

1

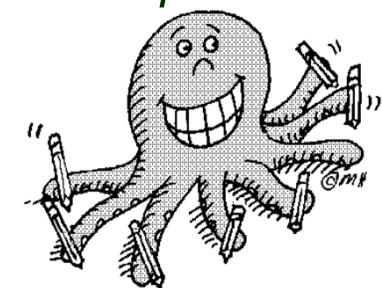
Introducción

- **Sistemas Concurrentes**
- **Breve historia de los SOs**
- **Arquitecturas de SOs**



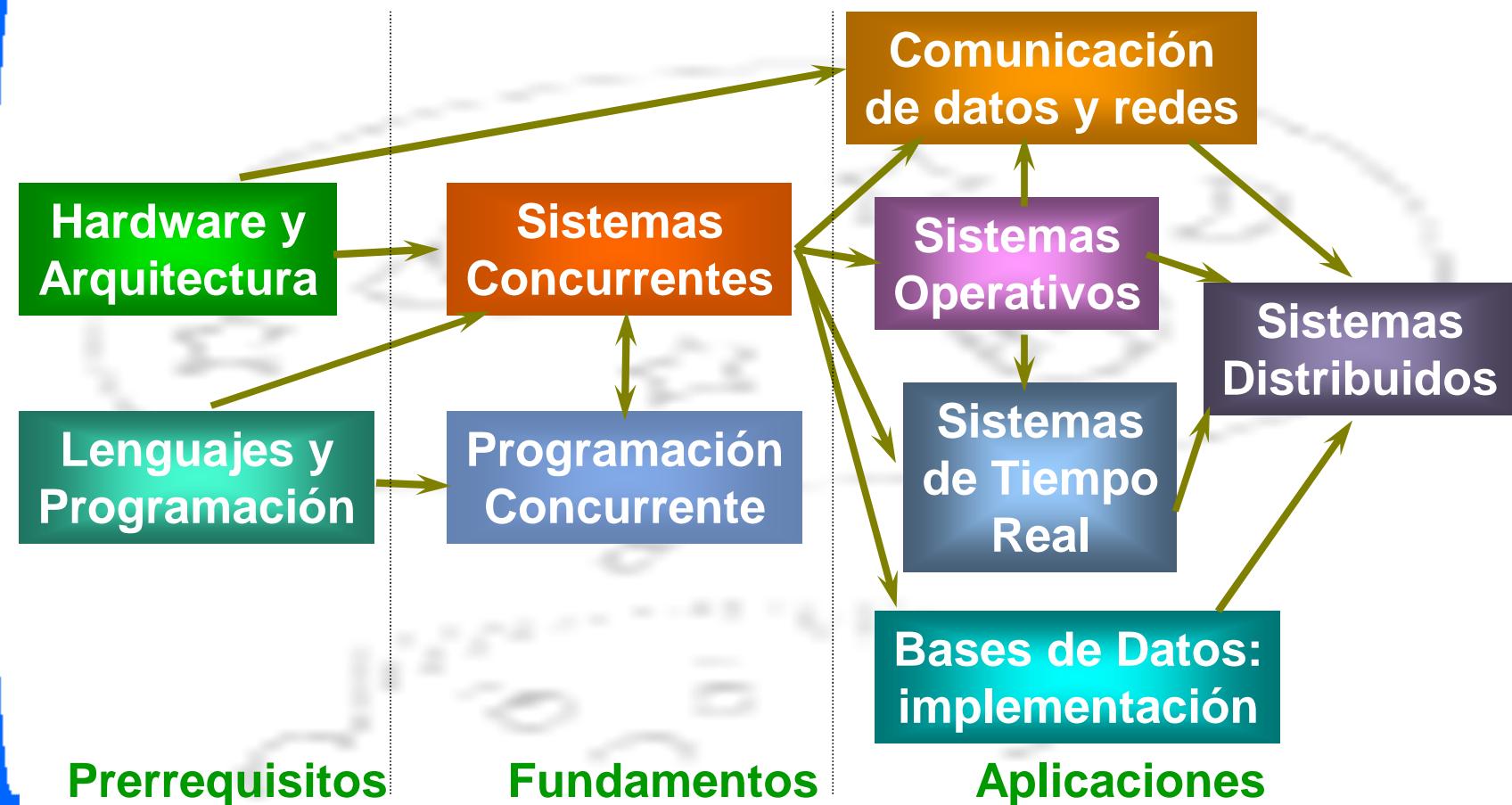
Sistemas Concurrentes

- **Sistema Concurrente** es aquel sistema software en cual existen diversas actividades separadas en progreso en el mismo instante.
- Algunos sistemas concurrentes: sistemas operativos, entornos de ventanas, sistemas de control de tráfico aéreo, etc.
- La Vida esta repleta de ellos: cocinar, conducir, nosotros mismos, etc.



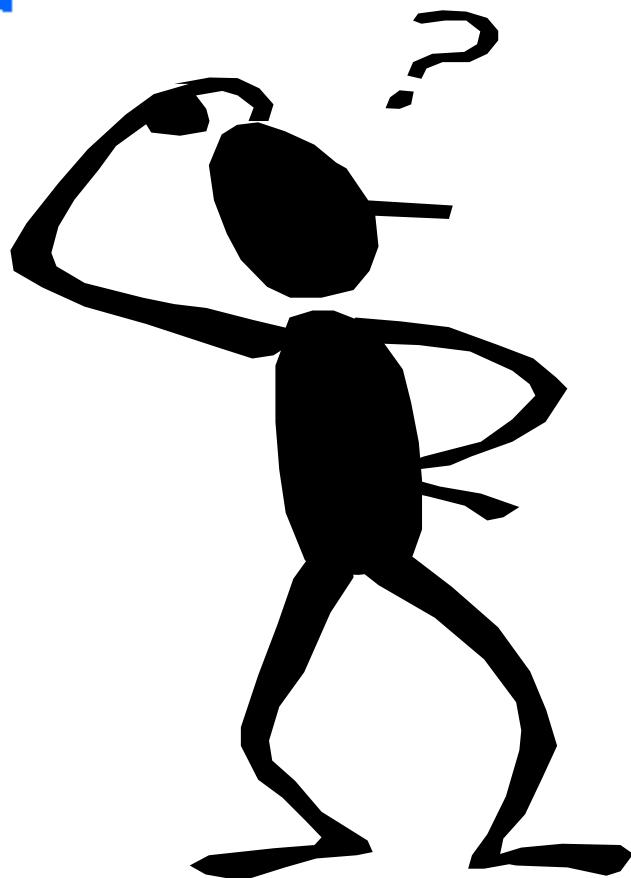


Sistemas Concurrentes





Breve historia de los SOs



Veremos SOs:

- por lotes
- multiprogramados
- de tiempo compartido
- de tiempo real
- distribuidos
- paralelos
- de Internet



