

Lección 1

Introducción a los sistemas operativos

Sistemas Operativos
Ingeniería Informática

Objetivos

- ▶ Conocimiento de los métodos de gestión interna de recursos en un sistema operativo.
- ▶ Introducción a las técnicas para la programación y modificación del sistema operativo.

Lecturas recomendadas

Base



1. Carretero 2020:
 1. Cap. 1 y 2
2. Carretero 2007:
 1. Cap. 2

Recomendada



1. Tanenbaum 2006:
 1. Cap. 1
2. Stallings 2005:
 1. Parte uno. Transfondo.
3. Silberschatz 2006:
 1. Cap. 1

¡ATENCIÓN!

- ❑ Estas transparencias son un guión para la clase.
- ❑ Los libros dados en la bibliografía junto con lo explicado en clase representa el material de estudio para el temario de la asignatura.

Contenidos

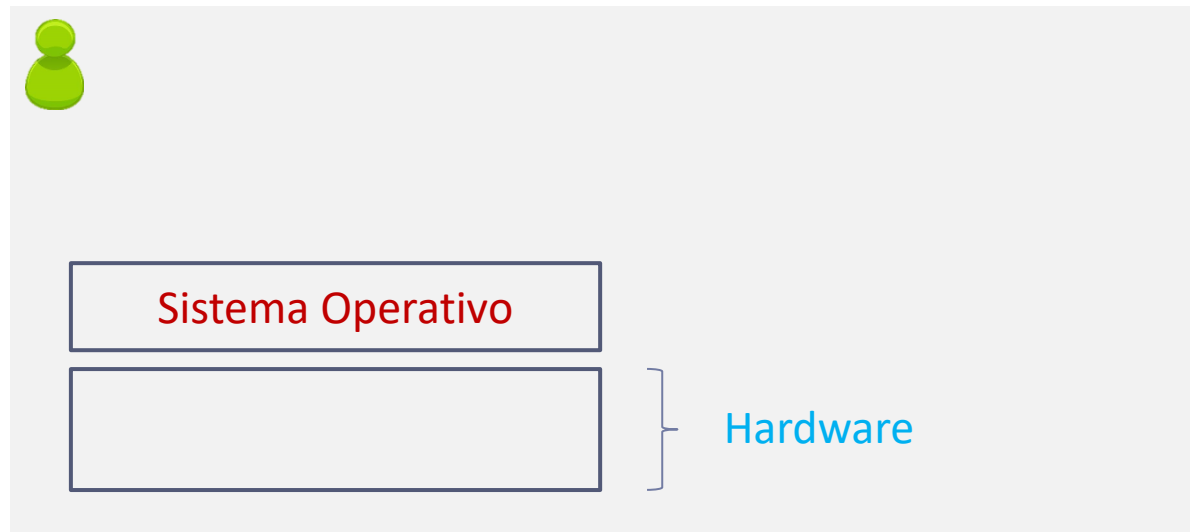
1. ¿Qué es un sistema operativo?
2. Evolución de los sistemas operativos.
3. Características.
4. Tareas de un sistema operativo.
5. Funcionamiento básico.
6. Estructura del sistema operativo.
7. Arranque del sistema operativo.

Contenidos

1. **¿Qué es un sistema operativo?**
2. Evolución de los sistemas operativos.
3. Características.
4. Tareas de un sistema operativo.
5. Funcionamiento básico.
6. Estructura del sistema operativo.
7. Arranque del sistema operativo.

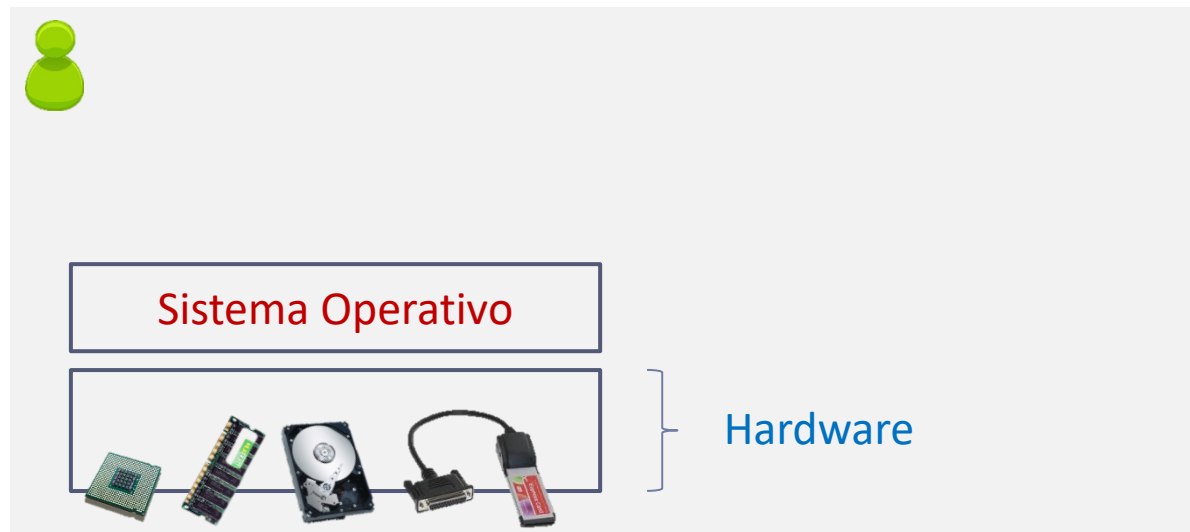
¿Qué es un sistema operativo?

- ▶ **Sistema operativo:** software destinado a permitir la comunicación del **usuario** con un **ordenador** y gestionar sus recursos de manera cómoda y eficiente.



¿Qué es un sistema operativo?

- ▶ **Sistema operativo:** software destinado a permitir la comunicación del **usuario** con un **ordenador** y gestionar sus recursos de manera cómoda y eficiente.
- ▶ Complejidad para encubrir el hardware subyacente (en el S.O.)



¿Qué es un sistema operativo?

- ▶ **Sistema operativo:** software destinado a permitir la comunicación del **usuario** con un **ordenador** y gestionar sus recursos de manera cómoda y eficiente.
- ▶ Complejidad para encubrir el hardware subyacente (en el S.O.)
- ▶ Programas del sistema para la gestión de los recursos.



¿Qué es un sistema operativo?

- ▶ **Sistema operativo:** software destinado a permitir la comunicación del **usuario** con un **ordenador** y gestionar sus recursos de manera cómoda y eficiente.
- ▶ Complejidad para encubrir el hardware subyacente (en el S.O.)
- ▶ Programas del sistema para la gestión de los recursos.



Contenidos

1. ¿Qué es un sistema operativo?
2. **Evolución de los sistemas operativos.**
3. Características.
4. Tareas de un sistema operativo.
5. Funcionamiento básico.
6. Estructura del sistema operativo.
7. Arranque del sistema operativo.

Evolución (asociada al hardware)

- ▶ Primera generación (años 50)
 - ▶ Monitor hardware.
- ▶ Segunda generación (años 60)
 - ▶ Procesamiento por lotes.
- ▶ Tercera generación (años 70)
 - ▶ Multiprogramación, tiempo compartido y multiusuario.
- ▶ Cuarta generación (años 80 – actualidad)
 - ▶ Sistemas distribuidos.
 - ▶ Interfaz gráfica, reconocimiento de voz, etc.
 - ▶ Virtualización.

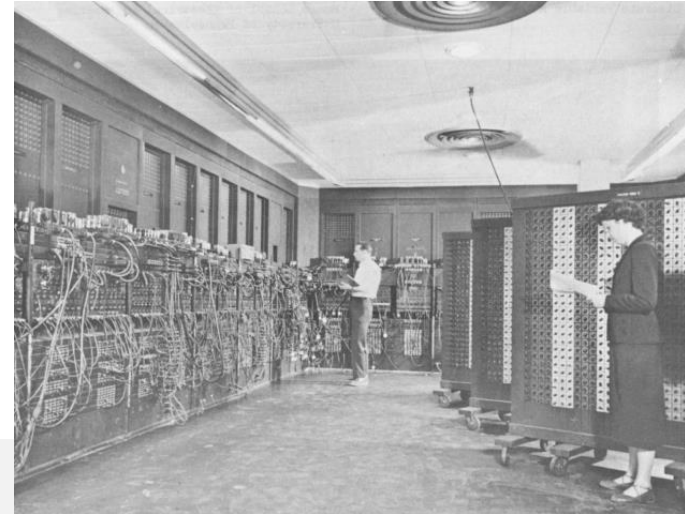
Primera generación (1945-1955)

monitor hardware

Hw

S.O.

- ▶ Tubos de vacío.
 - ▶ ENIAC, UNIVAC, ...
 - ▶ Grandes, lentos y caros.
 - ▶ Complicación para programar y depurar.
- ▶ Monitor hardware.
 - ▶ Asistente para el diagnóstico.



Segunda generación (1955-1965)

procesamiento por lotes

Hw

S.O.

► Transistores.

- IBM 7090, IBM 1620, ...
- Más pequeños, rápidos y fiables.
- Ligeras mejoras en su programación.



► Procesamiento por lotes.

- Trabajos ejecutados uno tras otro.
- Enlace de rutinas (librerías) a programas.
- Gestión de archivos, dispositivos de E/S y almacenamiento secundario.
- Aparición de OS/360



Tercera generación (1965-1980)

multiprogramación, multitarea y multiusuario

Hw

S.O.

- ▶ Circuitos integrados.
 - ▶ PDP-8, ...
 - ▶ Más pequeños, rápidos y fiables.
 - ▶ Fortran, C, Pascal, Basic, etc.
- ▶ Multiprogramación.
 - ▶ Gestión de recursos compartidos solapando esperas en CPU.



Tercera generación (1965-1980)

multiprogramación, multitarea y multiusuario

Hw

S.O.

- ▶ Circuitos integrados.
 - ▶ PDP-8, ...
 - ▶ Más pequeños, rápidos y fiables.
 - ▶ Fortran, C, Pascal, Basic, etc.



- ▶ Multiprogramación.
 - ▶ Gestión de recursos compartidos solapando esperas en CPU.
- ▶ Tiempo compartido y multiusuario.
 - ▶ Compartición de recursos y multiplexación del uso de la CPU.
 - ▶ Aparición de Multics, Unix, CP/M, etc.

Cuarta generación (1980-) distribuidos, con interfaz gráfica, etc.

Hw

S.O.

- ▶ El microprocesador.
 - ▶ IBM-PC, ...
 - ▶ Ordenadores personales.
 - ▶ C++, etc.



- ▶ **Distribuidos (de tiempo real)**
 - ▶ Conexión a Internet, aplicaciones de trabajo distribuido
- ▶ **Interfaz gráfica.**
 - ▶ System 5, Windows 1.0, etc.

Cuarta generación (2005-) virtualización, multiprocesamiento, etc.

Hw

S.O.

- ▶ Los *system-on-chip* y *multicores*.

- ▶ Tables, smartphones, ...
- ▶ Era post-PC.
- ▶ .net, java, python, php, etc.



- ▶ Virtualización.

- ▶ *Cloud computing*, *legacy computing*, desktop móvil, etc.

- ▶ Nuevas formas de interacción.

- ▶ Reconocimiento del habla, entrada multitáctil, etc..

Cuarta generación (2005-)

virtualización, multiprocesamiento, etc.

Hw

S.O.

- ▶ Los *system-on-chip* y *multicores*.

- ▶ Tables, smartphones, ...
- ▶ Era post-PC.
- ▶ .net, java, python, php, etc.



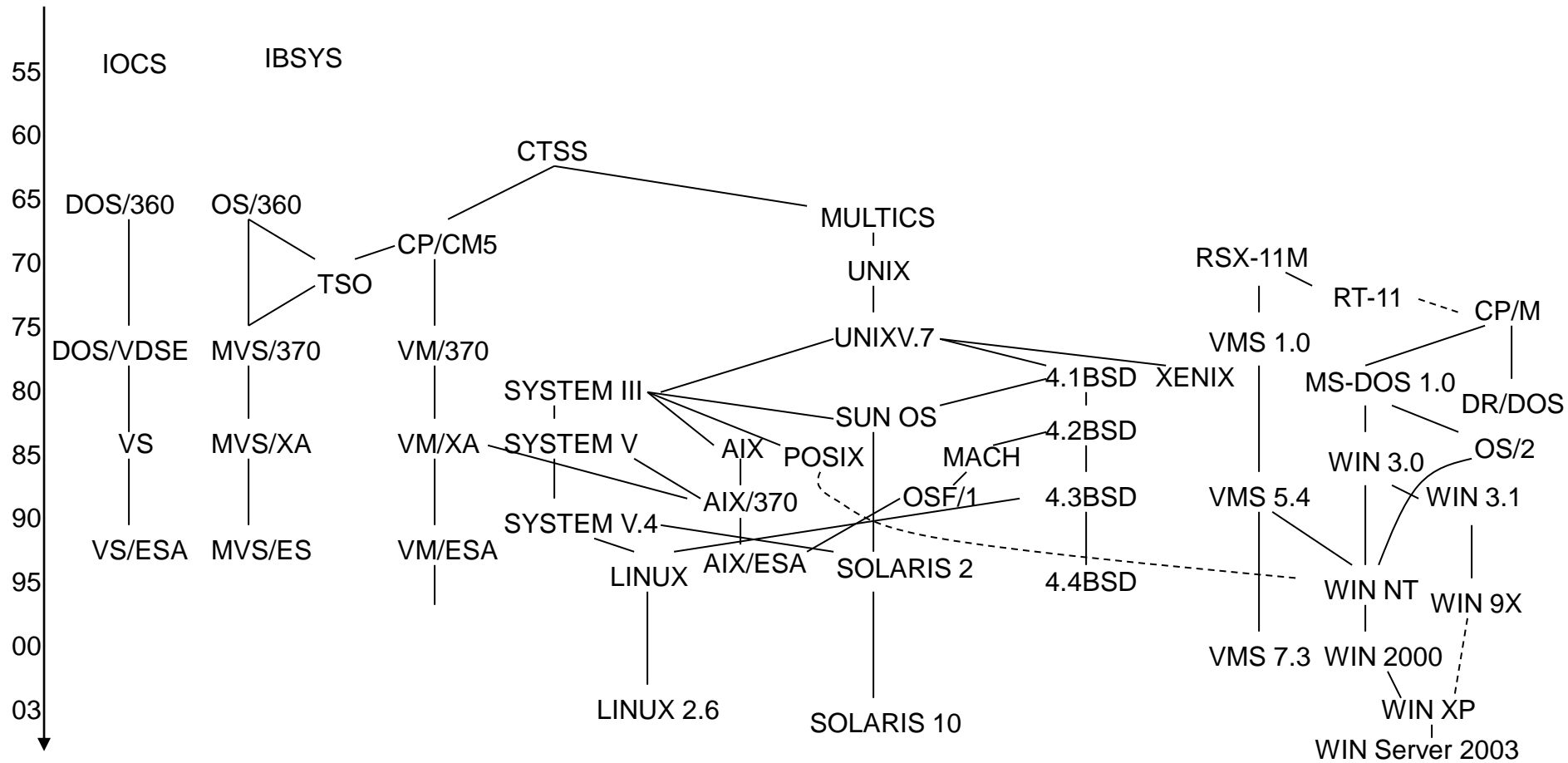
- ▶ Virtualización.

