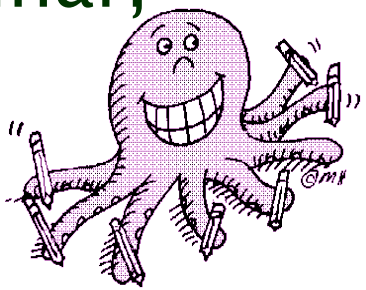
1

Introducción

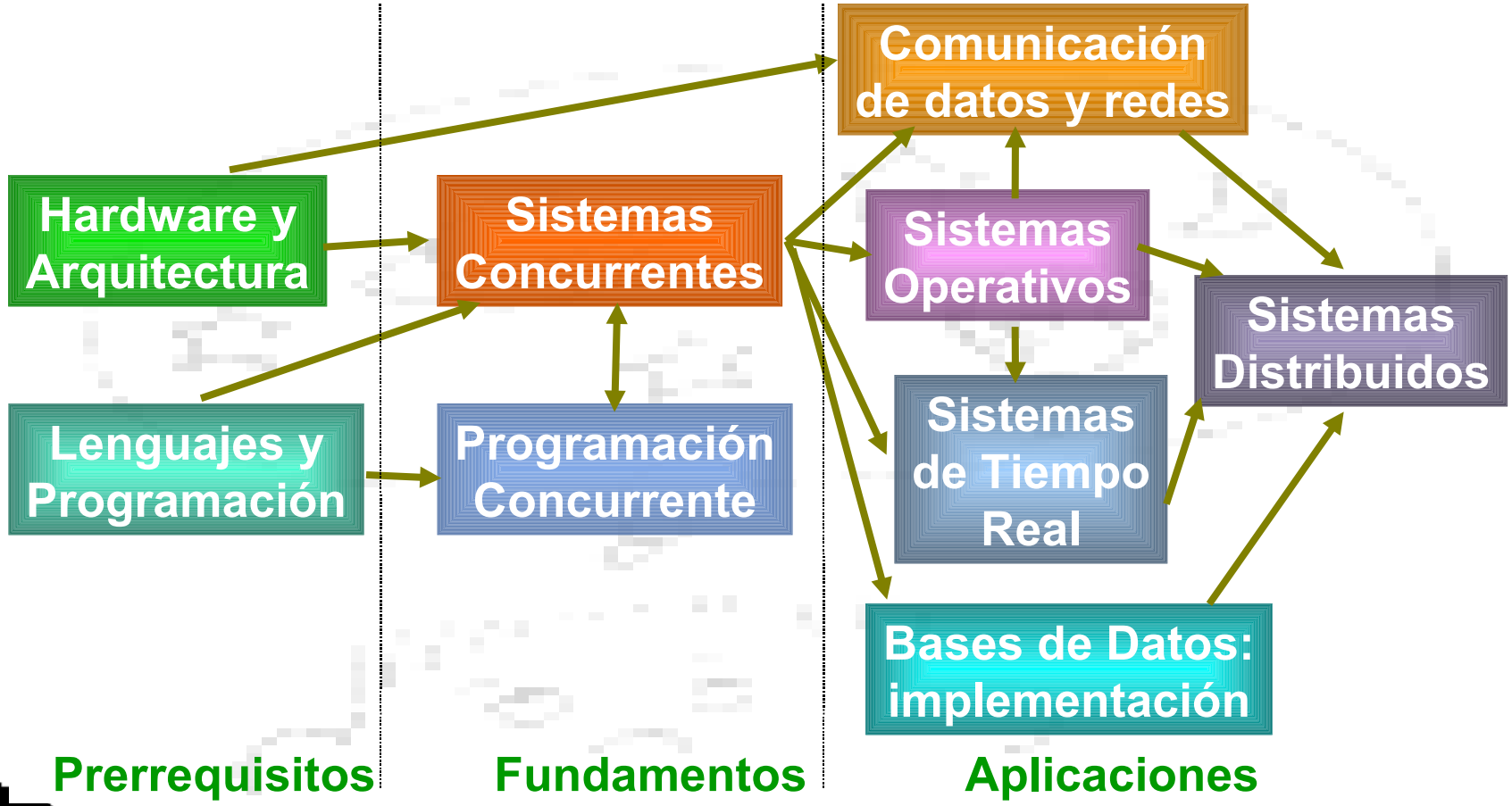
- Sistemas Concurrentes
- Definición de Sistema Operativo
- Breve historia de los SOs.
- Soporte hardware para los SOs.
- Estructuras de SOs.

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



Sistemas concurrentes en nuestras titulaciones

Asignaturas directamente relacionadas:

- Sistemas Operativos I y II
- Diseño de Sistemas Operativos
- Programación Concurrente
- Programación Distribuida y Paralela.
- Gestión de Bases de Datos.
- Arquitectura de Computadores II
- Sistemas Informáticos Distribuidos.
- Desarrollo de Sistemas de Tiempo Real.
- Bases de Datos Distribuidas.

¿Qué es un Sistema Operativo? ...

- Un sistema operativo (SO) es un programa de ordenador que gestiona los recursos de la máquina (CPU, memoria, dispositivos de E/S, discos, red, etc.)
- ¿Por qué no gestiona el usuario esos recursos?
  - Coexisten varias aplicaciones
  - El SO crea una máquina virtual “privada”
  - El SO eleva el nivel de abstracción para los usuarios

... ó mejor ¿qué hace?

- **SO como máquina virtual** – presenta al usuario una máquina abstracta más fácil de programar que el hardware de base, ocultando su complejidad, y tratando homogéneamente diferentes objetos de bajo nivel (archivos, procesos, dispositivos, etc.)
- **SO como gestor de recursos** – controla y protege los recursos (procesadores, dispositivos de E/S, memoria, etc.) de los programas y/o usuarios.

El hardware

- La realidad tangible es el hardware del computador, que contiene elementos de:
  - Procesamiento: CPU, DMA, procesadores de E/S
  - Memoria: RAM, disco, cintas,..
  - Dispositivos de E/S: teclado, ratón, pantalla, impresora,..
  - Dispositivos de comunicaciones: red ethernet, líneas serie y paralelas, ...