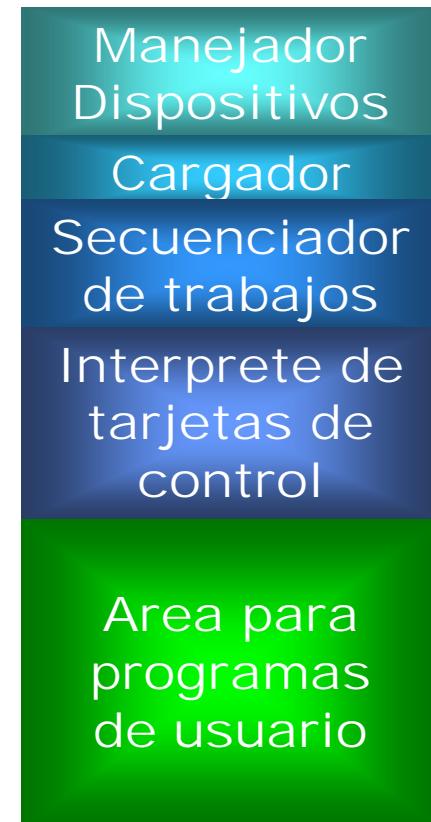
Breve historia de los SO's

- En un principio, el SO era sólo un fragmento de código que se enlazaba con los programas, se cargaba todo en memoria, y se ejecutaba con el programa - como una biblioteca en tiempo de ejecución.
- Problema: uso ineficiente de recursos caros (baja utilización de la CPU) ya que el tiempo de preparación de una tarea era significativo.



Los Sistemas por Lotes

- Los **sistemas batch** fueron los primeros SO's reales:
 - El monitor estaba almacenado en memoria.
 - Cargaba un trabajo en memoria (tarjetas).
 - Ejecutaba el trabajo.
 - Cargaba el siguiente.
 - Las tarjetas de control en el archivo de entrada indicaban que hacer.





Problemas (sists. Batch)

- El problema principal se debía a las largas esperas entre lotes de trabajos.
- La 2^a generación de ordenadores introduce:
 - Hardware separado para gestionar las E/S
 - Aparece del concepto interrupción
 - ⇒ un programa puede seguir ejecutándose mientras se está realizando una E/S.
- Nueva dificultad: ¿cómo gestionar la concurrencia? los SOs multiprogramados.

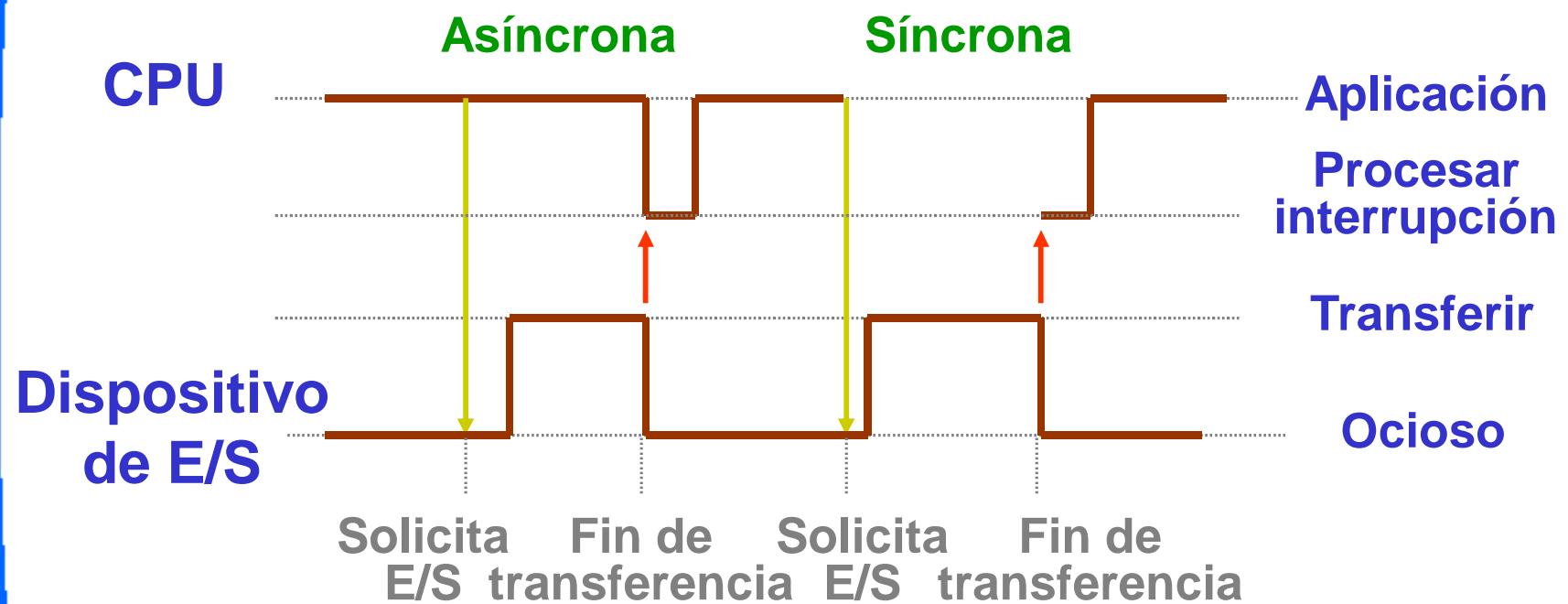


La Multiprogramación

- Los **sistemas multiprogramados** optimizan la productividad (*throughput*) del sistema:
 - Se benefician de que los dispositivos de E/S pueden operar asíncronamente.
 - Mantienen varios trabajos ejecutables cargados en memoria.
 - Solapan procesamiento de E/S de un trabajo con cómputo de otro.
 - Necesitan de interrupciones, o DMA.



Multiprogramación





Soporte para la multiprogramación

- El SO debe suministrar las rutinas de E/S.
- Debe existir una gestión de memoria para poder asignar y controlar la memoria repartida entre varios trabajos.
- Debe existir una planificación de la CPU: el SO elige uno de los trabajos listos para ejecutarse que hay en memoria.
- El SO realiza la asignación de los dispositivos a los trabajos.



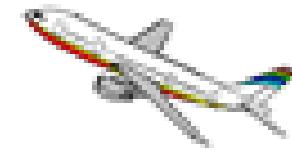
Sistemas de Tiempo Compartido

- Soportan el uso “interactivo” del sistema:
 - Cada usuario tiene la ilusión de disponer de la máquina completa.
 - Tratan de optimizar el tiempo de respuesta.
 - Basados en asignar fracciones de tiempo - se reparte el tiempo de CPU de forma equitativa entre los procesos.
 - Permiten la participación activa de los usuarios en la edición, depuración de programas, y ejecución de los procesos.