- ▶ *Cloud computing*, *legacy computing*, desktop móvil, etc.

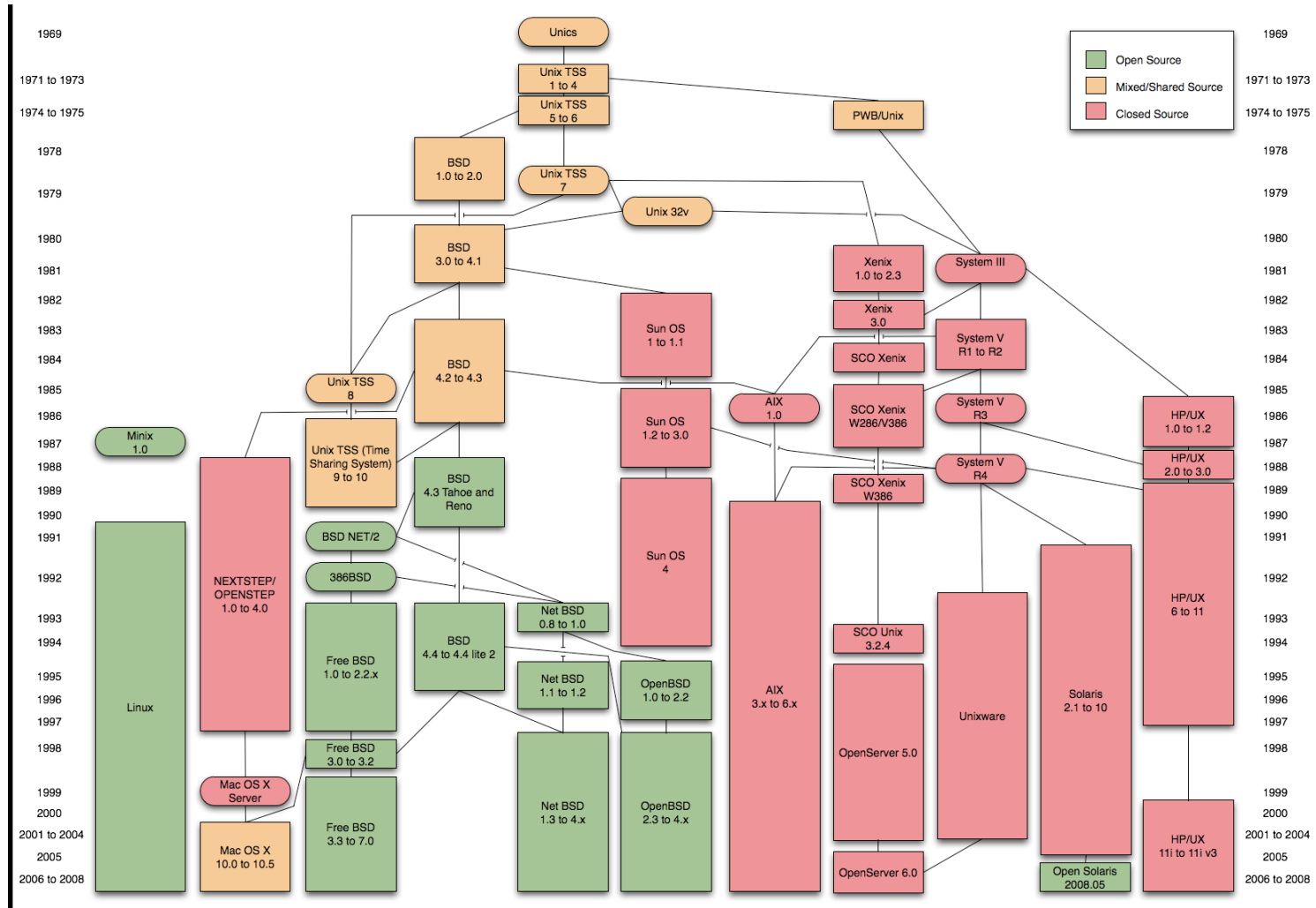
- ▶ Nuevas formas de interacción.

- ▶ Reconocimiento del habla, entrada multitáctil, etc..
- ▶ Pantallas 3D, holografías, etc.

Evolución de los sistemas operativos



Evolución de los sistemas operativos UNIX



Contenidos

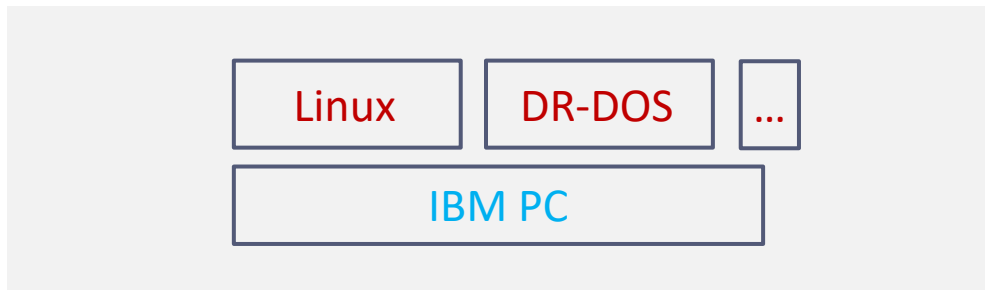
1. ¿Qué es un sistema operativo?
2. Evolución de los sistemas operativos.
3. **Características.**
4. Tareas de un sistema operativo.
5. Funcionamiento básico.
6. Estructura del sistema operativo.
7. Arranque del sistema operativo.

Principales características

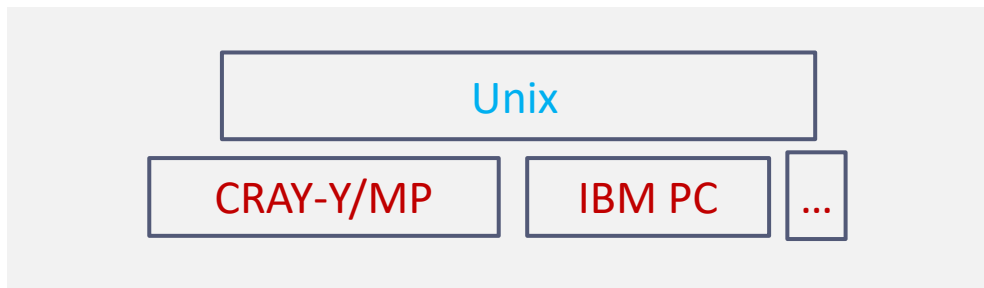
- ▶ Versátil
 - ▶ Portabilidad
- ▶ Adaptativo
- ▶ Multidisciplinar
- ▶ Complejo
- ▶ Delicado

1) Versátil

- ▶ Mismo equipo, diferentes SSOO: **IBM PC**

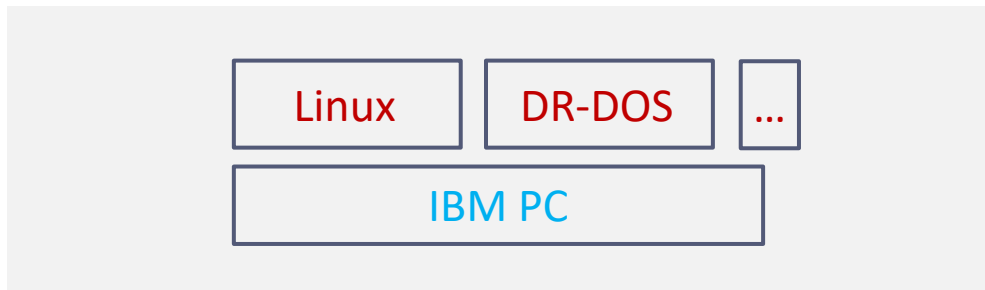


- ▶ Mismo SO, diferentes equipos: **Unix**

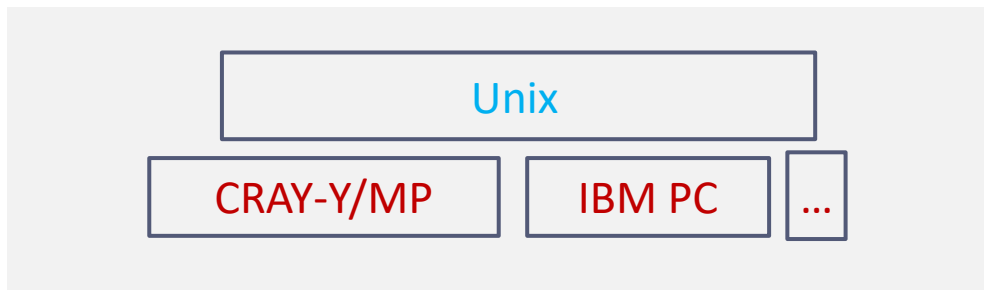


Versátil: Portabilidad

- ▶ Mismo equipo, diferentes SSOO: IBM PC



- ▶ Mismo SO, diferentes equipos: Unix



Portabilidad

Versátil



Mainframe
OS/360, z/OS, ...



Supercomputador
Unix, Linux, ...



Miniordenadores y PC
Unix, MacOS, Windows, ...



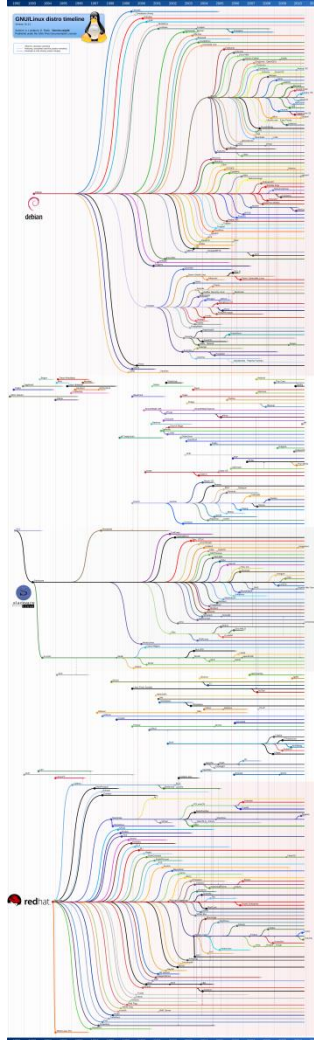
Empotrados
VxWorks, QNX, LynxOS,
Android, iOS,
Windows Embedded, ...

2) Continuos cambios para adaptarse

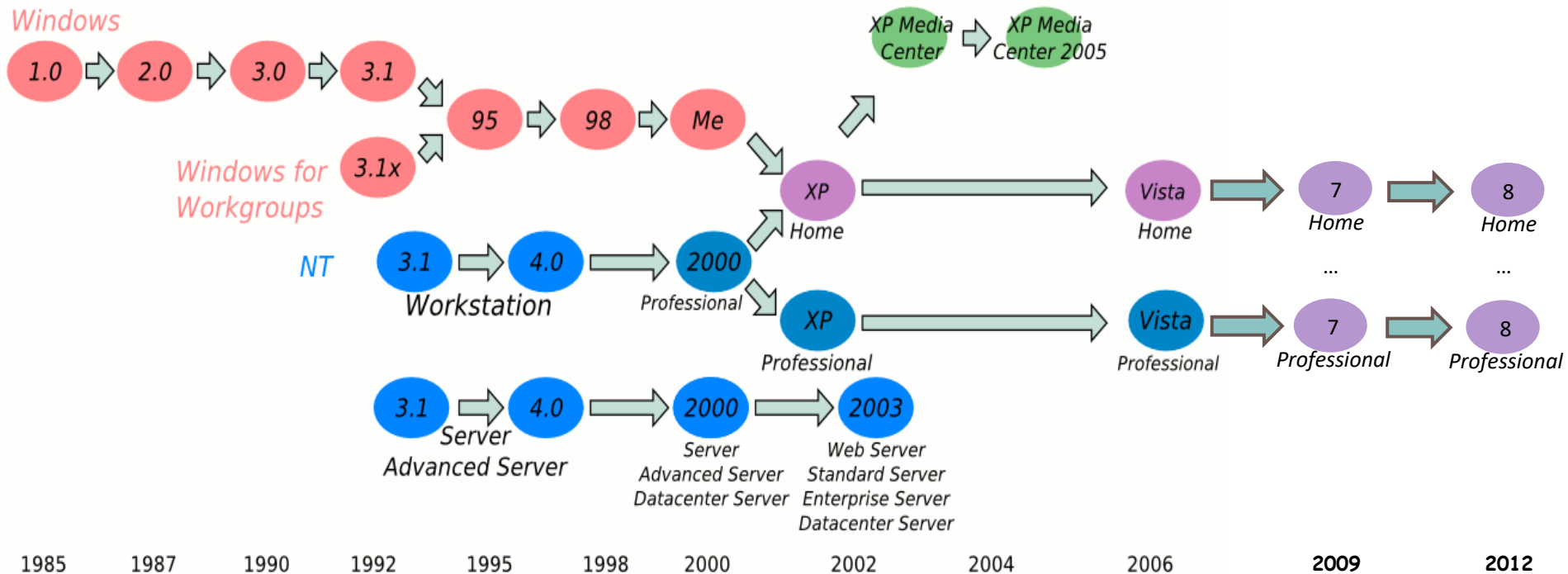
- ▶ A las nuevas demandas de los usuarios:
 - ▶ Reconocimiento de voz, entrada multitáctil, etc.
- ▶ A la evolución o nuevo tipo de hardware:
 - ▶ Controladores para todo tipo de nuevos dispositivos
 - ▶ Sistemas multicore, virtualización, etc.
- ▶ A integrar soluciones de distintos entornos:
 - ▶ Procesamiento por lotes, multiprogramación, tiempo compartido, etc.
 - ▶ Multiusuario, trabajo colaborativo, etc.
 - ▶ Sistemas distribuidos, servicios en red, etc.