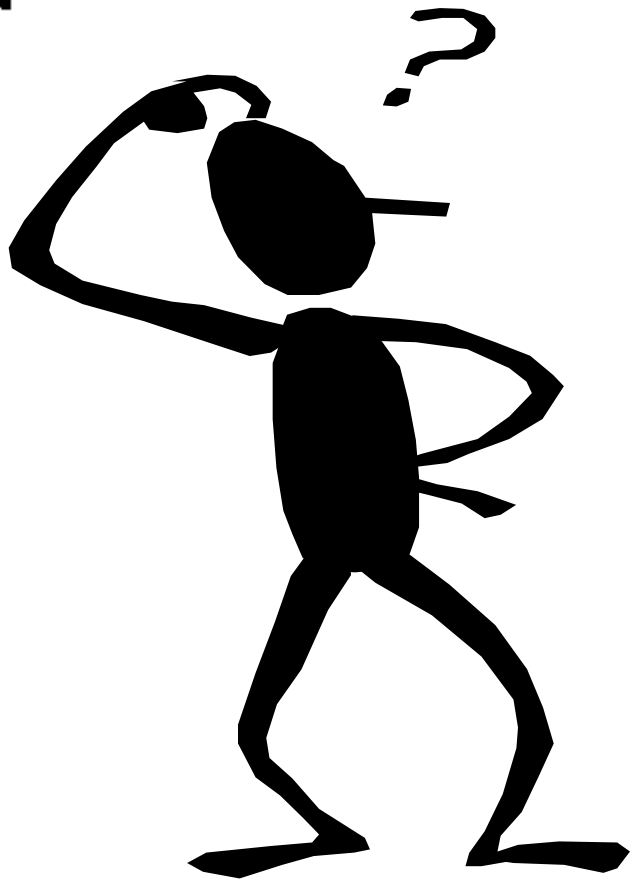
Las abstracciones

- Abstracción=considerar una cualidad, estado, acción o fenómeno con independencia del objeto en que existe o por que existe".
- El SO simplifica el hardware construyendo una serie de abstracciones:
  - Proceso: una CPU dedicada a un programa.
  - Memoria virtual: simula una RAM infinita.
  - Archivo: memoria permanente de datos con tipo.
  - Canal: dispositivo E/S de datos con tipo.
  - Shell: interfaz de usuario programable …

Las técnicas

- Ideamos técnicas para construir abstracciones
  - Planificadores de recursos: CPUs, discos,etc.
  - Gestión de memoria: paginación, ...
  - Sistema de archivos: designación, ...
  - Interrupciones, sondeo
  - Bufering
  - Hebras (*threads*).
  - Dominios, listas de acceso, capacidades, criptosistemas de clave pública, ...

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.

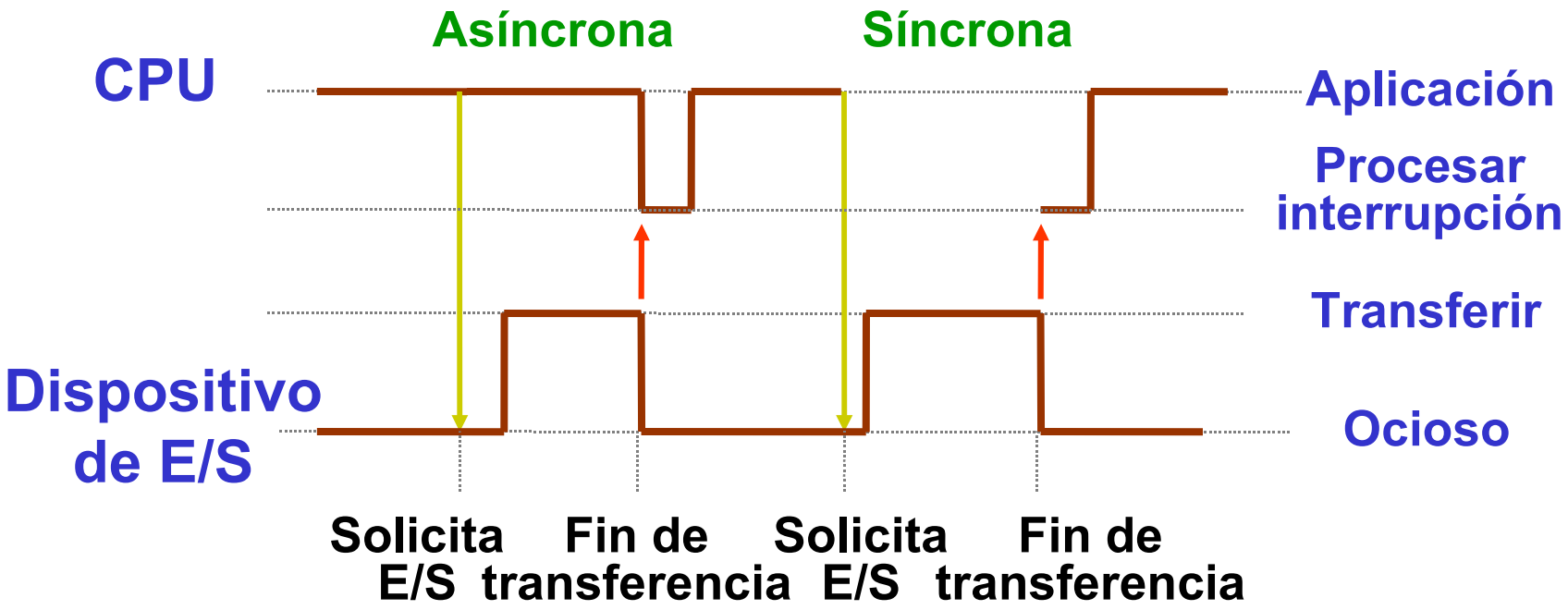
Problemas (sists. Batch)

- El problema principal se debía a las largas esperas entre lotes de trabajos.
- La 2ª generación de ordenadores introduce:
  - Hardware separado para gestionar las E/S
  - Aparece del concepto interrupción
  - un programa puede seguir ejecutándose mientras se esta 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



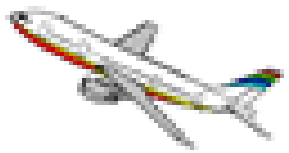
Soporte para la multiprogramación

- 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. 2
  - Permiten la participación activa de los usuarios en la edición, depuración de programas, y ejecución de los procesos.