SOs de Tiempo-Real

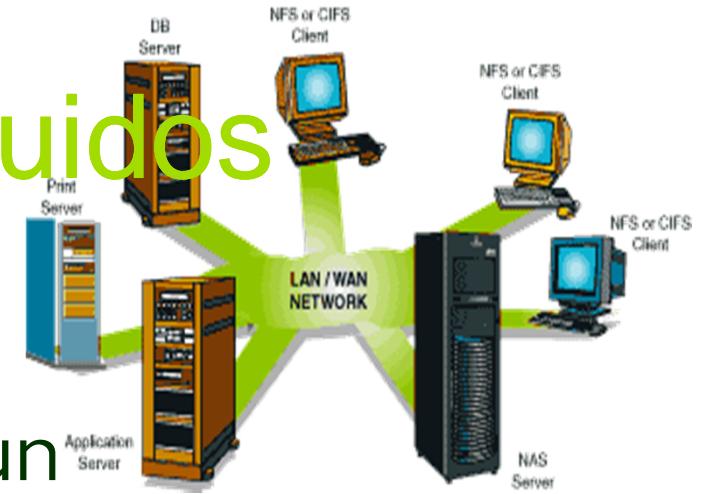


- Los STRs se suelen usar en aplicaciones especializadas, p.ej. sistemas de control, ...
 - 💡 Garantizar la respuesta a sucesos físicos en intervalos de tiempo fijos.
 - 💻 Problema: poder ejecutar las actividades del sistema con el fin de satisfacer todos los requisitos críticos de las mismas.
 - Con el uso de aplicaciones multimedia sobre PC's, todos los SOs tienen requisitos de tiempo-real.





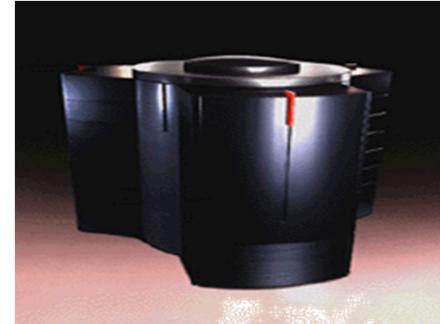
Sistemas Distribuidos



- Un **sistema distribuido** es un sistemas multicomputador pero que no tiene una memoria común, ni reloj.
- El objetivo básico es compartir recursos distribuidos: hardware (impresoras, discos, ...) o software (correo, IRC, Web, etc).
- Permiten un aumento de la fiabilidad del sistema. Si una parte falla, el resto de sistema puede seguir su ejecución, al menos, parcialmente.



SOs Paralelos



- SOs para **sistemas multiprocesadores** – sistemas de computador con varios procesadores que comparten una única memoria y un reloj.
- Estos sistemas sustentan aplicaciones paralelas cuyo objetivo es obtener aumento de velocidad de tareas computacionalmente complejas, p. ej. predicción meteorológica, simulaciones de explosiones atómicas, etc.
- **Multiprocesamiento Simétrico (SMP)** - todos los procesadores pueden ejecutar código del SO y de las aplicaciones.



SOs de Internet



- Algunos vendedores mantienen que el PC es más complicado y caro de lo necesario: tiene el hardware/software necesario para mantener gran funcionalidad que nunca se utiliza.
- Proponen el **computador de red** que suministra la funcionalidad mínima para ejecutar un navegador de Internet, el cual cargará los *applets* que suministran la funcionalidad necesaria (SO y aplicaciones).



Estructuras de SOs

- Servicios (componentes) de un SOs
- Propiedades de una buena estructura
- Arquitecturas de SOs:
 - Monolítica
 - Capas
 - Máquina Virtual
 - Microkernel



Componentes de un SO

- Veremos los componentes de un SO y cómo estos se organizan.
- ¡Ten presente a partir de ahora que en los SOs, como en la vida real, nada es tan simple como parece! El diseño y la implementación del sistema no es tan clara en el mundo real como en los modelos.
- Los conceptos que veremos están presentes en todos los SOs, sin embargo, cada sistema los puede implementar de forma diferente.