Evolución de los sistemas operativos

Distribuciones Linux (distros) hasta 2010



Evolución de los sistemas operativos Windows



3) Software multidisciplinar

- ▶ Software multidisciplinar:
 - ▶ Integra trabajos de diferentes áreas: interfaces de usuario, software de sistema, inteligencia artificial, etc.

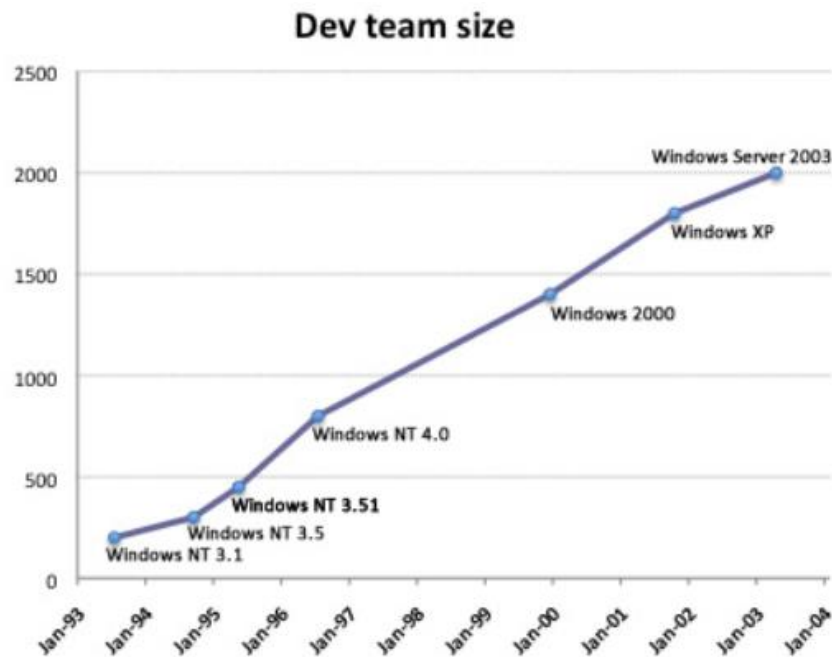


4) Software Complejo

- ▶ Software complejo:
 - ▶ Muchas líneas de código.
 - ▶ Muchos equipos de trabajo.

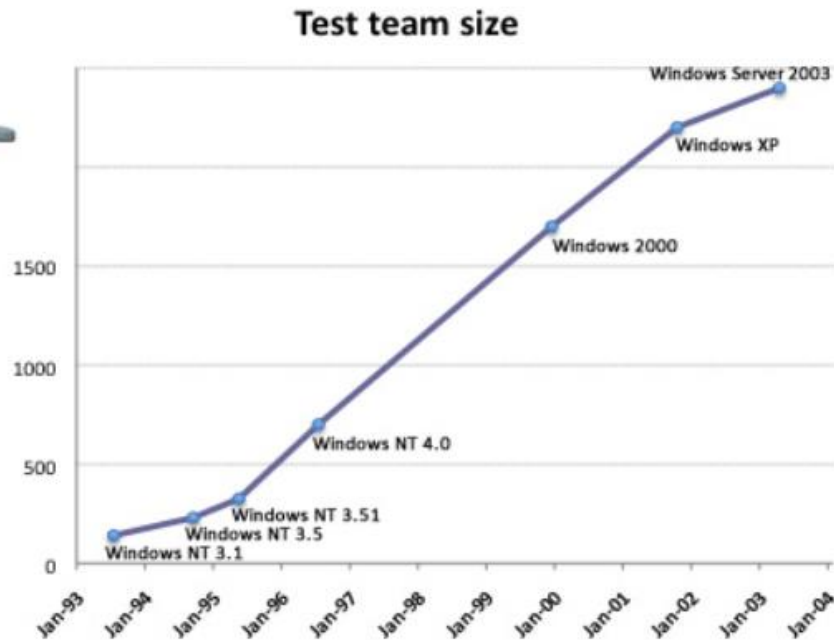
4) Software Complejo

- ▶ Software complejo:
 - ▶ Muchas líneas de código.
 - ▶ Muchos equipos de trabajo.



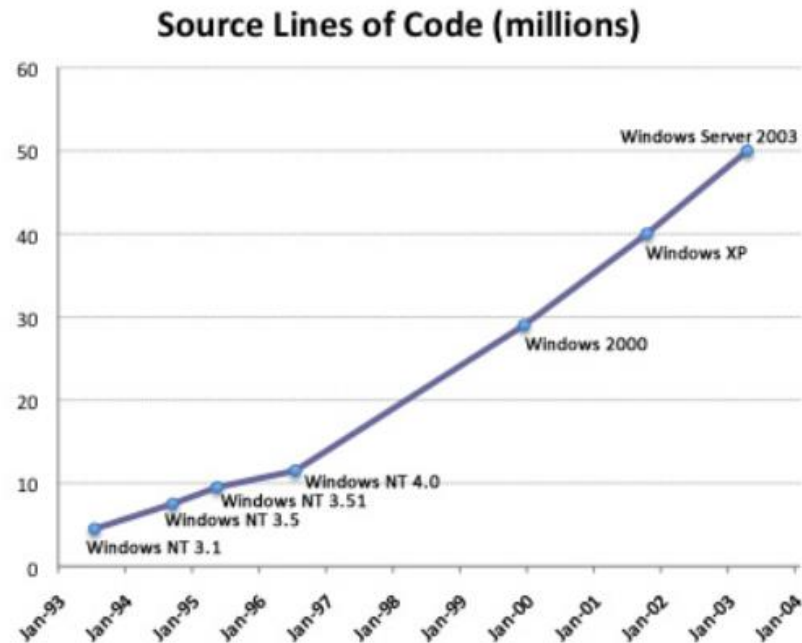
4) Software Complejo

- ▶ Software complejo:
 - ▶ Muchas líneas de código.
 - ▶ Muchos equipos de trabajo.



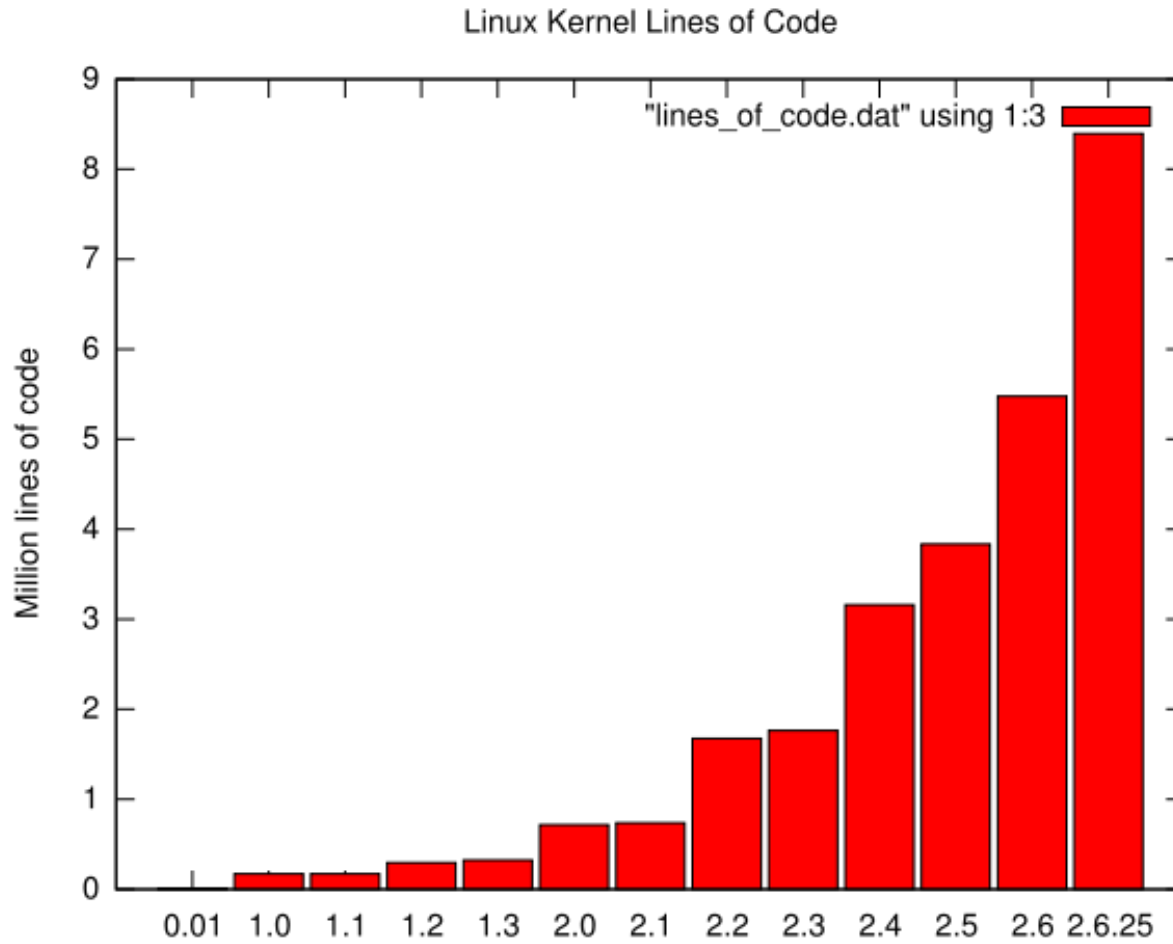
4) Software Complejo

- ▶ Software complejo:
 - ▶ Muchas líneas de código.
 - ▶ Muchos equipos de trabajo.



Complejidad de los sistemas operativos

Linux



Fedora Core 9 => ~200

5) Software delicado

- ▶ Software delicado:
 - ▶ Un fallo en un driver (software controlador de un dispositivo) puede bloquear todo el sistema.
 - ▶ Trata datos de distintas aplicaciones de distintos usuarios que no deben ser perdidos o trasladados a manos incorrectas.

5) Software delicado

- ▶ Software delicado:
 - ▶ Un fallo en un driver (software controlador de un dispositivo) puede bloquear todo el sistema.
 - ▶ Trata datos de distintas aplicaciones de distintos usuarios que no deben ser perdidos o trasladados a manos incorrectas.

(a) Industry Average: "about 15 - 50 errors per 1000 lines of delivered code." He further says this is usually representative of code that has some level of structured programming behind it, but probably includes a mix of coding techniques.

(b) Microsoft Applications: "about 10 - 20 defects per 1000 lines of code during in-house testing, and 0.5 defect per KLOC (KLOC IS CALLED AS 1000 lines of code) in released product (Moore 1992)." He attributes this to a combination of code-reading techniques and independent testing (discussed further in another chapter of his book).

(c) "Harlan Mills pioneered 'cleanroom development', a technique that has been able to achieve rates as low as 3 defects per 1000 lines of code during in-house testing and 0.1 defect per 1000 lines of code in released product (Cobb and Mills 1990). A few projects - for example, the space-shuttle software - have achieved a level of 0 defects in 500,000 lines of code using a system of format development methods, peer reviews, and statistical testing."

Principales características

resumen

- ▶ Versátil
 - ▶ Portabilidad
- ▶ Adaptativo
- ▶ Multidisciplinar
- ▶ Complejo
- ▶ Delicado

Objetivos en el diseño de un sistema operativo



Sistema Operativo

Hardware

- ▶ **Rendimiento: eficiencia y velocidad**
 - ▶ Baja sobrecarga, uso adecuado de los recursos
- ▶ **Estabilidad: robustez y resistencia**
 - ▶ Tiempo de funcionamiento, degradación aceptable, fiabilidad e integridad
- ▶ **Capacidad: prestaciones, flexibilidad y compatibilidad**
- ▶ **Seguridad y protección**
 - ▶ Protección entre usuarios
 - ▶ Sistema seguro para 'los malos'
- ▶ **Portabilidad**
- ▶ **Claridad**
- ▶ **Extensibilidad**