SOs de Tiempo-Real

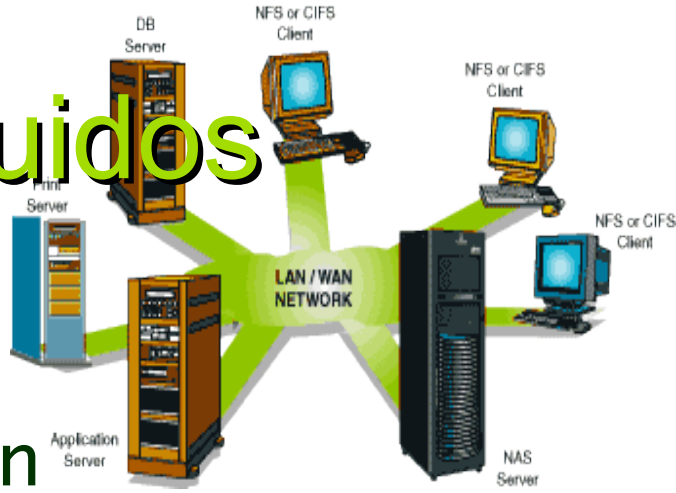


- Los STRs se suelen usar en aplicaciones especializadas, p.ej. sistemas de control, ...

- Garantizar la respuesta a sucesos físicos en intervalos de tiempo fijos.

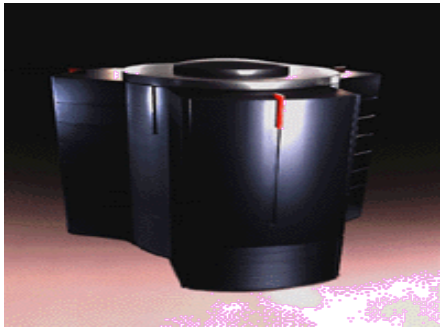
- Problema: poder ejecutar las actividades del sistema con el fin de satisfacer todos los requisitos críticos de las mismas.
- Con el uso de aplicaciones multimedia sobre PC's, todos los SOs tienen requisitos de tiempo-real.

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.

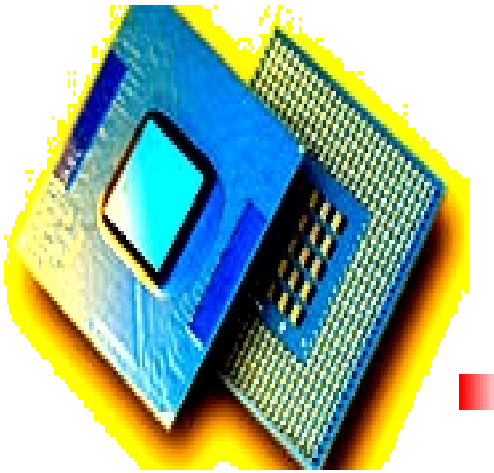
SOs de Internet

- 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).

SOs para WID

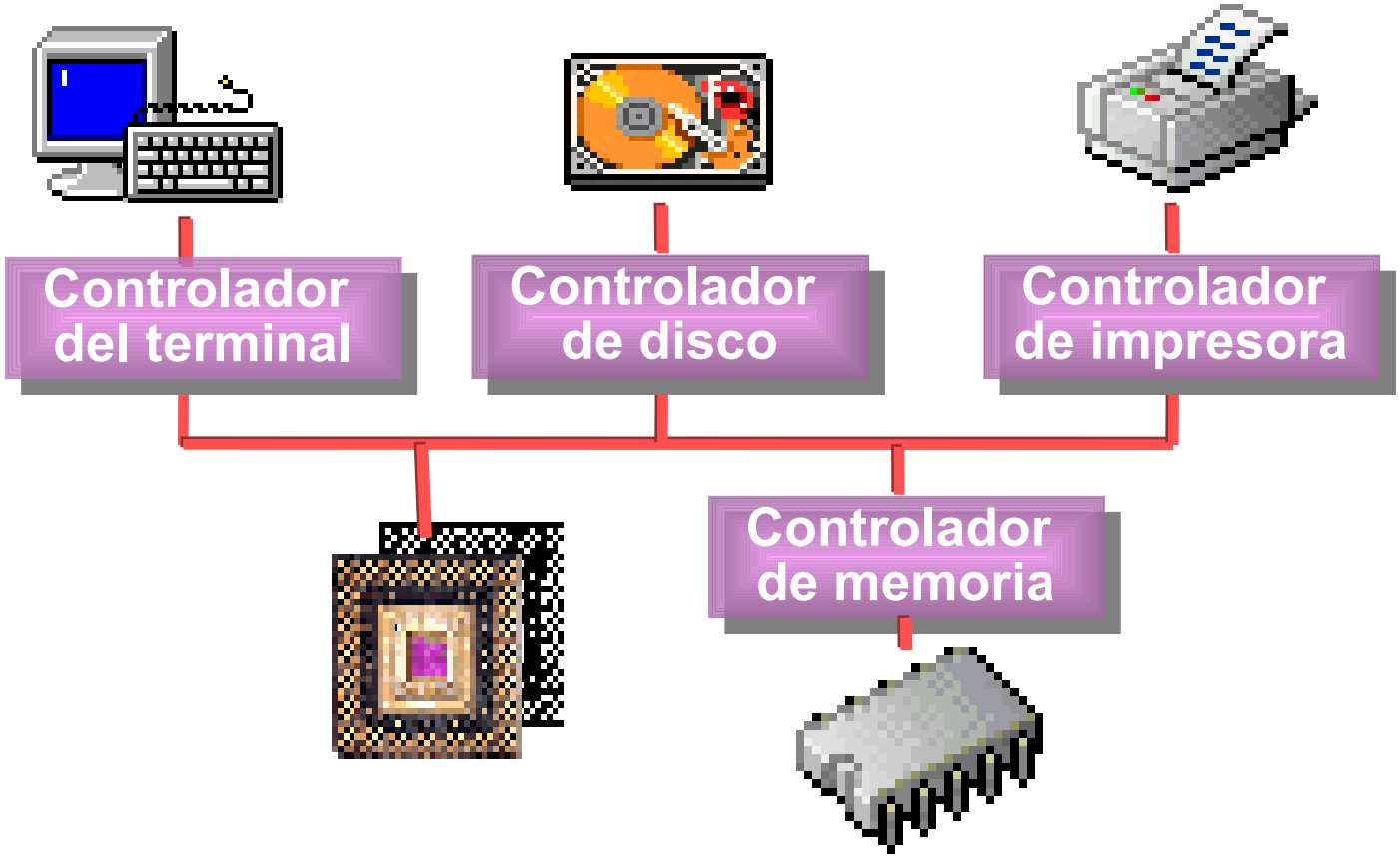
- WID = **Wireless Information Devices**
- Los WID son pequeños y móviles
- Es muy comprometido reducir un SO de PC para ellos: su SO debe gestionar muy bien sus escasos recursos; en especial: su limitada potencia eléctrica, de CPU y escasa memoria.