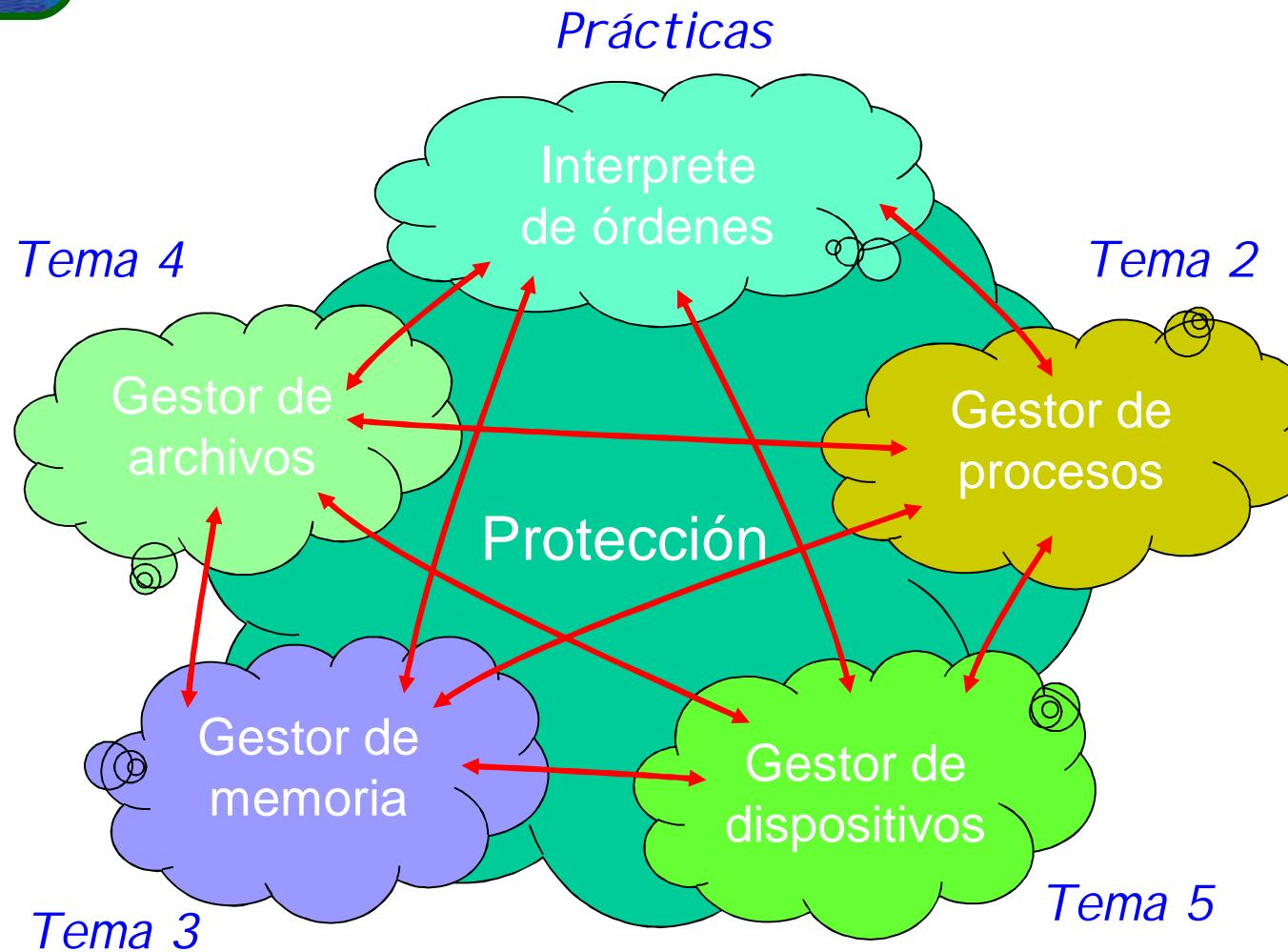


Servicios del SO

- Algunos de los servicios que suministra el SO a los procesos de aplicación:
 - Ejecutar diferentes actividades
 - Acceso a información permanente
 - Acceso a los dispositivos
 - Suministrar memoria para ejecutarlos
 - El uso de los recursos debe ser seguro
 - Interfaz de acceso a los servicios del SO
- Estos servicios se suelen integrar en componentes.



Componentes Unix





Gestión de procesos

- Podemos definir **proceso** como una instancia de un programa en ejecución.
- El SO es responsable de las siguientes actividades relacionadas con los procesos:
 - crear/destruir procesos,
 - suspender/reanudar procesos,
 - suministrar los mecanismos para sincronizar y comunicar procesos.



Gestión de memoria principal

- La memoria principal es el almacenamiento de acceso directo para la CPU y los dispositivos de E/S. Es volátil.
- El SO es responsable de:
 - asignar/desasignar memoria a los programas.
 - mantener la pista de la memoria utilizada actualmente y quién la usa.
 - decidir cuanta memoria asignar a cada proceso, y cuando debe ser retirado de memoria un proceso.



Gestión de archivos

- Un **archivo** es una colección de información con nombre. Es la entidad básica de almacenamiento persistente.
- El **sistema de archivos** suministra:
 - primitivas para manipular archivos y directorios: crear, borrar, leer, escribir, renombrar, ...
 - correspondencia entre archivos y su almacenamiento secundario.
- También, suministra servicios generales: backup, contabilidad y cuotas, etc.



Gestión de Entradas/Salidas

- Los SOs ofrecen a los programas una interfaz estándar de los dispositivos, es decir, utilizan las mismas funciones independientemente del dispositivo al que accedan.
- Un **manejador de dispositivo** es un módulo que gestiona un tipo de dispositivo. El encapsula el conocimiento específico del dispositivo, p.ej., inicialización, interrupciones, lectura/escritura, etc.



Sistema de protección

- Protección referencia al mecanismo para controlar los accesos de los programas a los recursos del sistema.
- El mecanismo de protección debe:
 - distinguir entre uso autorizado o no,
 - especificar que control se debe imponer, y
 - suministrar los medios para su aplicación.
- La protección suele ser un mecanismo general a todo el SO, es decir, no está localizada en un único módulo.



Intérprete de órdenes

- Programa o proceso que maneja la interpretación de órdenes del usuario desde un terminal, o archivo de órdenes, para acceder a los servicios del SO.
- Puede ser una parte estándar del SO, p. ej. el *command.com* de MS-DOS.
- O bien, un proceso no privilegiado que hace de interfaz con el usuario. Esto permite la sustitución de un interprete por otro. P. ej. en Unix tenemos *csh*, *bash*, *ksh*, etc.



La gran cuestión

- Un SO consta de los elementos vistos anteriormente, entre los cuales existen muchas y complejas relaciones.
- Las preguntas importantes son:
 - ¿Cómo se organiza todo esto?
 - ¿Cuales son los procesos y dónde están?
 - ¿Cómo cooperan esos procesos?
- Es decir, **¿cómo construir un sistema complejo que sea eficiente, fiable y extensible?**



Características de un “buen” SO

- **Eficiente** - debe ser lo más eficiente posible, pues consume muchos ciclos de CPU.
- **Fiable** - Fiabilidad se refiere a dos aspectos diferentes pero relacionados:
 - Robustez, el SO debe responder de forma predecible a condiciones de error, incluidos fallos hardware.
 - El SO debe protegerse activamente a sí mismo y a los usuarios de acciones accidentales o malintencionadas.



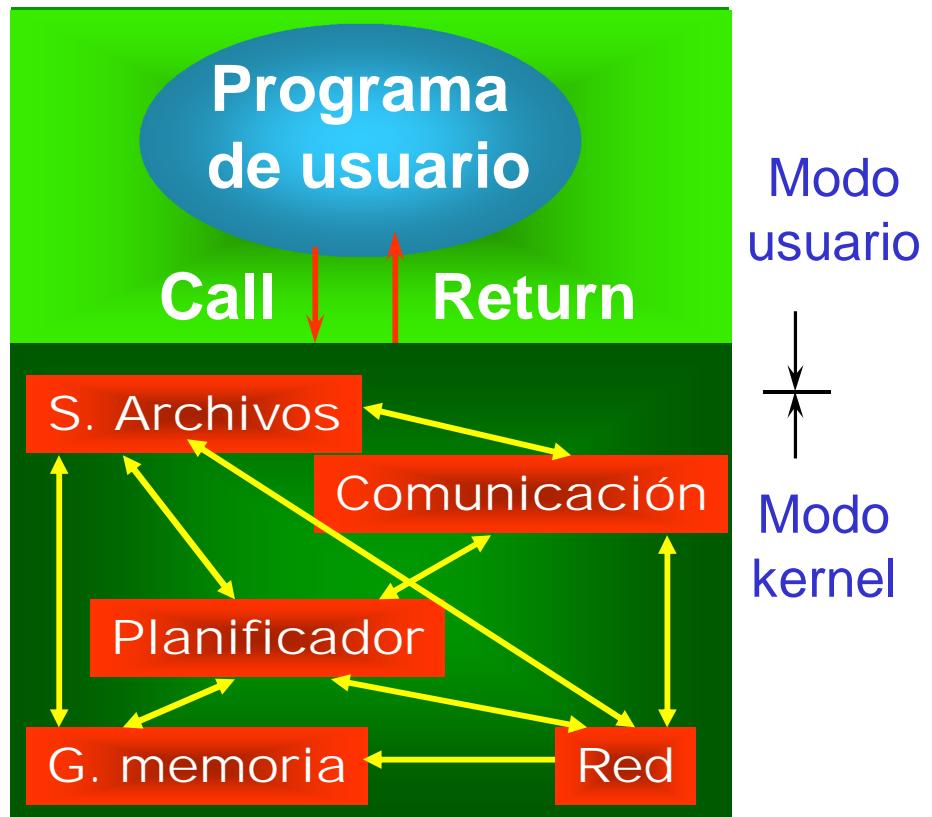
Características de un buen SO (ii)

- **Extensibilidad** - la aparición constante de nuevo hardware y de nuevos tipos de aplicaciones, exigen al SO la adición de nueva funcionalidad. En lugar de construir un nuevo SO cada vez, se pretende construir un sistema que pueda extenderse, es decir, su funcionalidad pueda variar o crecer, de una forma sencilla. Por ejemplo, que un sistema tradicional soporte aplicaciones de tiempo real, que soporte nuevos sistemas de archivos, etc.