Objetivos en el diseño de un sistema operativo

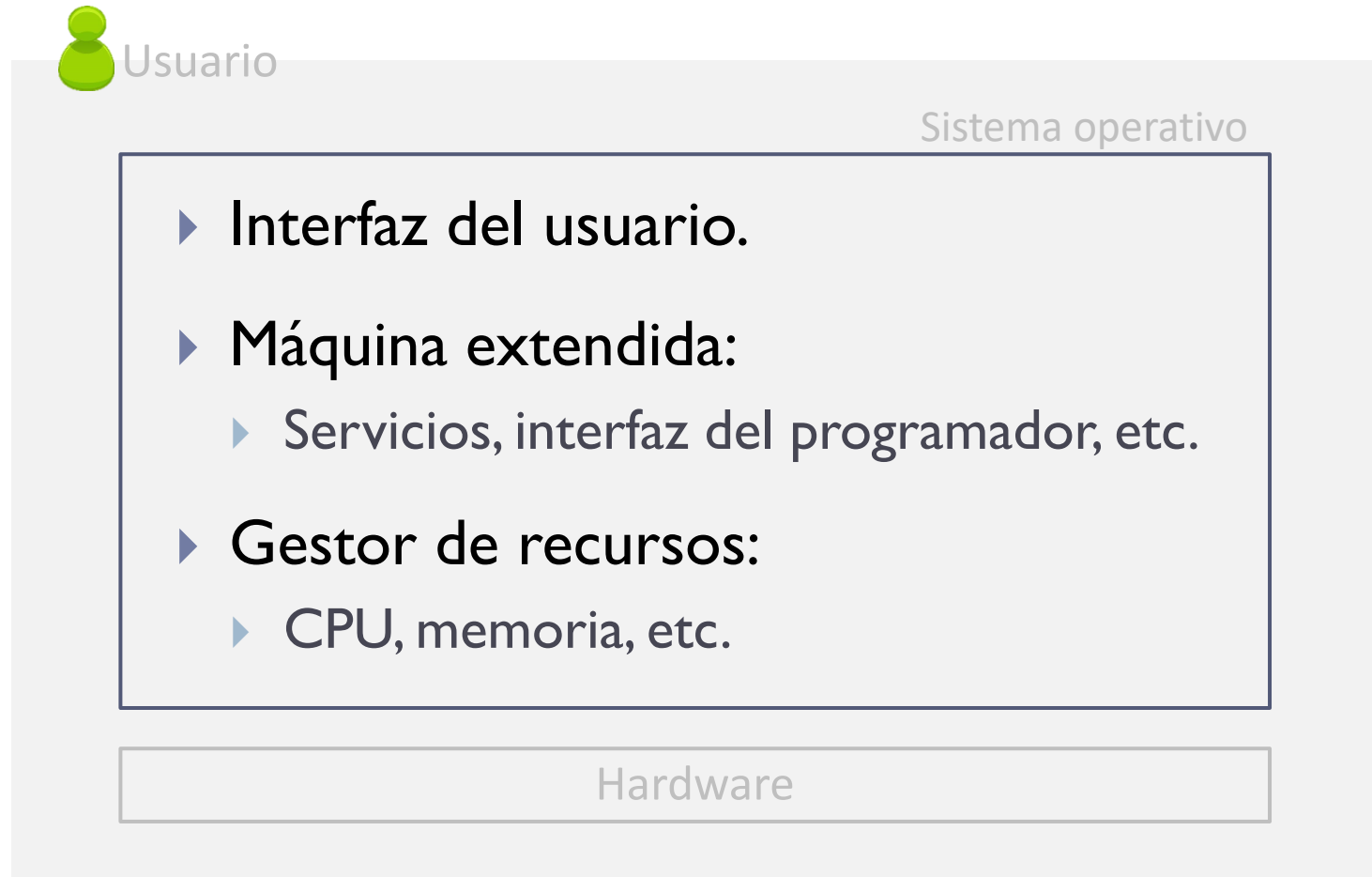


- ▶ Butler Lampson:
«selecciona cualquier terna de objetivos de diseño»
- ▶ Muchos **objetivos** son **antagónicos**:
 - ▶ Eficiencia vs protección
 - ▶ A más comprobaciones, más sobrecarga
 - ▶ Claridad vs compatibilidad
 - ▶ Implementación difícil de antiguos estándares (ej.: signal en Unix)
 - ▶ Flexibilidad vs seguridad
 - ▶ A más cosas que se puedan hacer, más posibles agujeros de seguridad
- ▶ Pero **no todos** son antagónicos:
 - ▶ La portabilidad tiende a mejorar la claridad del código

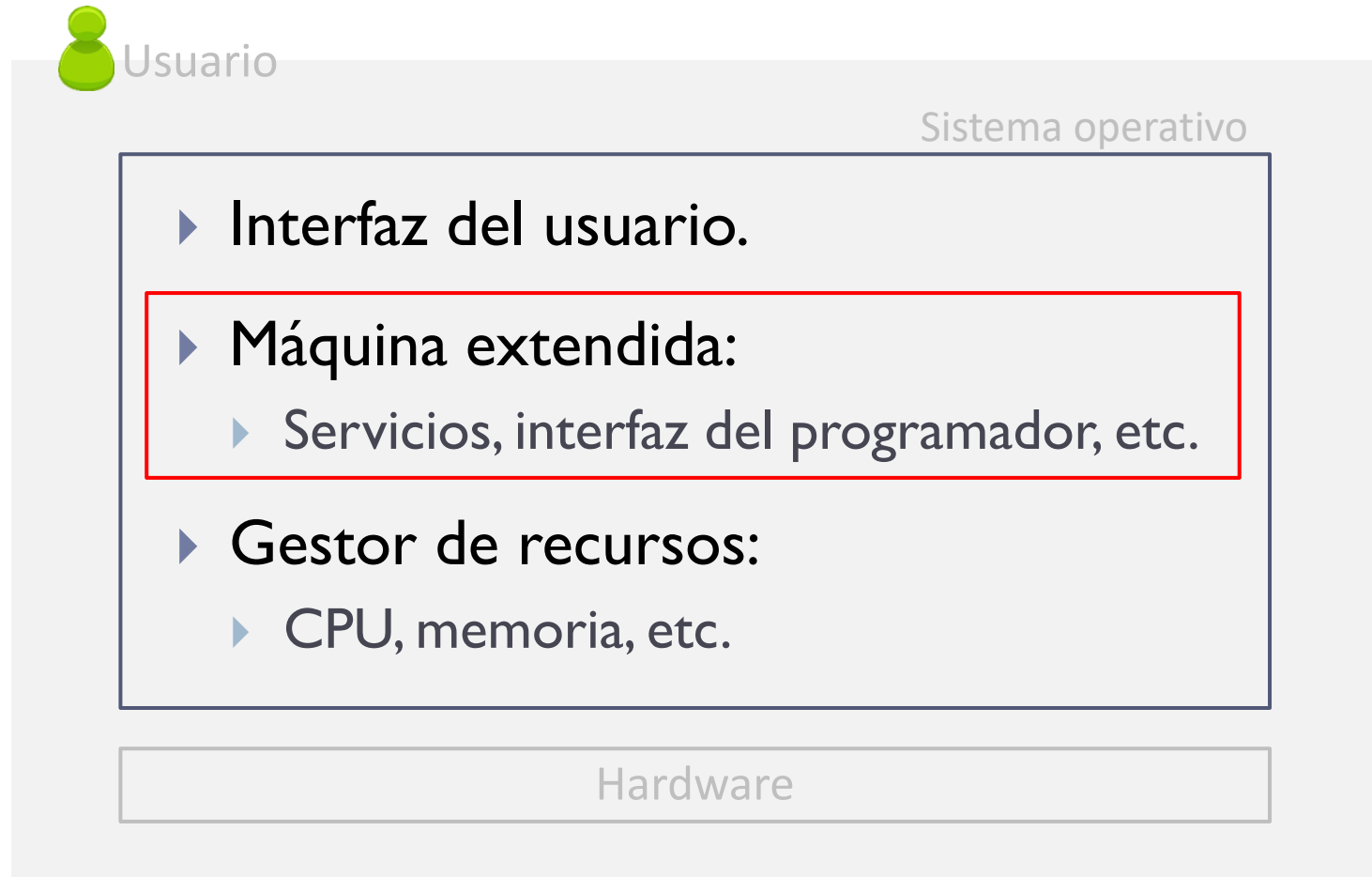
Contenidos

1. ¿Qué es un sistema operativo?
2. Evolución de los sistemas operativos.
3. Características.
4. **Tareas de un sistema operativo.**
5. Funcionamiento básico.
6. Estructura del sistema operativo.
7. Arranque del sistema operativo.

Funciones del sistema operativo



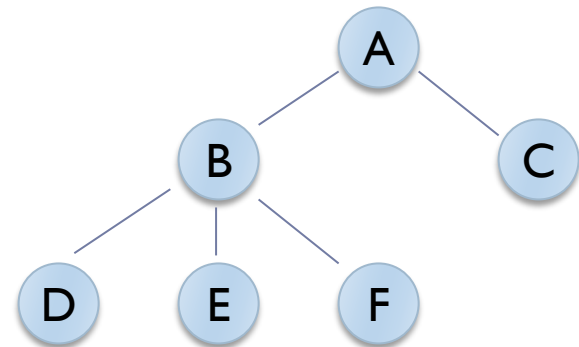
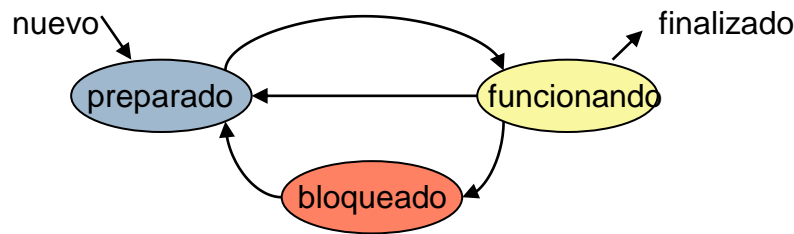
Funciones del sistema operativo



Abstracciones fundamentales

Procesos

- ▶ Procesos, tabla de procesos, árbol de procesos
- ▶ Imagen básica, planificación, señales
- ▶ Identificación de usuario y grupo
- ▶ Intérprete de mandatos (*shell*)

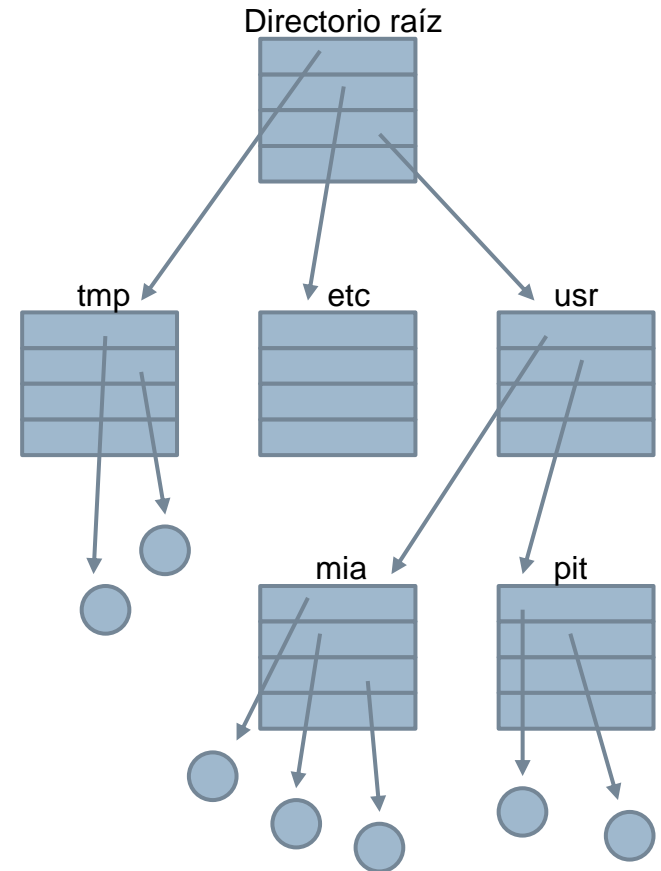


Árbol de procesos

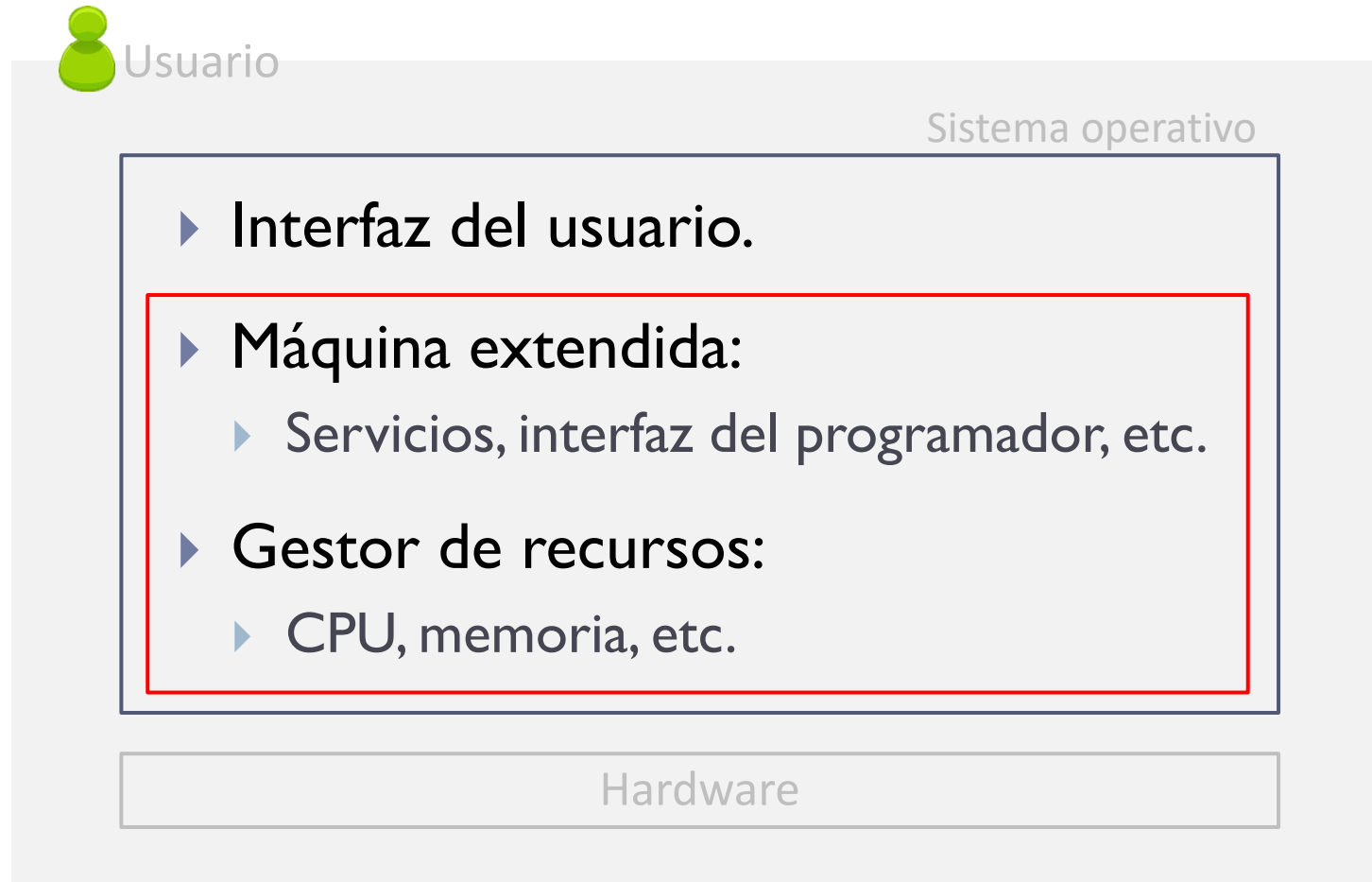
Abstracciones fundamentales

Archivos

- ▶ Archivos y directorios
- ▶ Ruta, directorio de trabajo y raíz
- ▶ Protección
- ▶ Descriptor de archivo
- ▶ Archivos especiales:
 - ▶ Dispositivos E/S
 - ▶ E/S de bloque y de caracteres
 - ▶ Pipes
- ▶ Estándares entrada/salida/error