Soporte Hardware al SO



- En lo que sigue veremos:
  - Interrupciones y excepciones
  - Protección hardware: Modo Dual de Operación
    - **Llamadas al sistema**
- Necesitaremos repasar: el funcionamiento de un sistema de computador

Sistema de computador



Funcionamiento del sistema

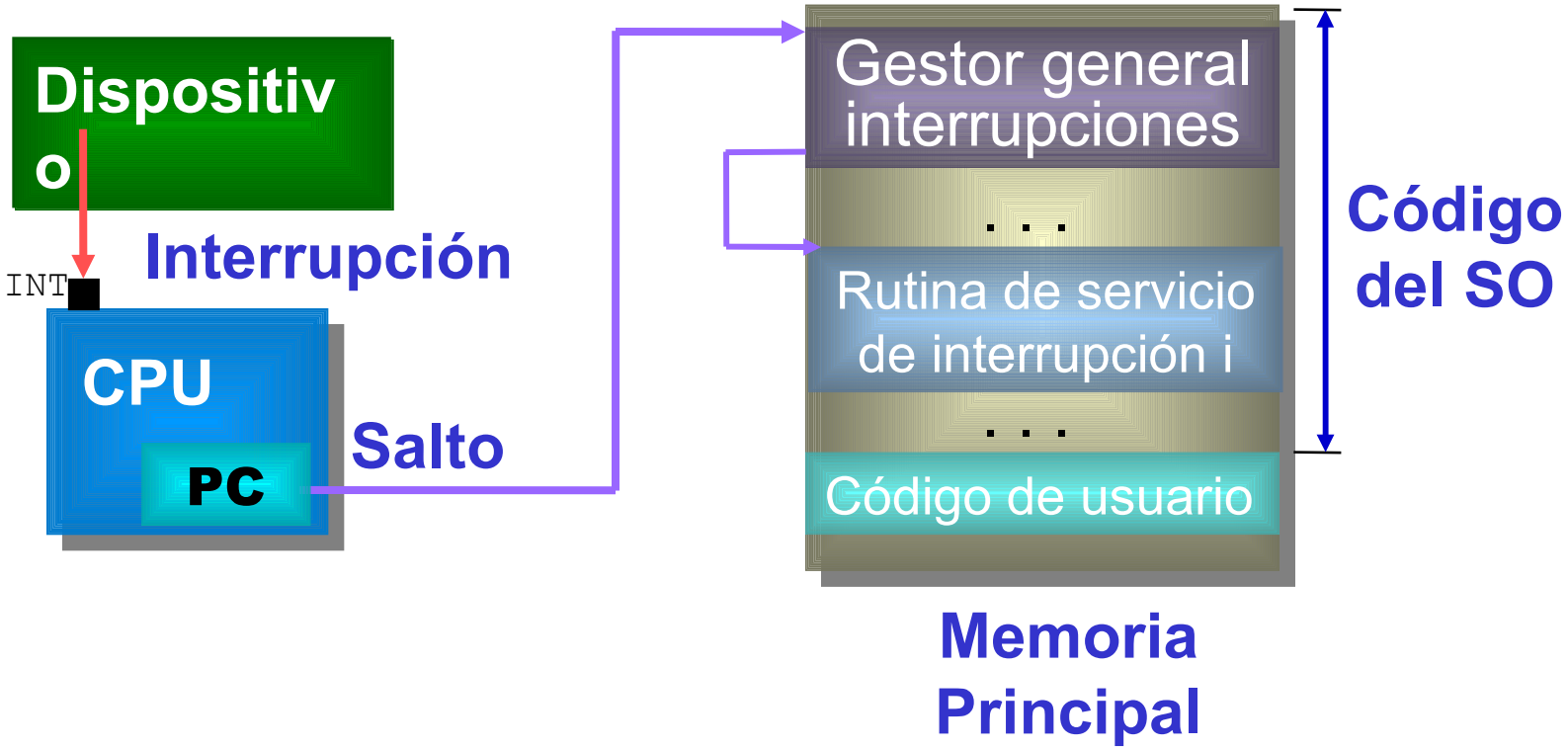
- CPU y controladores se comunican mediante un bus común. La CPU mueve datos desde/hacia memoria principal a/desde los búferes locales.
- Los dispositivos de E/S y la CPU operan concurrentemente.
- Los controladores de dispositivos gestionan el tipo particular de dispositivo; tienen un búfer local, colas de peticiones y registros de estado; informan a la CPU que han finalizado mediante una interrupción.

Interrupciones y excepciones

- Una **interrupción** es un suceso externo al procesador que cambia el flujo normal de ejecución del procesador. Es un evento asíncrono.
- Una **excepción** es un suceso inesperado interno al procesador (p.ej. desbordamien-to aritmético, dirección inválida, rastreo de programas, instrucción privilegiada, etc.). Es un evento síncrono.

No todos los manuales de arquitectura definen y tratan estos dos conceptos de igual manera.

Manejo de las interrupciones



Gestor de interrupciones

- Realiza las siguientes operaciones:
  - Salva el contenido de los registros de la CPU, los que usa el proceso ejecutándose
  - Determina el tipo de interrupción (*sondeo* o *interrupciones vectorizadas*) y ejecuta la rutina de servicio de interrupción (ISR) adecuada para tratar con el dispositivo.
  - Restaura los registros salvados anteriormente.