Estructura Monolítica

- Los componentes del SO se ejecutan en modo kernel; la relación entre ellos es compleja.
- El modelo de obtención de servicios es la **llamada a procedimiento**.





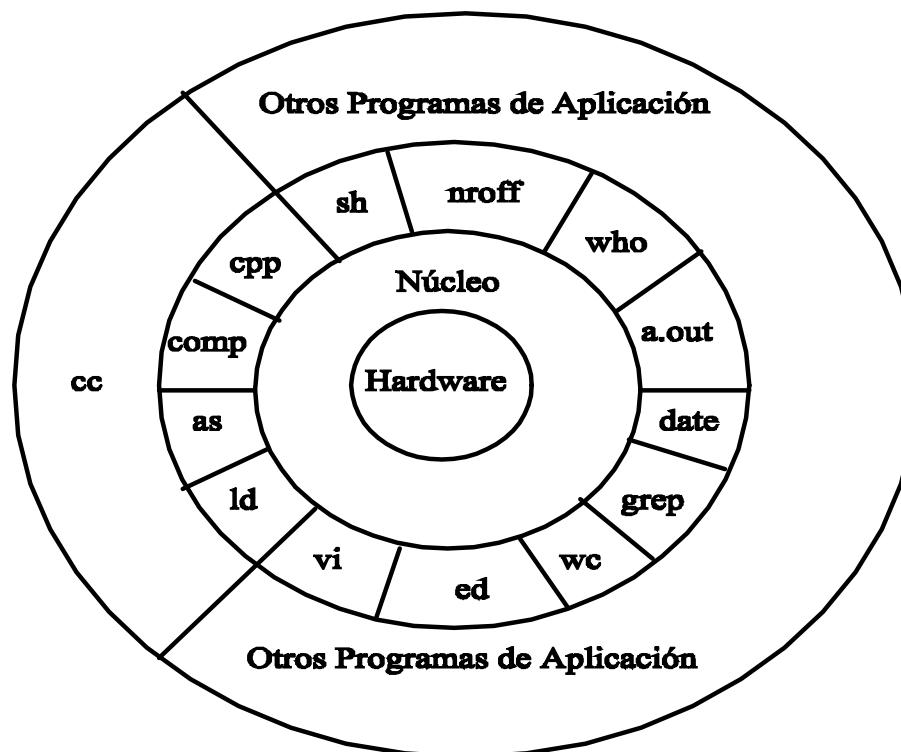
Problemas de los sists. monolíticos

- Son difíciles de comprender, ya que son un único programa (cientos de MB), por tanto, difíciles de modificar y mantener.
- No fiables: un fallo de algún módulo puede provocar la “caída” del sistema.
P.ej, Pantalla azul de Windows o *panic* de Unix.
- Por esto, los diseñadores buscan mejores formas de estructurar un SO para simplificar su diseño, construcción, depuración, ampliación y mejora de sus funciones.



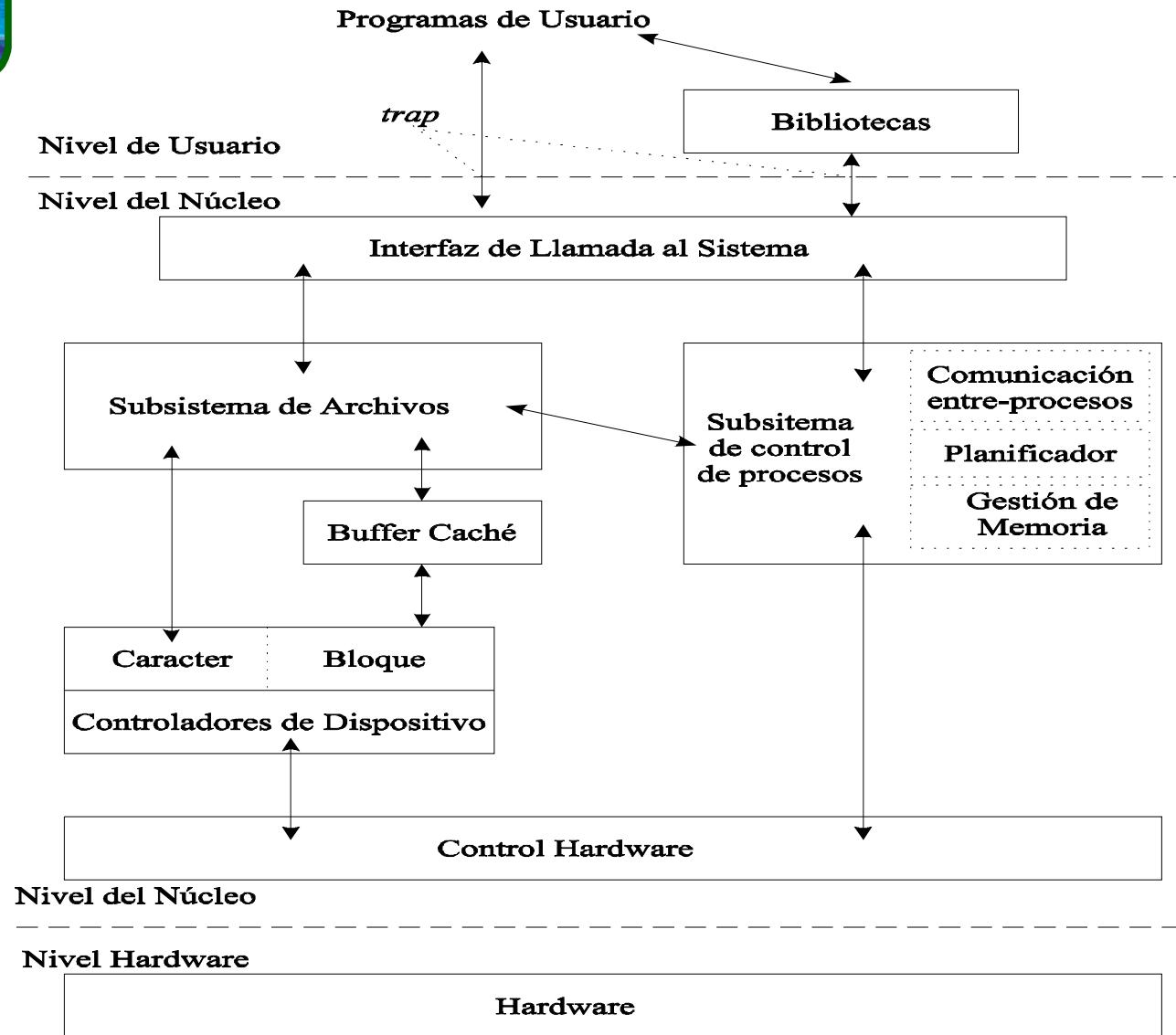
Estructura Sistema Unix

El Sistema Operativo interacciona directamente con el hardware, proporcionando servicios comunes a los programas





