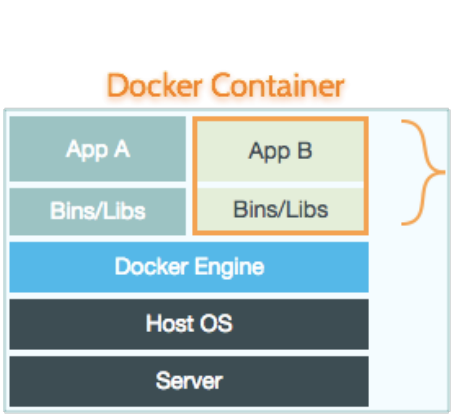
Tipo 2 (hosted)

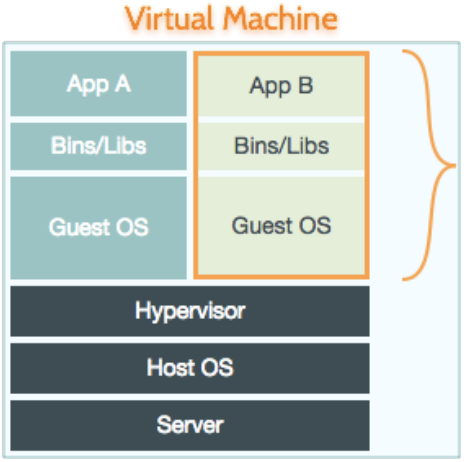


- Los contenedores son una tecnología que ofrece unas ventajas similares a las VMs pero aprovechando mejor los recursos:
  - Los contenedores tardan milisegundos en arrancar
  - Consumen únicamente la memoria que necesita la app ejecutada en el contenedor. Una VMs reserva la memoria completa

Contenedores

- Un contenedor es un **paquete** que contiene una aplicación y todo el sw necesario para que se ejecute (python, gcc, libs,...)
- Para ejecutar un contenedor no se necesita **hypervisor** porque no se ejecuta un sistema operativo invitado y no hay que simular HW
- El contenedor es ejecutado directamente por el **kernel** como si fuera una aplicación más pero de forma **aislada del resto**
  - Requiere que la aplicación este realizada para el sistema operativo subyacente





VMs	Contenedores
Más pesadas	Más ligeras
Varios procesos	Un único proceso
Conexión por red (aunque esté en local)	Acceso directo al contenedor
Más seguridad porque están más aisladas del host	Potencialmente menor seguridad porque se ejecutan como procesos en el host

Diferencias en despliegue de software

	Ships within ...	Manual deployment takes ...	Automated deployment takes ...	Boots in ...
Bare Metal	days	hours	minutes	minutes
Virtualization	minutes	minutes	seconds	less than a minute
Lightweight Virtualization	seconds	minutes	seconds	seconds