Excepciones

- La gestión de excepciones es similar a la usada para las interrupciones. Al detectarse la excepción, se transfiere el control al manejador de excepciones (su dirección viene dada por vector de excepciones).
- La diferencia básica con las interrupciones es que las excepciones suelen tratarse en espacio de usuario; el SO sólo las notifica.
- Trampa – instrucción máquina que genera una excepción si se cumple cierta condición.

Control del SO

- Los SOs están controlados por interrupción, esto es, los manejadores de interrupciones y excepciones (trampas) son los puntos de entrada al SO. Cuando no hay procesos de usuario realiza su propio trabajo o se para.
- Cada vez son más frecuentes las aplicaciones controladas por interrupción, p. ej. aplicaciones WIMP (Ventanas, Iconos, Menus, Punteros), como Delphi, Vbasic, Java, etc.

Protección hardware

- Un SO multitarea debe evitar:
  - las interferencias entre los procesos de usuario,
  - y entre los procesos y el SO.
- Para llevarlo a cabo de forma segura necesitamos apoyo hardware:
  - Modo dual de operación de la CPU
  - Protección de memoria: Tema
  - Protección de la CPU: Tema 2

Instrucciones privilegiadas

- Definimos *instrucciones privilegiadas* a aquellas que pueden potencialmente dañar a otros procesos. P. ej. operaciones de E/S, actualizar el reloj, desactivar interrupciones, manipular la MMU o el bit de protección, etc.
- Protegemos a los procesos y al SO si controlamos la ejecución de instrucciones privilegiadas.

Modo Dual de Operación de la CPU

- Las CPUs tienen al menos dos *modos de funcionamiento*:
  - **Modo usuario** – el intento de ejecución de una instrucción privilegiada en este modo produce una excepción.
  - **Modo kernel** (*supervisor* o *sistema*) – en él se puede ejecutar cualquier instrucción.
    - El código del SO se ejecuta en este modo. Por ello, se denomina kernel al código del SO que se ejecuta en este modo.

Cambio de modo

- El cambio de modo usuario a modo kernel se produce cuando:
  - Se produce una interrupción, excepción, o una trampa.
  - El cambio de modo se materializa:
    - Salva información sobre que estaba haciendo (ver Tema 2).
    - "Activando" el ***bit de modo*** del registro PSW.
- El retorno a modo usuario es más simple: deshacer lo andado.

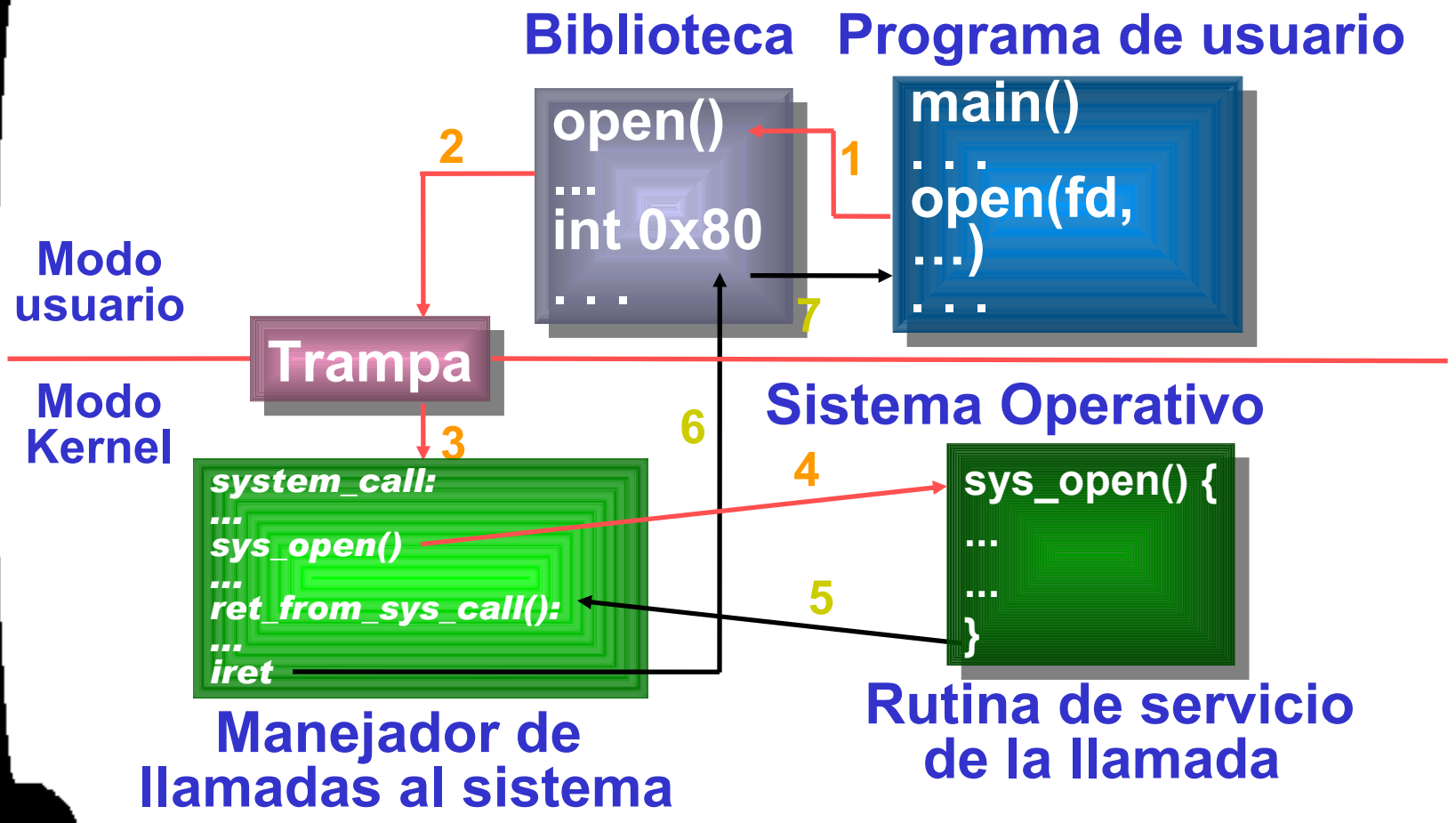
Llamadas al Sistema

- ¿Cómo puede un proceso de usuario realizar una operación privilegiada, p. ej. una E/S, crear un proceso, obtener memoria, etc.?
- Solución: los servicios del SO se obtienen invocando un procedimiento del sistema a través de una **llamada al sistema**.
- Las llamadas al sistema se implementan a través de una trampa o, como algunos la denominan, “interrupción software”.