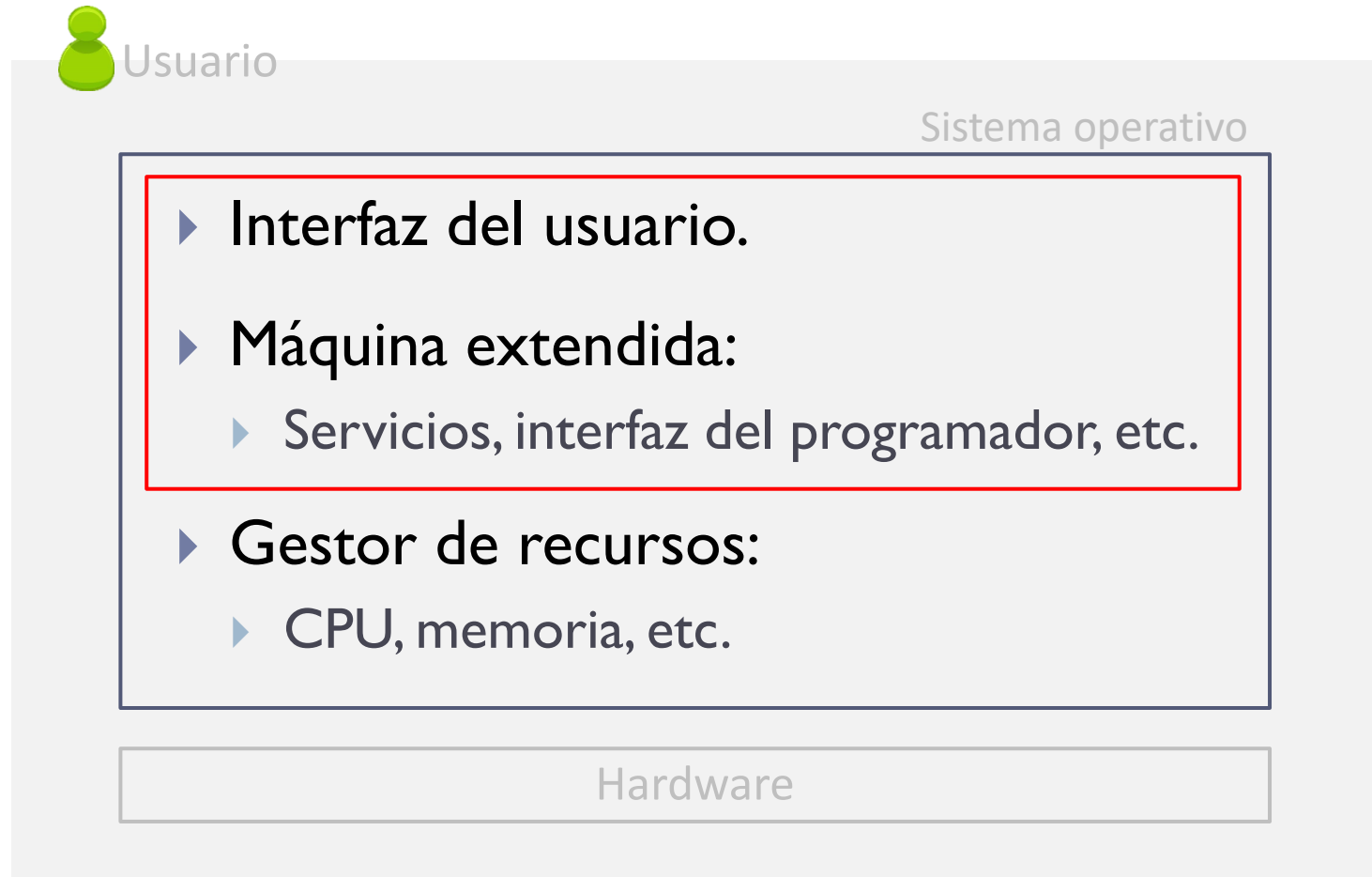
Funciones del sistema operativo



Tareas principales del sistema operativo

- ▶ Gestión de **Procesamiento** – Planificación
 - ▶ Planificación
 - ▶ Prioridades, multiusuario
- ▶ Gestión de **Memoria**
 - ▶ Reparto de memoria entre procesos, con protección y compartición
- ▶ Gestión de **Almacenamiento** – Sistema de Archivos
 - ▶ Ofrecer una visión lógica unificada para usuarios y programas que sea independiente del medio físico
- ▶ Gestión de **Dispositivos**
 - ▶ Encubriendo las dependencias de hardware
 - ▶ Gestión de accesos concurrentes

Funciones del sistema operativo



Interfaz del usuario

- ▶ **Interfaz del programador:**
Interfaz de la máquina extendida.
 - ▶ Mediante **llamadas al sistema**.
- ▶ **Interfaz de usuario:**
Interacción SSOO/Usuario.
 - ▶ Mediante *shell*:
 - ▶ Interfaz de **línea de mandatos** o CLI
 - ▶ **Interfaz gráfica** o GUI

Llamadas al sistema

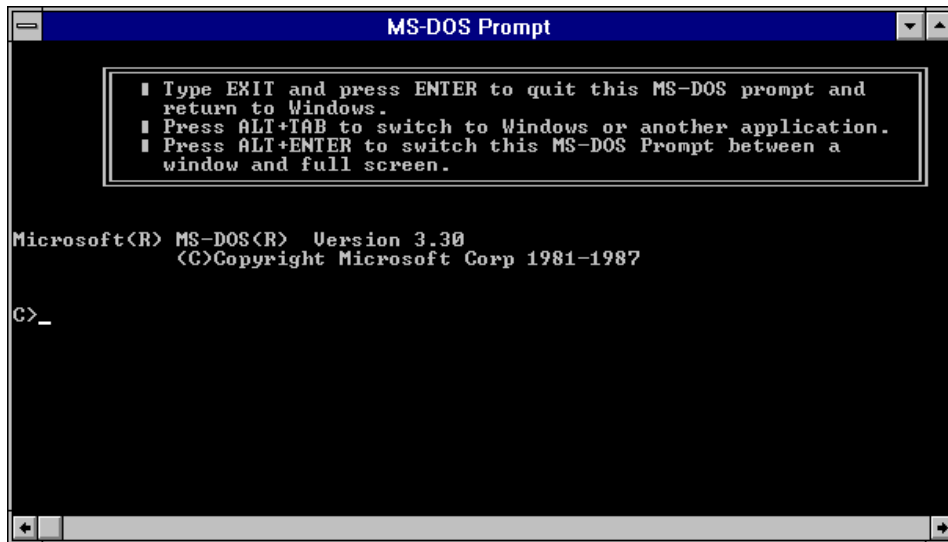
- ▶ **Interfaz del programador:**
 - ▶ llamadas al sistema
- ▶ Interfaz de usuario:
 - ▶ Interfaz de línea de mandatos o CLI
 - ▶ Interfaz gráfica o GUI

- ▶ Los programas de usuario acceden a los servicios del sistema operativo a través de llamadas al sistema.
- ▶ Son vistas por los usuarios programadores como llamadas a funciones:

`leidos = read(filedesc, buffer, nbytes) ;`

Intérprete de mandatos

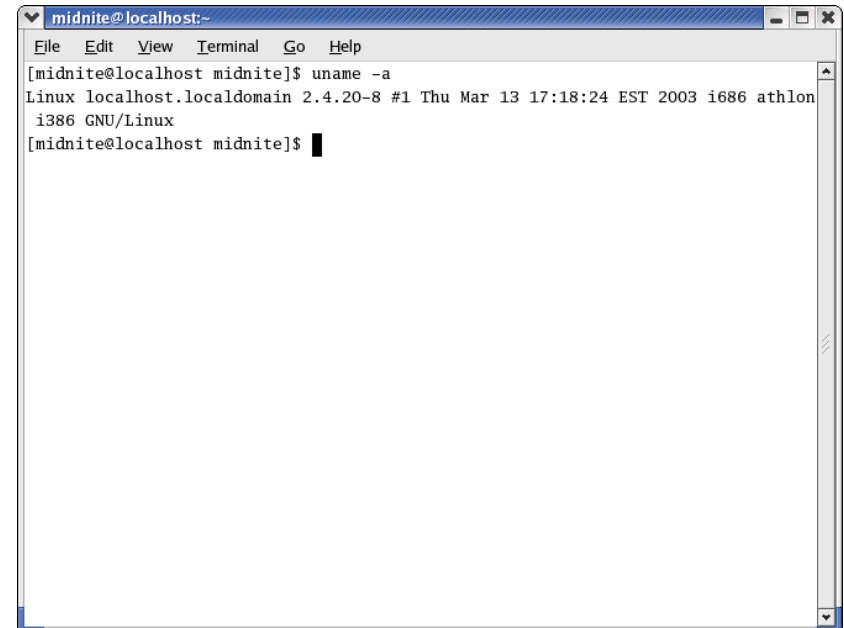
- ▶ Interfaz del programador:
 - ▶ llamadas al sistema
- ▶ Interfaz de usuario:
 - ▶ Interfaz de línea de mandatos o CLI
 - ▶ Interfaz gráfica o GUI



A screenshot of the MS-DOS Prompt window. The title bar is blue and says "MS-DOS Prompt". Inside the window, there is a text box with instructions: "Type EXIT and press ENTER to quit this MS-DOS prompt and return to Windows.", "Press ALT+TAB to switch to Windows or another application.", and "Press ALT+ENTER to switch this MS-DOS Prompt between a window and full screen." Below this, the text "Microsoft(R) MS-DOS(R) Version 3.30" and "(C)Copyright Microsoft Corp 1981-1987" is displayed. The prompt "C>_" is visible at the bottom left.



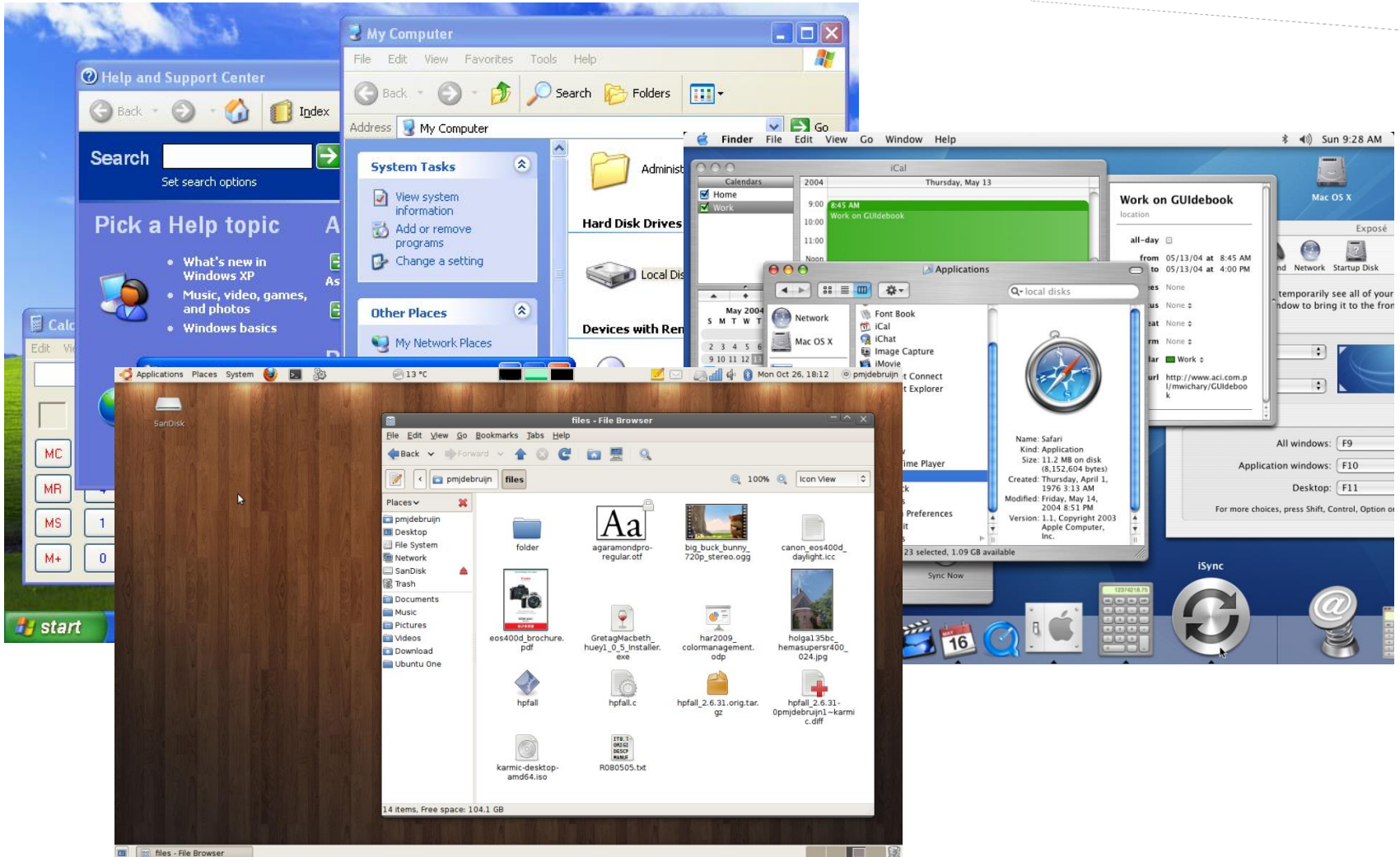
A screenshot of a terminal window titled "/bin/tcsh (tty1)". The prompt is "[localhost:~] midnite%".