Arquitectura Unix

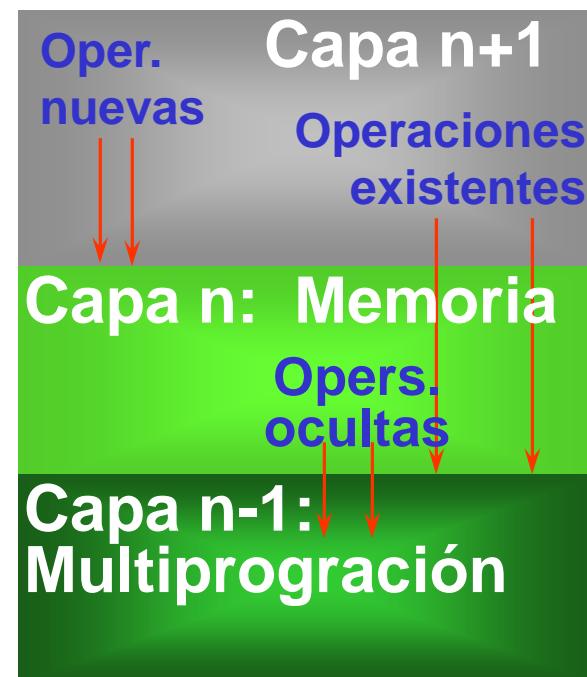




Estructura de Capas

- Implementado como una serie de capas; cada una es una máquina más abstracta para la capa superior.
- Por modularidad, las capas se seleccionan para que cada una utilice sólo funciones de las capas inferiores.

Capa en desarrollo





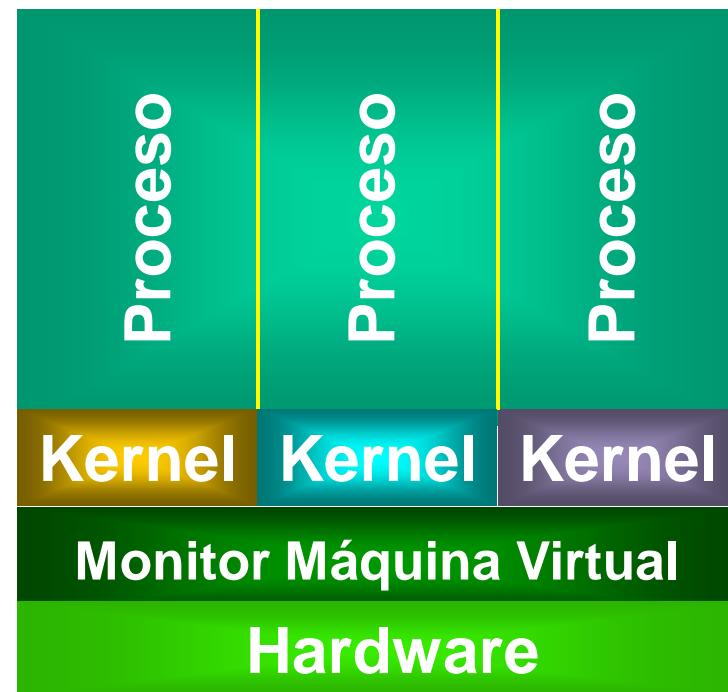
Problemas de la estructura de capas

- Los sistemas de capas son jerárquicos pero los reales son más complejos, p. ej.,
 - El sistema de archivos podría ser un proceso en la capa de memoria virtual.
 - La capa de memoria virtual podría usar archivos como almacén de apoyo de E/S.
- Existe sobrecarga de comunicaciones entre las distintas capas.
- A menudo, los sistemas se modelan con esta estructura pero no se construyen así.



Máquina Virtual

- Sigue el enfoque de capas para su conclusión lógica.
- La Máquina Virtual crea múltiples réplicas idénticas del hardware.
- El SO crea la ilusión de múltiples procesos, cada uno ejecutándose en su propia CPU con su propia memoria.





Características deseables de la M.V.

- ✓ El aislamiento de cada máquina virtual asegura la protección de los recursos.
- ✓ Sirve investigar/desarrollar SO's: no interrumpe el funcionamiento del sistema, y permite usar sus herramientas (editor, compilador, etc.)
- ✓ Permite la ejecución de aplicaciones realizadas para otro SO, p. ej. ventana MS-DOS de Windows 9x.



Características no deseables de la M.V.

- ✖ Dado el aislamiento de cada máquina virtual, la compartición de recursos no es fácil.
- ✖ Son difíciles de implementar perfectamente debido a la complejidad de crear un duplicado exacto del hardware.
P. Ej. Los bits de la palabra de estado en un procesador Intel tienen diferente significado según estemos en modo real o protegido.



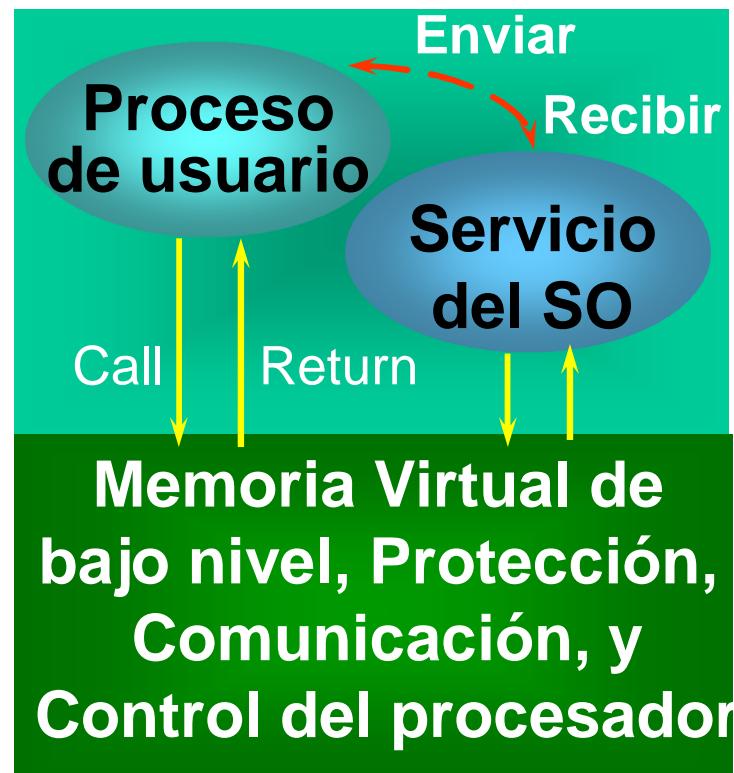
Algunos conceptos

- El **espacio de direcciones** de un proceso son aquellas direcciones de memoria que el proceso puede direccionar. (Tema 3)
- El SO confina la ejecución de un proceso a su espacio de direcciones.
- El SO es parte del espacio de direcciones de todos los procesos.
- Una llamada a un procedimiento es posible si la dirección del procedimiento está en el espacio de direcciones del llamador.