Realizando una llamada al sistema

- Se genera una trampa a una posición específica del SO. P. ej. en Linux sobre Intel x86, con **`INT 0x80`**.
- Este salto transfiere el control a la rutina de servicio de trampas del SO, y el bit de modo se activa (estamos en modo kernel).
- El SO verifica que los parámetros son correctos y legales. Si es así, ejecuta la solicitud, y devuelve el control a la siguiente instrucción. Si no, devuelve error.

Arquitectura del sistema



Llamadas al sistema: paso de parámetros

- Para pasar parámetros se utiliza alguno de estos tres métodos:
  - Paso de parámetros en los registros. P.ej. como en Linux:

```
load eax, valor
load bx, valor
...
INT 0X80
```

  - Almacenarlos en una tabla de memoria y pasar la dirección de la tabla como parámetro en un registro.
  - El programa empuja los parámetros en la pila, y el SO los recupera de la pila.

Estructuras de SOs

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

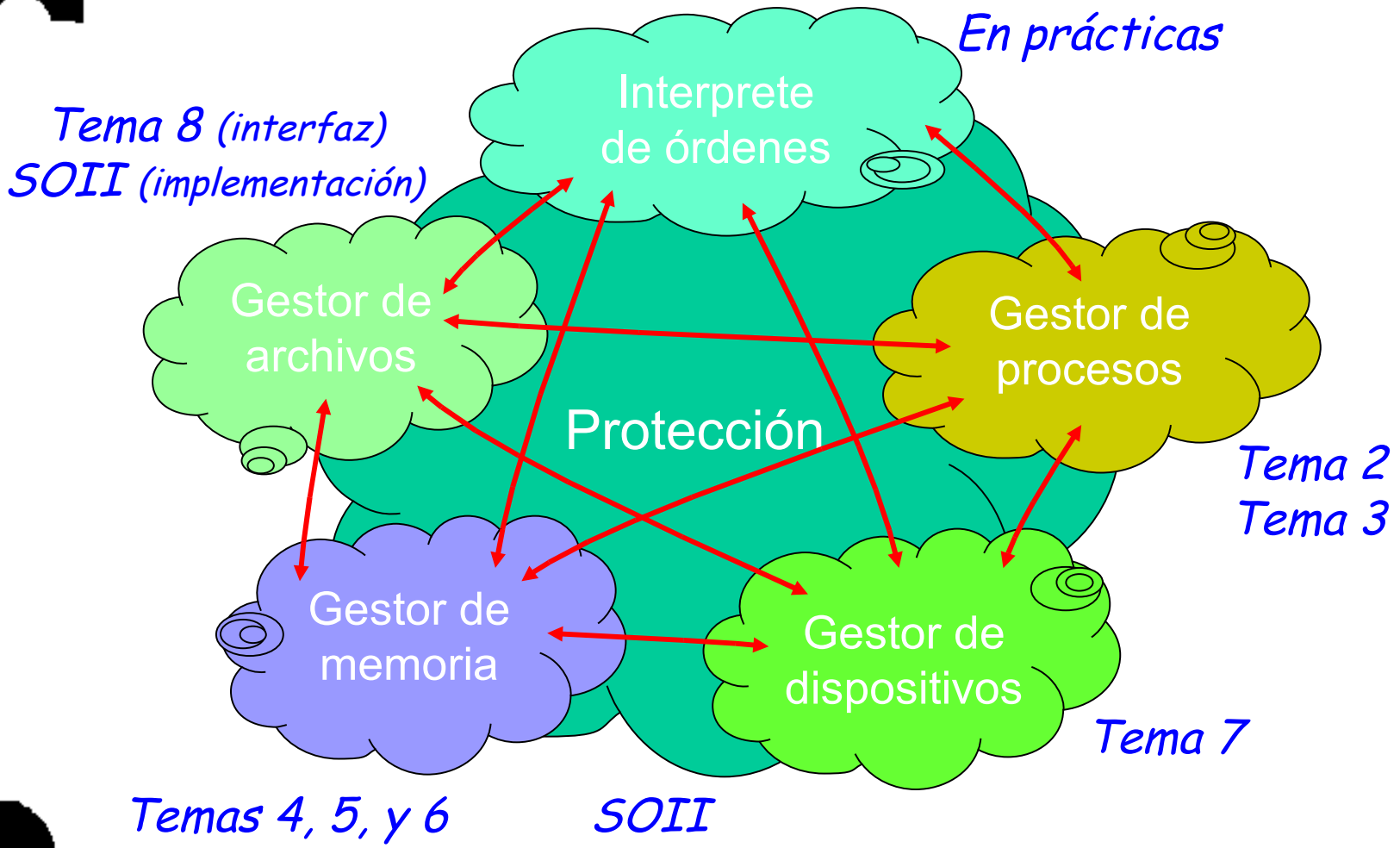
Componentes de un SO

- 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 del SO



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 almacenamien-to 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 de 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 esta localizada en un único módulo.