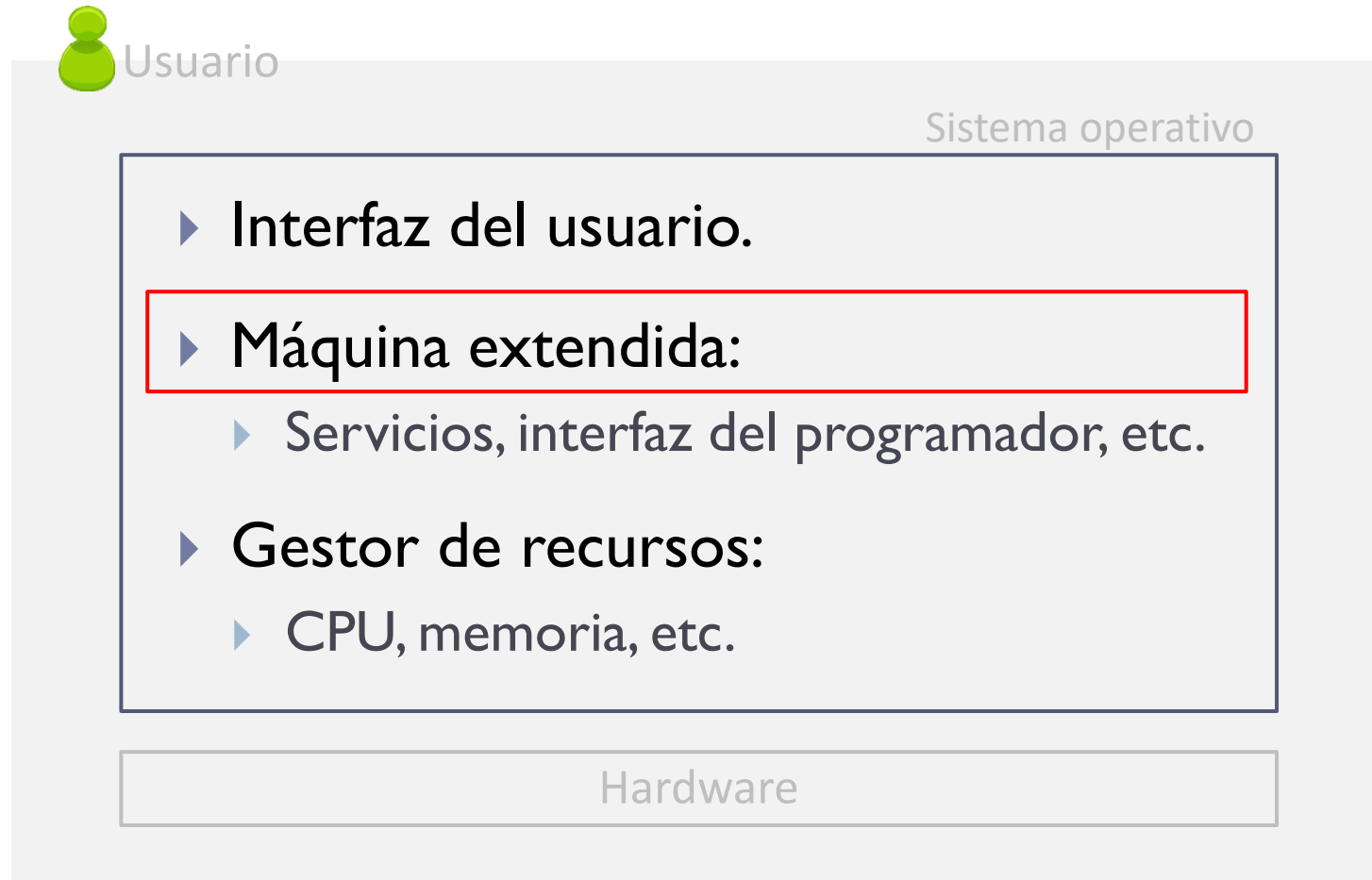
A screenshot of a Linux terminal window titled "midnite@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Go", and "Help". The terminal shows the command "uname -a" being executed, with the output: "Linux localhost.localdomain 2.4.20-8 #1 Thu Mar 13 17:18:24 EST 2003 i686 athlon i386 GNU/Linux". The prompt "midnite@localhost midnite\$" is visible at the bottom.

Interfaz gráfica

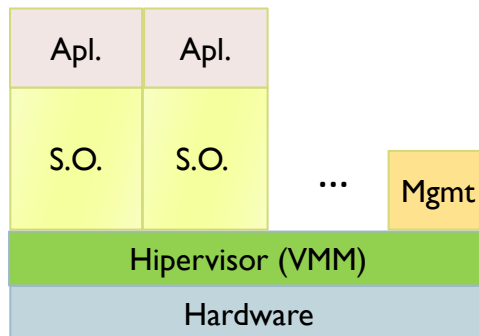
- Interfaz del programador:
 - llamadas al sistema
- Interfaz de usuario:
 - Interfaz de línea de mandatos o CLI
 - Interfaz gráfica o GUI



Funciones del sistema operativo

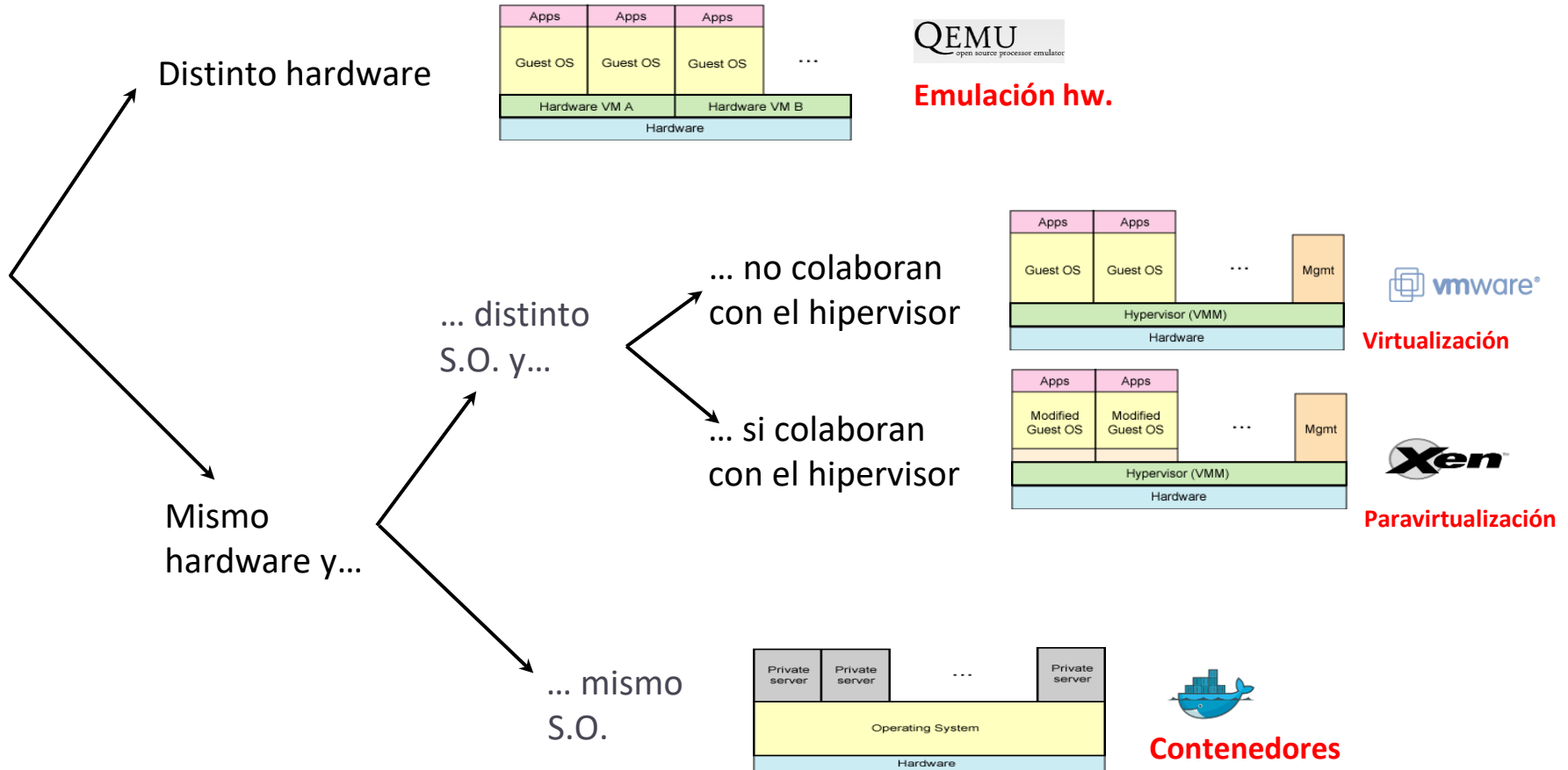


Máquinas virtuales



- ▶ El sistema operativo virtualiza ciertos elementos del hardware; ¿Por qué no **virtualizar todo**?
- ▶ **IBM** ha usado esta idea en sus mainframes desde principio de la década de los **70**.
- ▶ Un hipervisor virtualiza todo el ordenador, de manera que permite que múltiples sistemas operativos (o copias del mismo) ejecuten a la vez.
- ▶ La virtualización:
 - ▶ **[I]** supone **cierta sobrecarga**
 - ▶ **[V]** ofrece un **aislamiento excelente** entre sistemas y la flexibilidad en la reserva de recursos lo que **mejora el coste**, especialmente en «granjas»

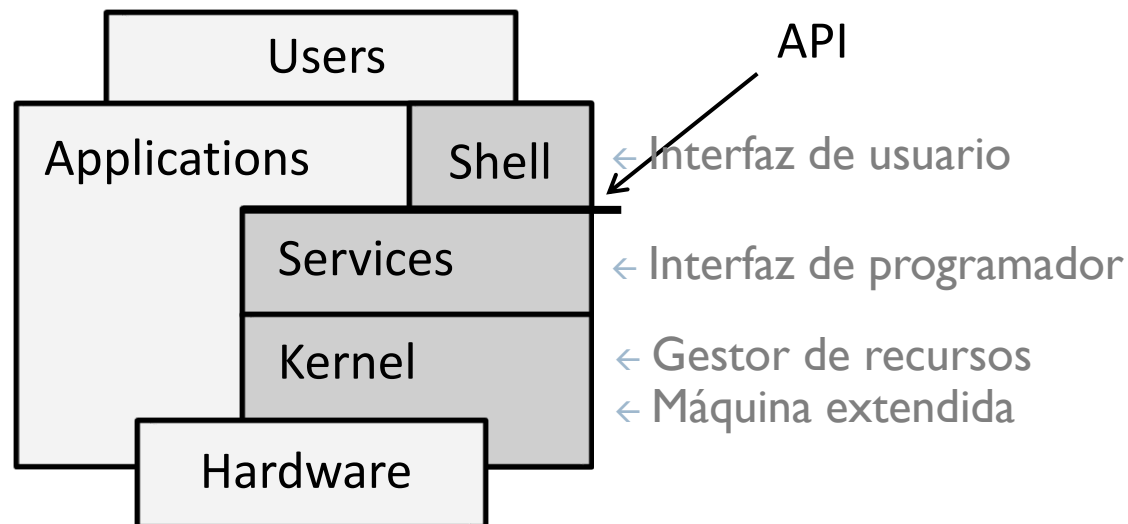
Máquinas virtuales



Contenidos

1. ¿Qué es un sistema operativo?
2. Evolución de los sistemas operativos.
3. Características.
4. Tareas de un sistema operativo.
5. **Funcionamiento básico.**
6. Estructura del sistema operativo.
7. Arranque del sistema operativo.

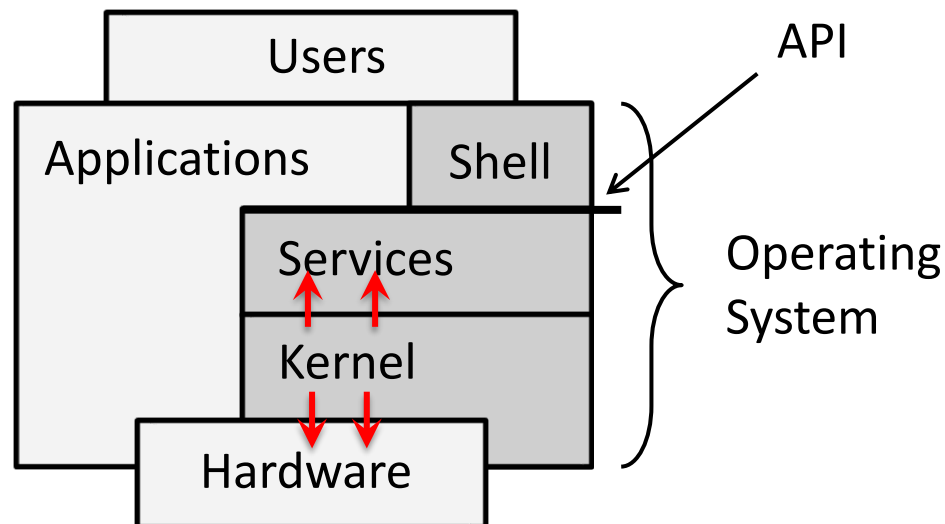
Niveles del sistema operativo



Contextos donde está presente el SO

arranque

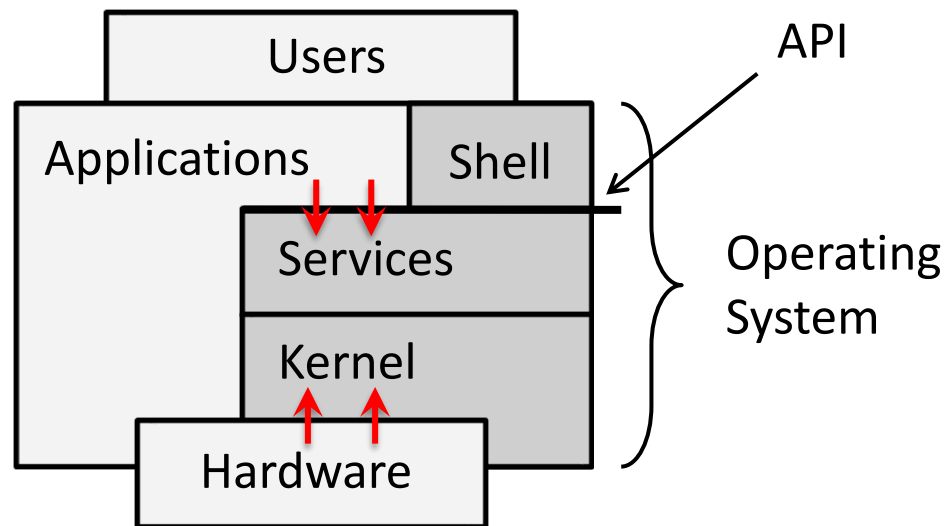
- ▶ Durante el arranque, el SO se encarga de **inicializaciones**:
 - ▶ Del hardware: enciende e inicializa el hardware
 - ▶ De los procesos: se encarga de inicializar las estructuras de datos internas del SO y de ejecutar procesos en el orden apropiado



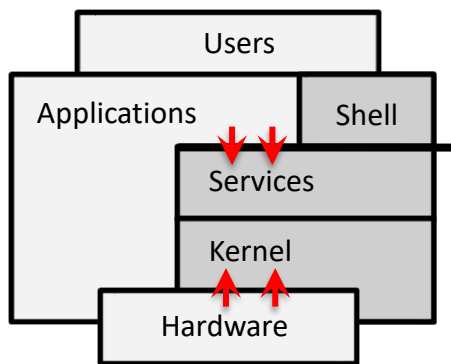
Contextos donde está presente el SO

tratamiento de eventos

- ▶ Tras arrancar, el SO se encarga de **atender** a las **peticiones**:
 - ▶ Del hardware: interrupción y excepción (interrupción de la CPU)
 - ▶ De las aplicaciones: llamada al sistema



¿Cómo es el mecanismo para tratar eventos?