Arquitectura Microkernel

- Algunas de las funciones del SO se implementan como procesos de usuario.
- El (micro)kernel sólo necesitaría contener la funcionalidad básica para crear y comunicar procesos.





Ventajas del μkernel

- Más fiable – un posible error de un servicio del SO queda confinado en el espacio de direcciones del proceso que lo implementa.
- Es extensible y *personalizable* – podemos cambiar un servicio del SO, cambiando el proceso que lo implementa. Podemos ejecutar programas realizados para otro SO distinto.

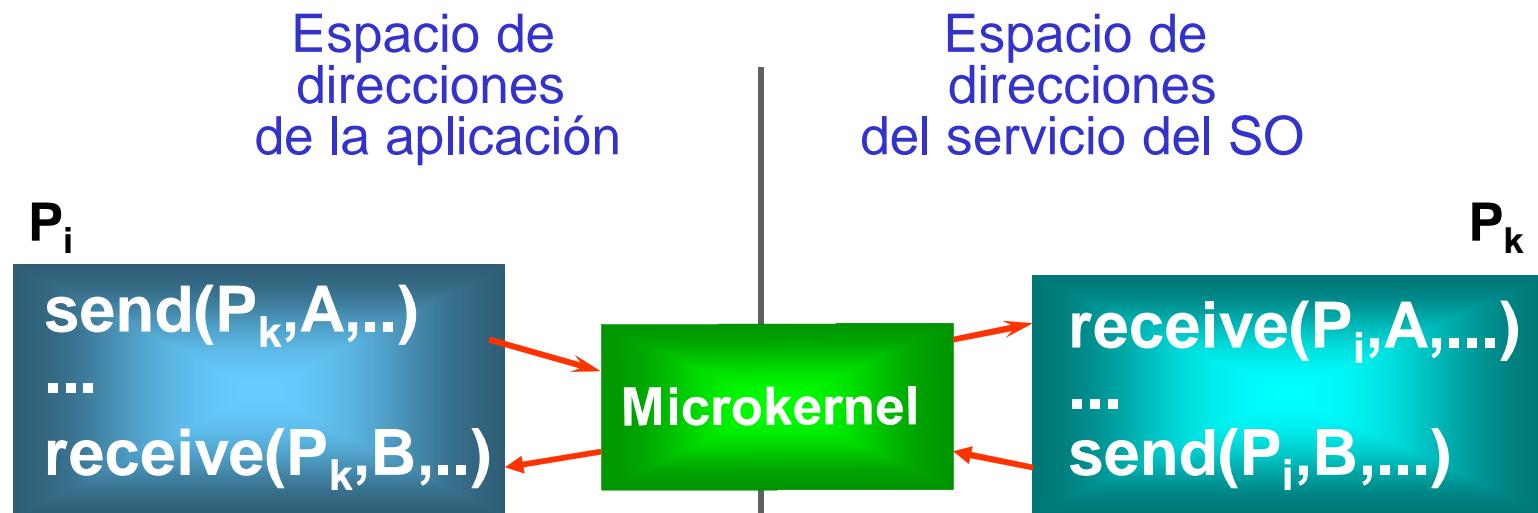


Modelo de comunicación

- Se obtiene un servicio del SO mediante un **paso de mensajes**:
 1. La aplicación envía el mensaje de solicitud, **send(P,A)**, y espera la respuesta con **receive(P,b)**.
 2. El kernel verifica el mensaje y lo entrega al servidor.
 3. El servidor que esta en espera, realiza el servicio solicitado y devuelve el resultado.