Intérprete de órdenes

- Programa o proceso que maneja la inter-pretación 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

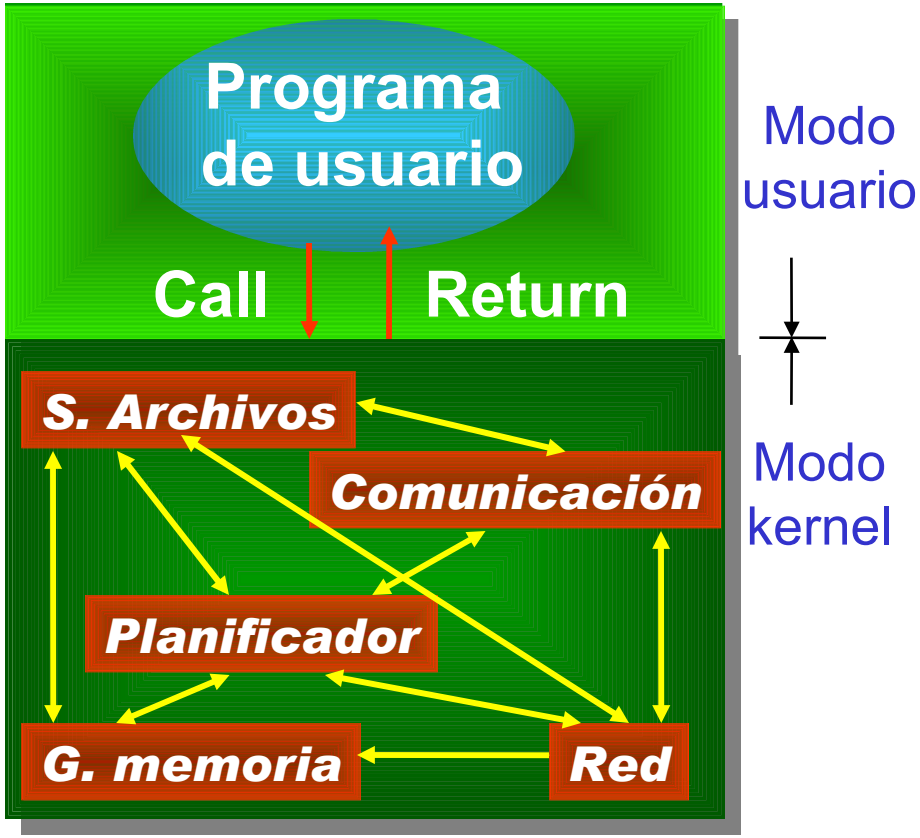
- Eficiente – debe ser lo más eficiente posible, pues consume muchos ciclos de CPU.
- Fiable – Fiabilidad se refiere a dos aspectos diferentes pero relacionados:
  - Robustez, el SO debe responder de forma predecible a condiciones de error, incluidos fallos hardware.
  - El SO debe protegerse activamente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



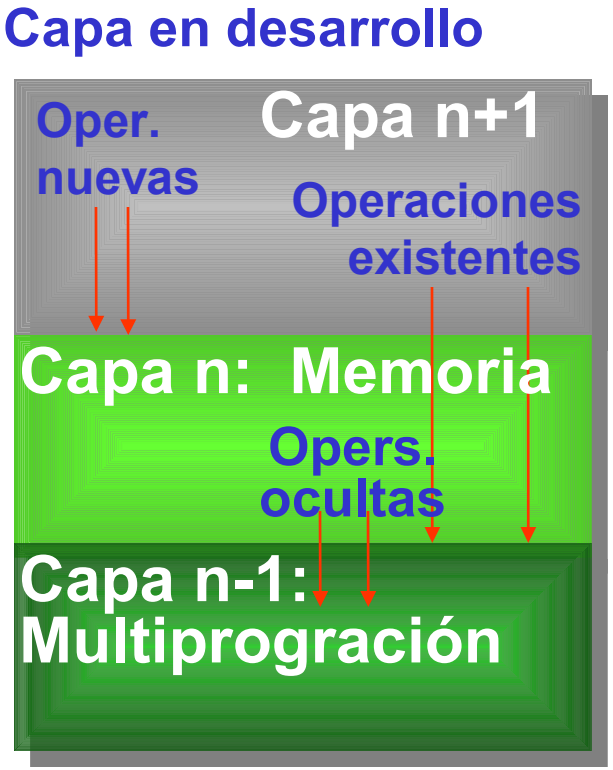
Problemas de los sists. monolíticos

- 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 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.

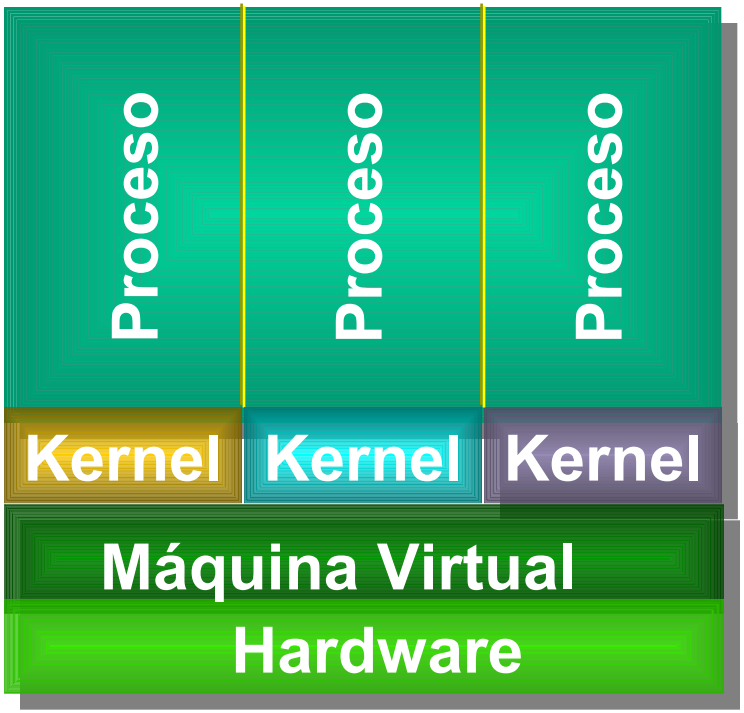


Problemas de la estructura de capas

- 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 conclu-sión 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.