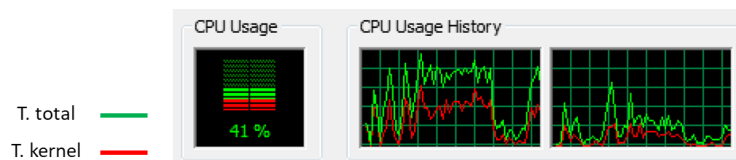


► Atención a **peticiones**:

- **Asíncronas** (interrupciones hardware)
- **Síncronas** (llamadas al sistema de las aplicaciones) y (excepciones)

► **Separación segura** de aplicaciones y SO (“**Blindar**” el SO):

- Espacios de memoria separados.
- CPU con al menos 2 modos de ejecución:
 - **Modo usuario**: Ejecución de procesos de usuario.
 - **Modo núcleo**: Ejecución del núcleo del SO.



Programación orientada a eventos

aspectos generales

```
int main ( ... )  
{  
    ...  
    On (event1, handler1) ;  
    ...  
}
```

1) Asociar el manejador
(handler1) al evento

Programación orientada a eventos

aspectos generales

```
void handler1 ( ... )  
{  
}
```

2) Codificar la función
manejador que tratará el evento

```
int main ( ... )  
{  
    ...  
    On (event1, handler1) ;  
    ...  
}
```

1) Asociar el manejador
(handler1) al evento

Programación orientada a eventos

aspectos generales

```
int global1;  
...
```

3) Para comunicar funciones,
se usa variables globales

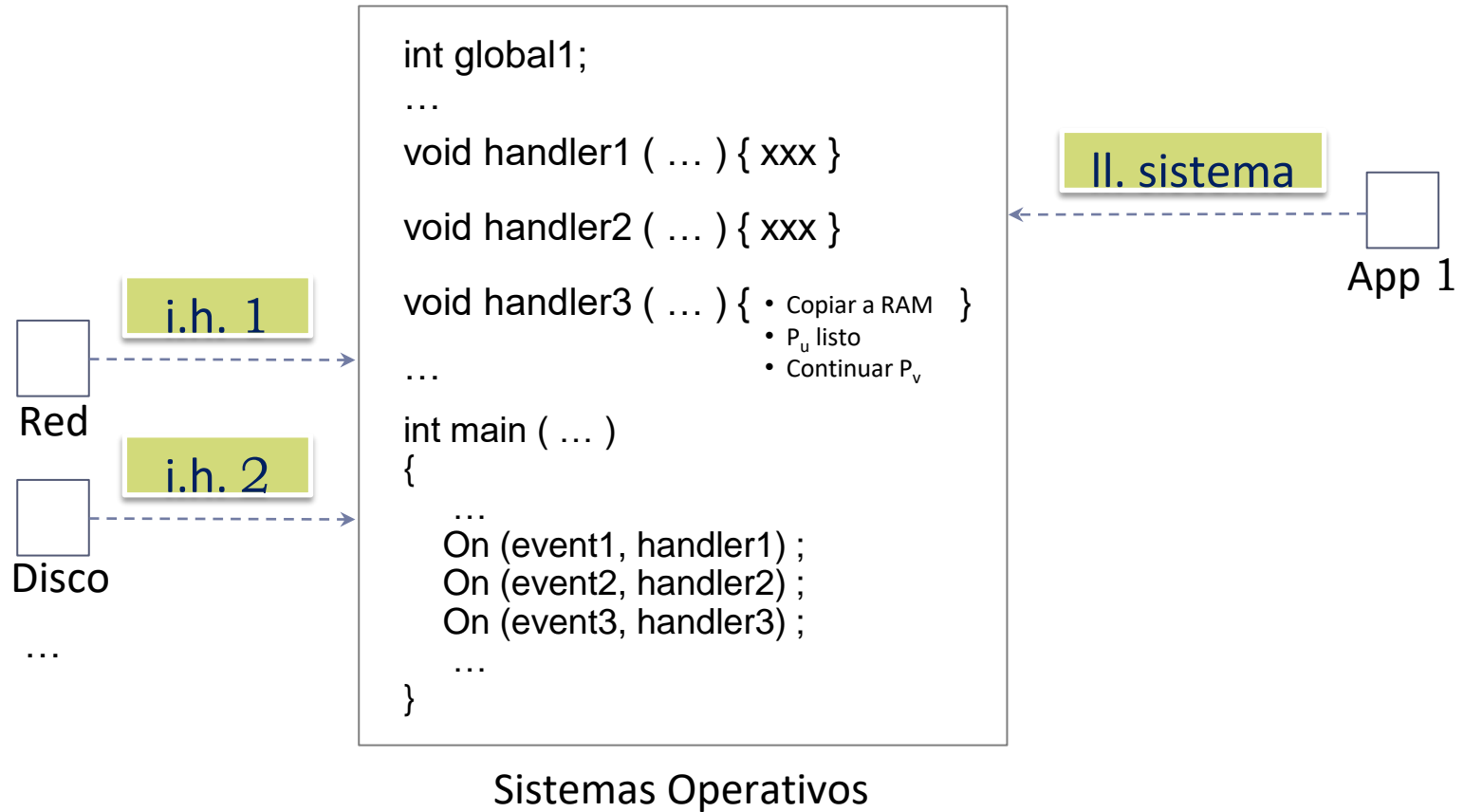
```
void handler1 ( ... )  
{  
}
```

2) Codificar la función
manejador que tratará el evento

```
int main ( ... )  
{  
    ...  
    On (event1, handler1) ;  
    ...  
}
```

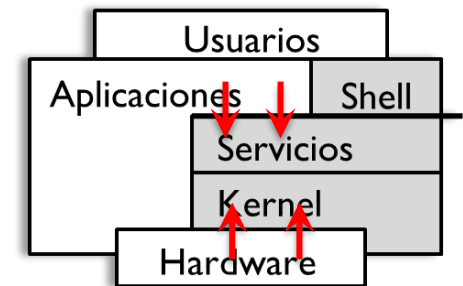
1) Asociar el manejador
(handler1) al evento

Programación orientada a eventos en el sistema operativo



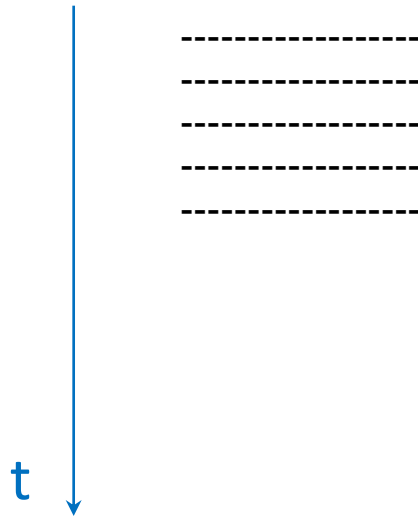
Programación orientada a eventos en el sistema operativo (+ blindar)

- Cuando un hardware necesita atención por parte del sistema operativo:
 - Genera una interrupción cuya rutina de tratamiento es parte del SO.
- Cuando un proceso necesita un servicio lo solicita al sistema operativo:
 - Llama función de biblioteca que hace una llamada al sistema.
 - Se genera una interrupción por software y el sistema operativo entra en ejecución para realizar la función solicitada por registros/pila.



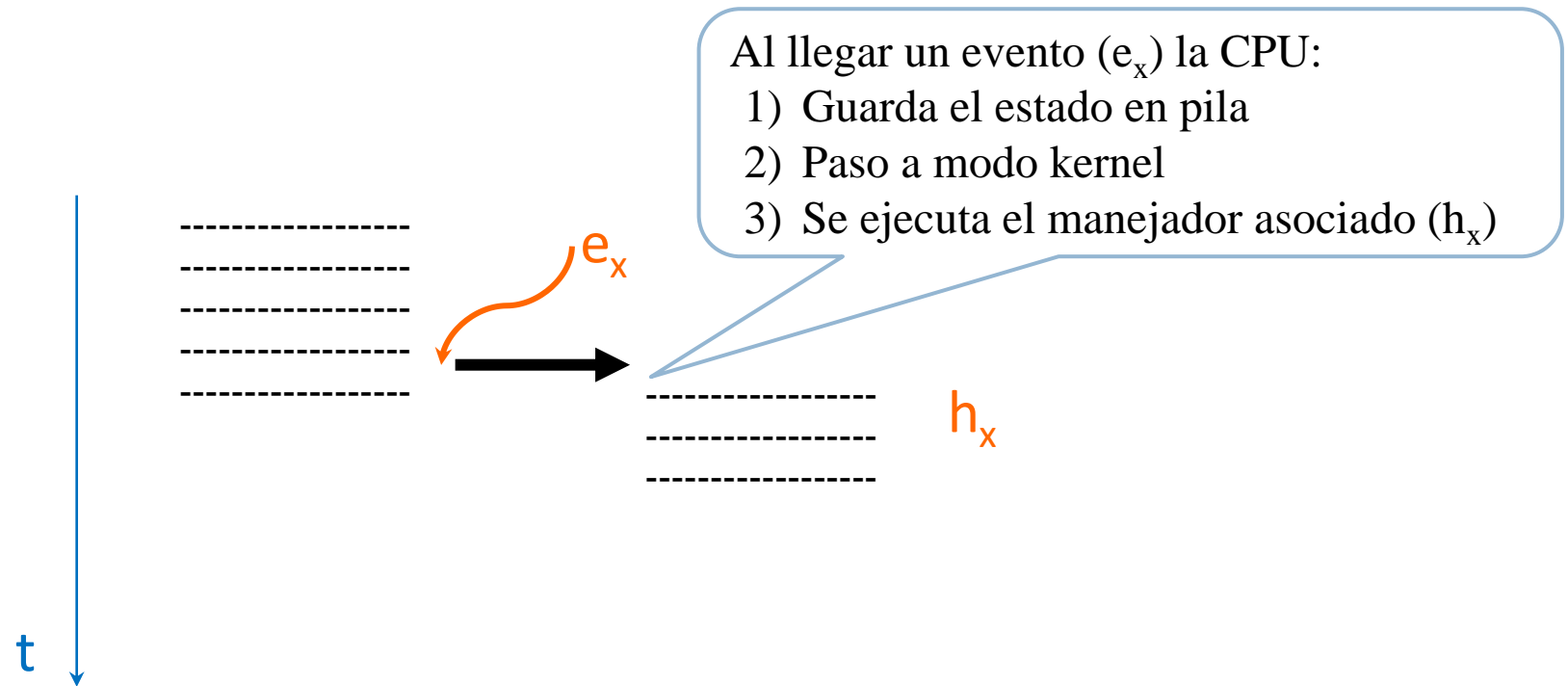
Interrupción hardware (asíncrona)

ejecución (general)