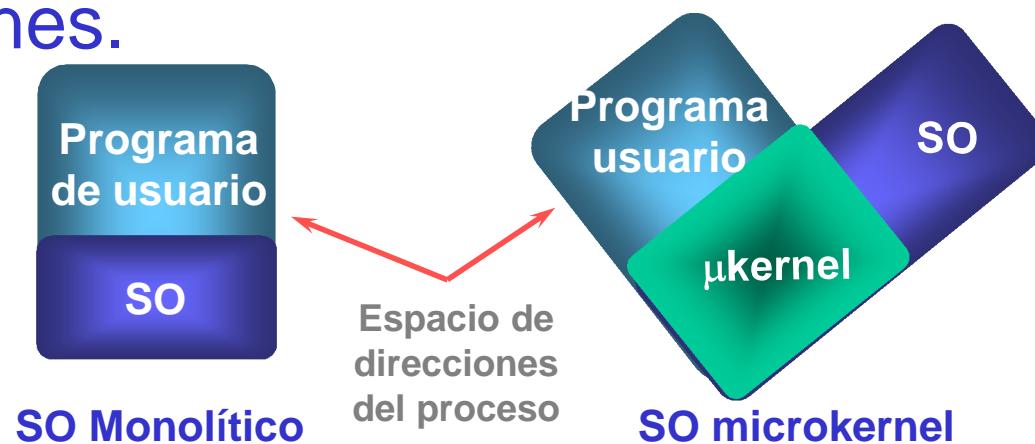
Esquema





μkernel frente a monolítica

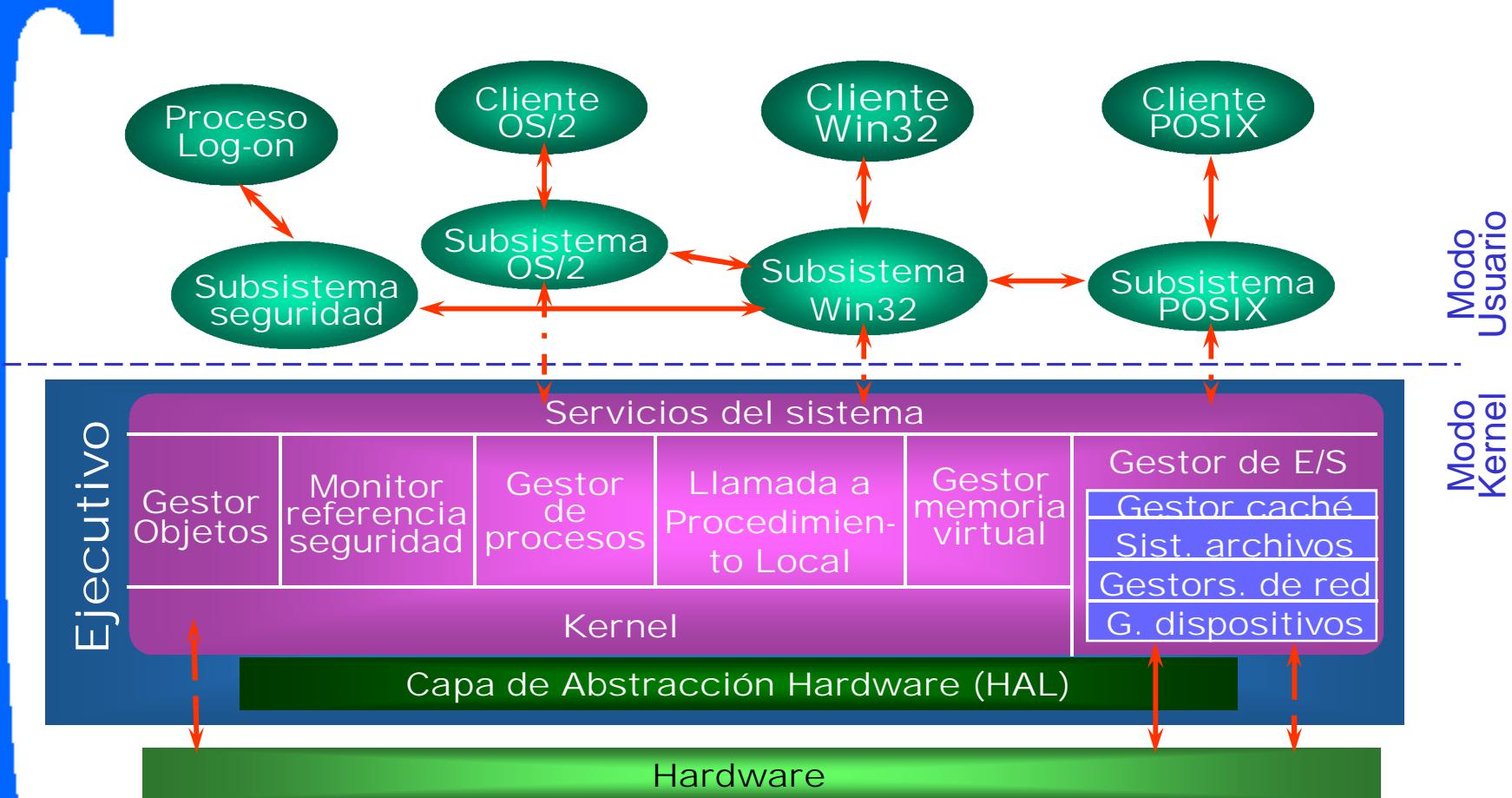
- ✗ La estructura μkernel tiene peor rendimiento que la monolítica; para obtener un servicio se realizan más cambios de modo y espacios de direcciones.

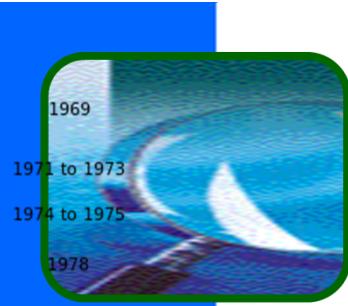


- ✓ La estructura μkernel es más flexible que la monolítica.

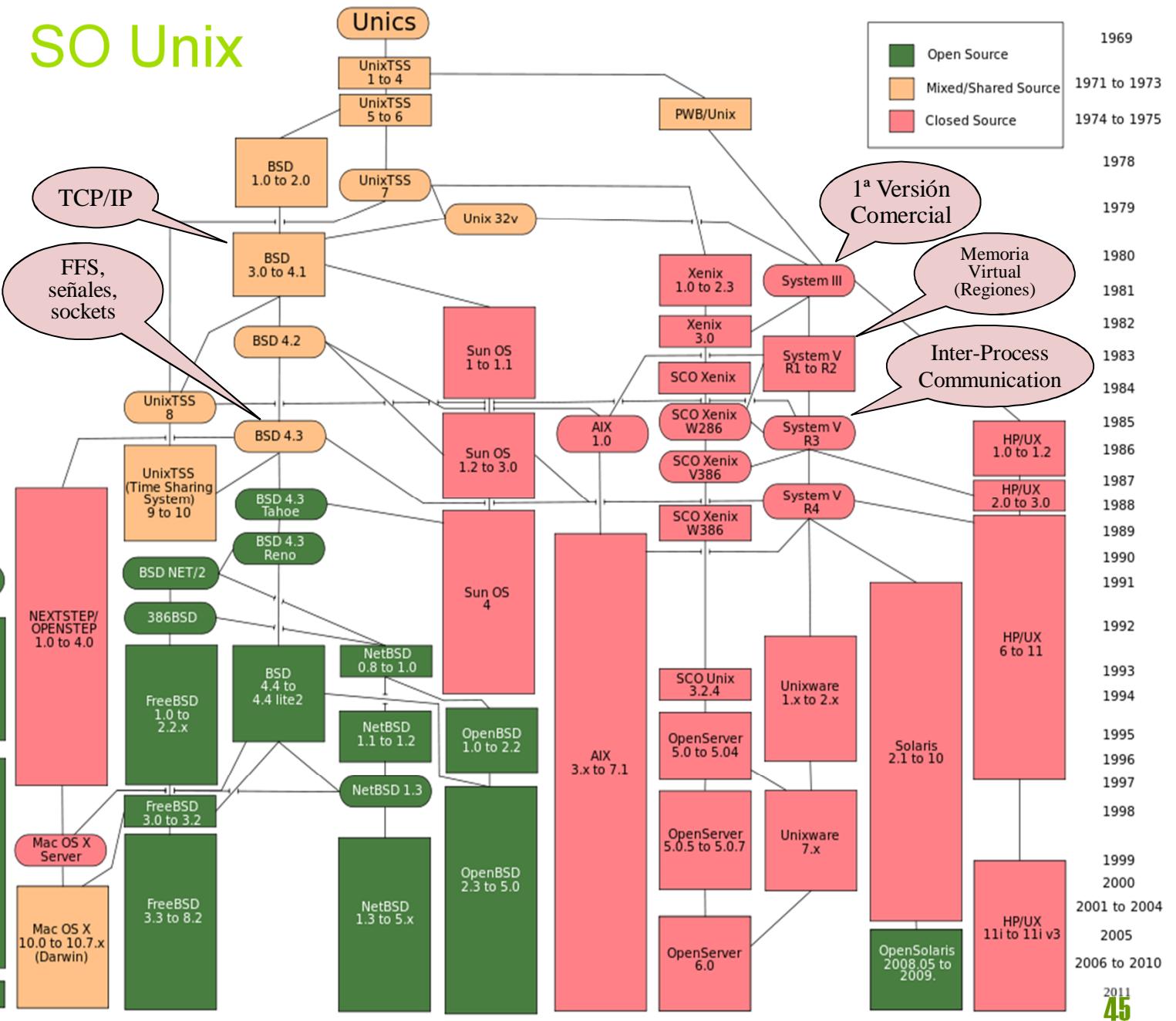


Arquitectura Windows





Evolución SO Unix



1

Introducción

- **Sistemas Concurrentes**
- **Breve historia de los SOs**
- **Arquitecturas de SOs**