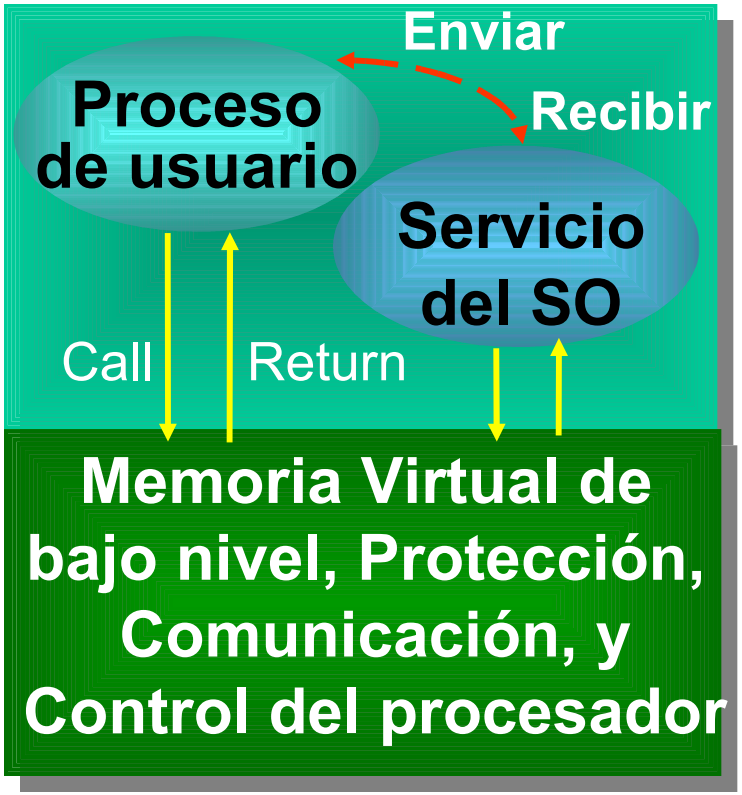
  P. Ej. Los bits de la palabra de estado en un procesador Intel tienen diferente significado según estemos en modo real o protegido.

Algunos conceptos

- El espacio de direcciones de un proceso son aquellas direcciones de memoria que el proceso puede direccionar. (Tema 4)
- 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 esta 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.



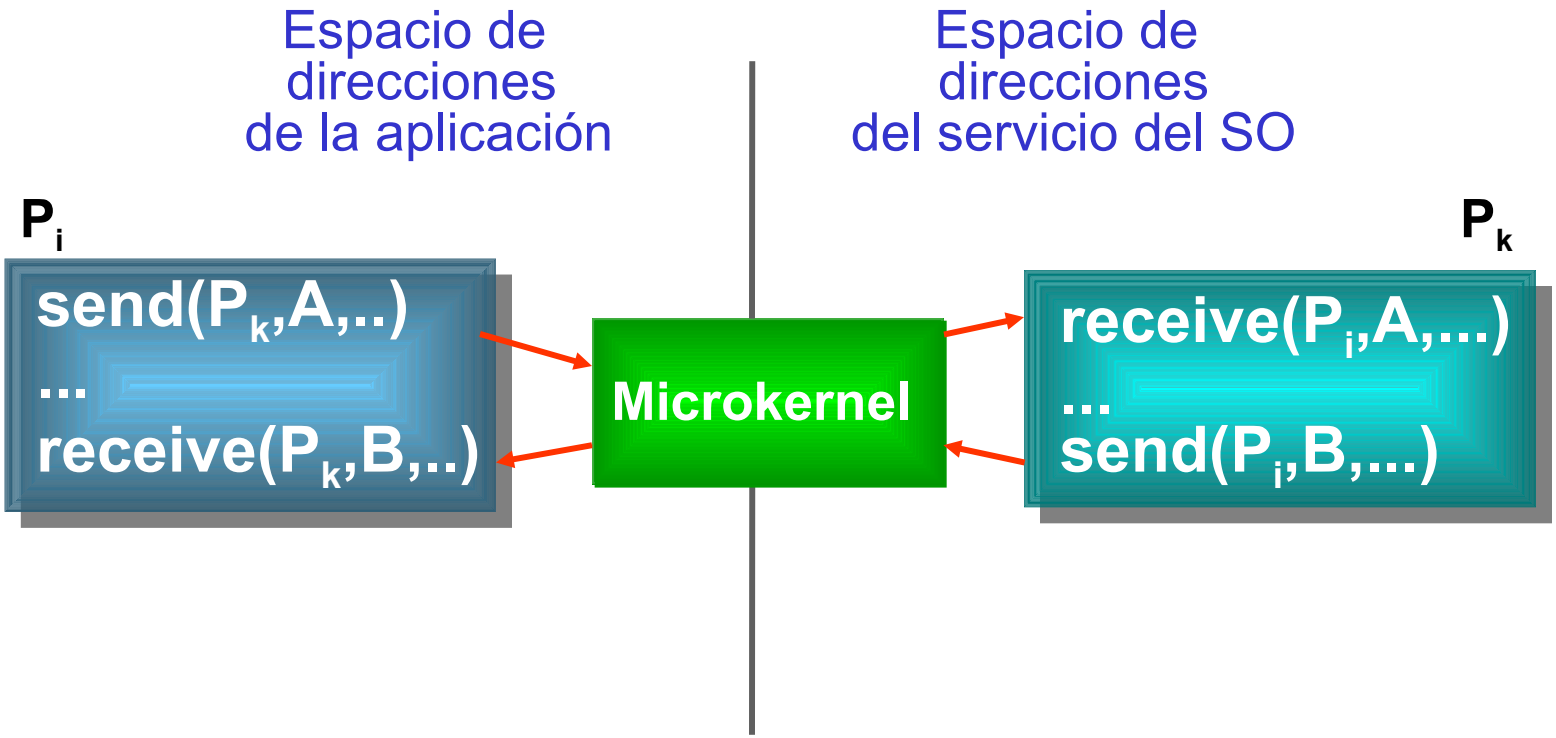
Ventajas del µkernel

- 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 (más en Tema 3):
  - La aplicación envía el mensaje de solicitud, **`send(P,A)`**, y espera la respuesta con **`receive(P,b)`**.
  - El kernel verifica el mensaje y lo entrega al servidor.
  - El servidor que esta en espera, realiza el servicio solicitado y devuelve el resultado.

Esquema


Espacio de direcciones de la aplicación
Espacio de direcciones del servicio del SO
P_i
P_k
send(P_k,A,..)
...
receive(P_k,B,..)
Microkernel
receive(P_i,A,...)
...
send(P_i,B,...)

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.

Un microkernel: Windows NT™