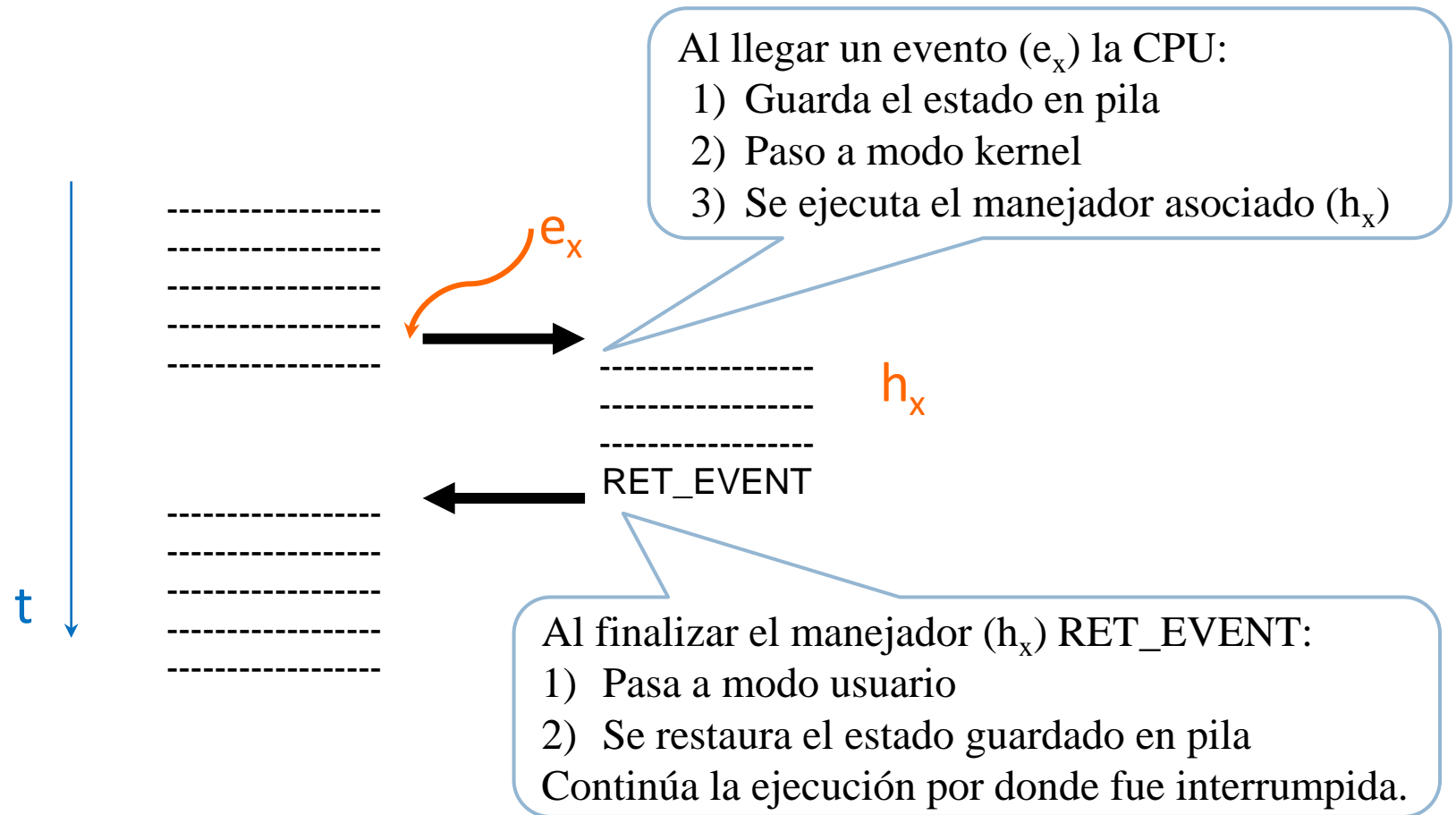
Interrupción hardware (asíncrona)

ejecución (general)



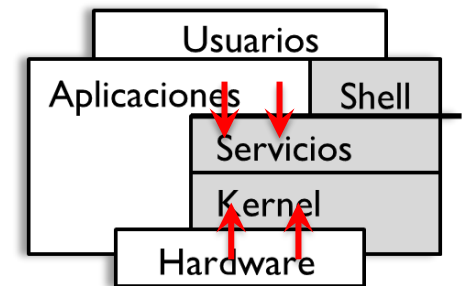
Interrupción hardware (asíncrona)

ejecución (general)



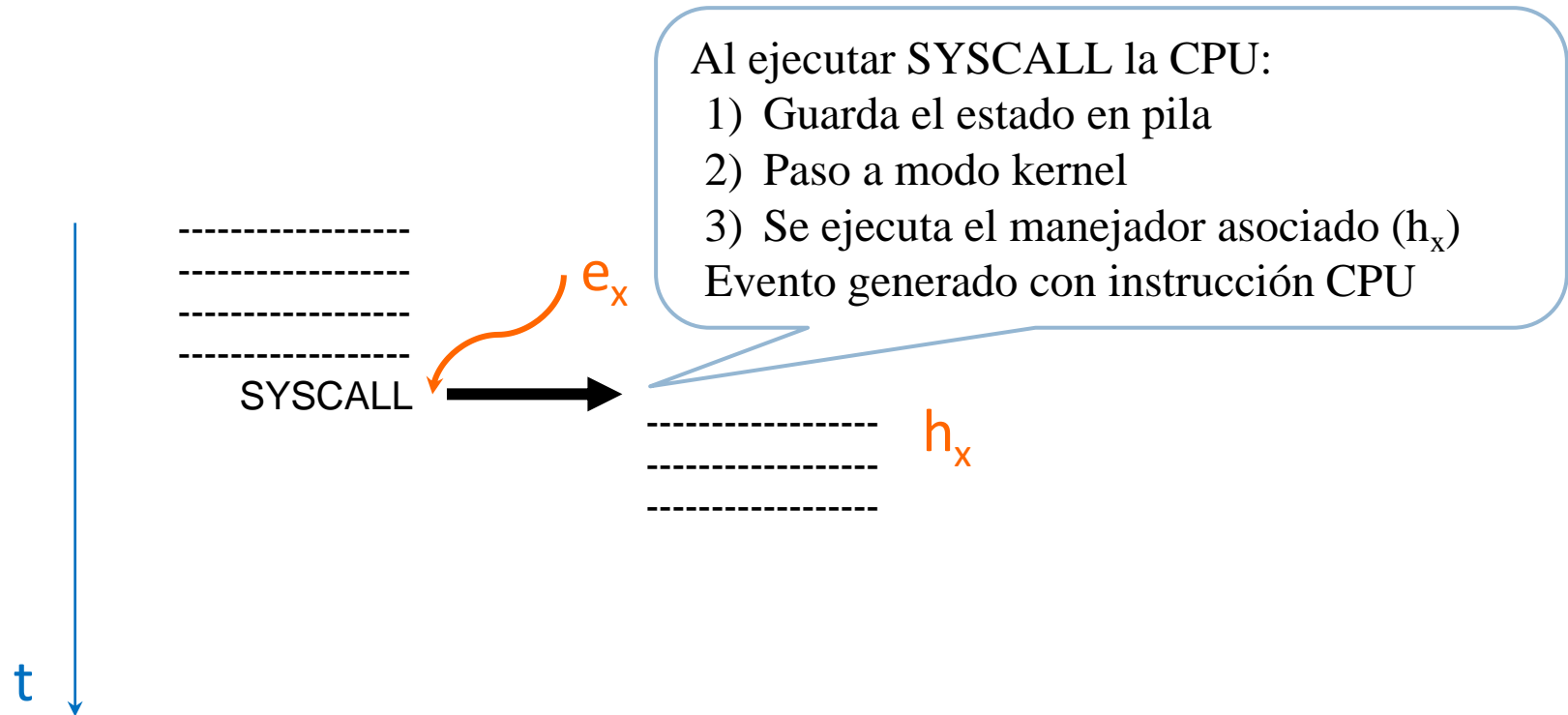
Programación orientada a eventos en el sistema operativo (+ blindar)

- Cuando un hardware necesita atención por parte del sistema operativo:
 - ▣ Genera una interrupción cuya rutina de tratamiento es parte del SO.
- Cuando un proceso necesita un servicio lo solicita al sistema operativo:
 - ▣ Llama función de biblioteca que hace una llamada al sistema.
 - ▣ Se genera una interrupción por software y el sistema operativo entra en ejecución para realizar la función solicitada por registros/pila.



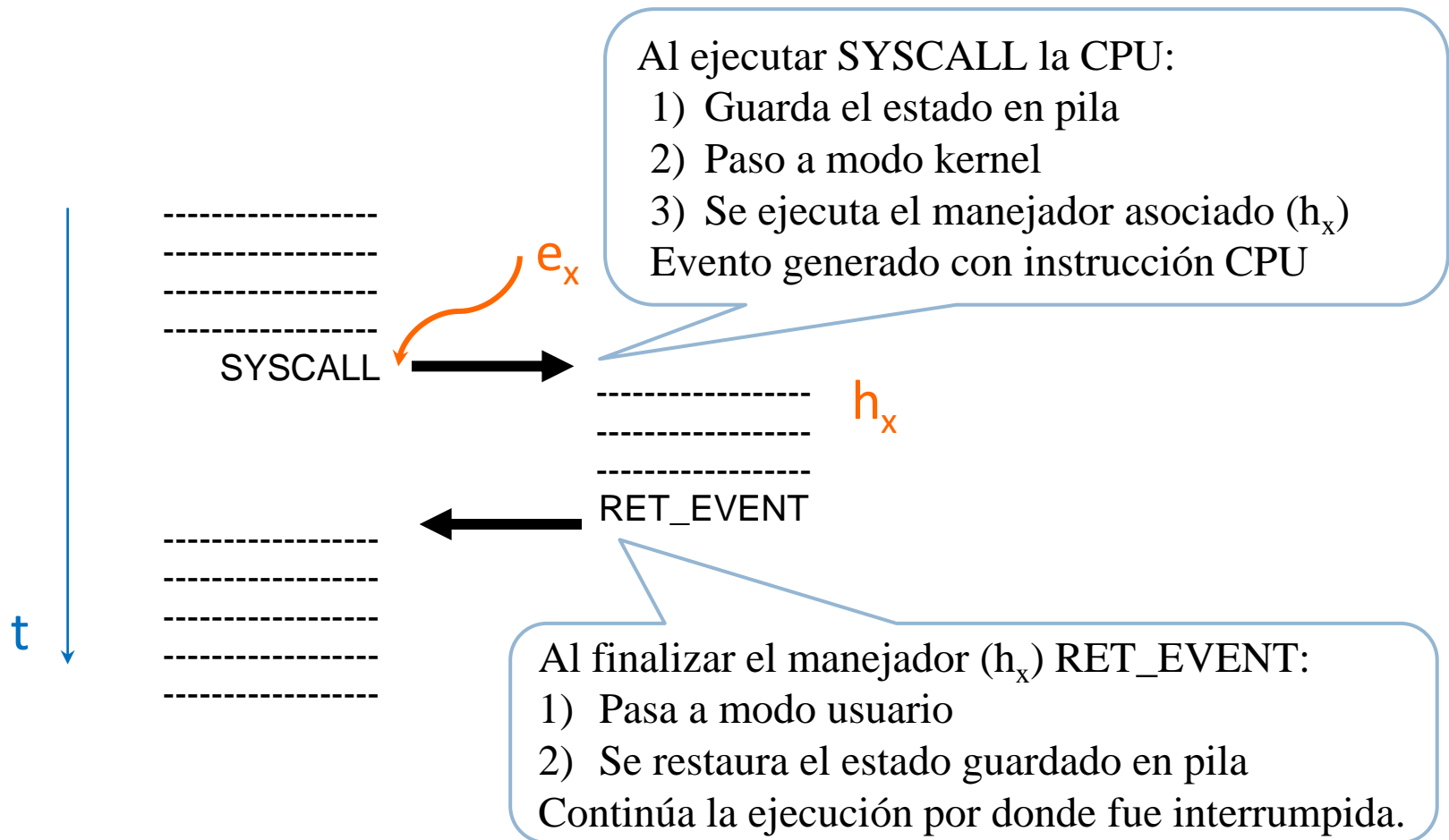
Ejecución petición de servicio

ejecución (general)



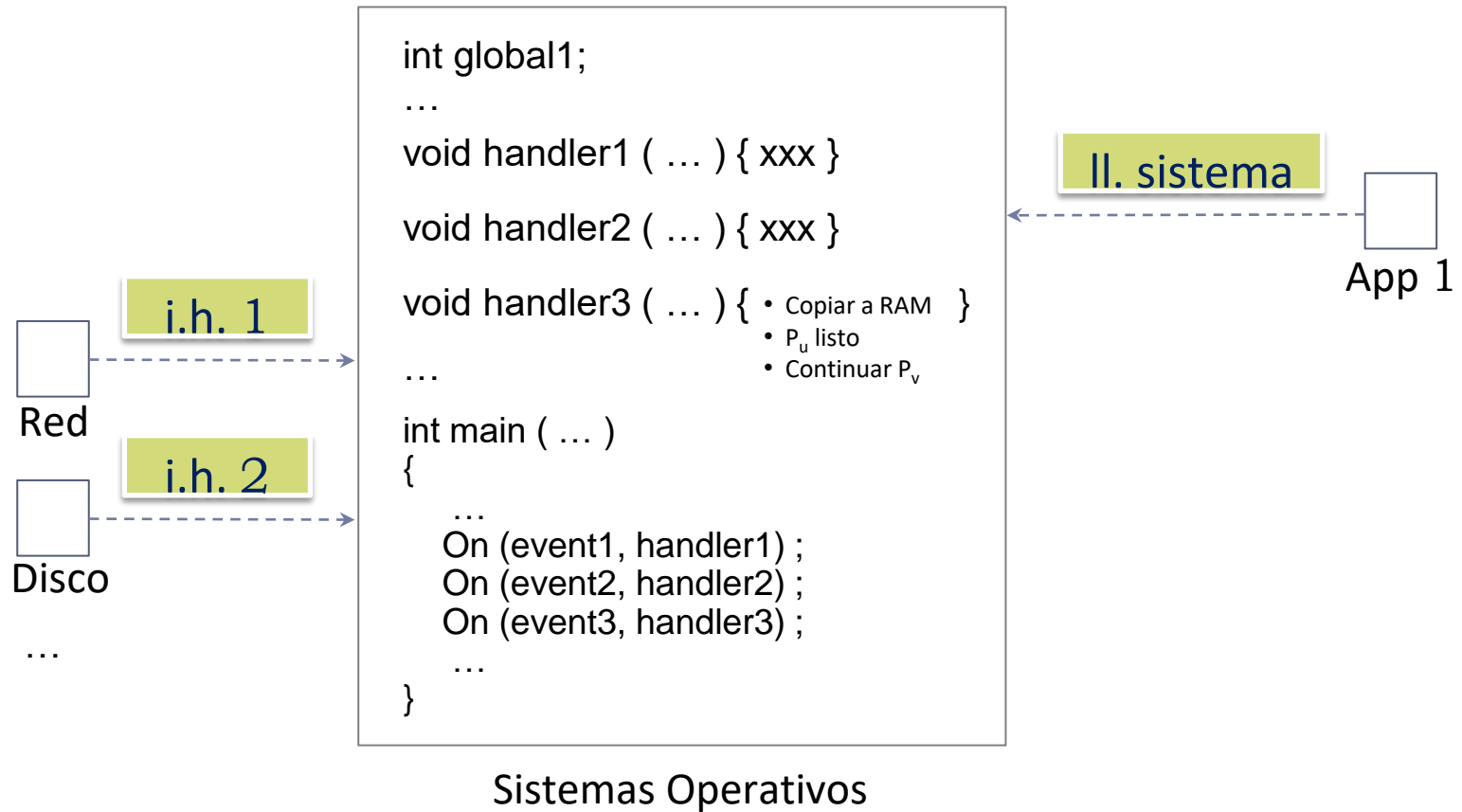
Ejecución petición de servicio

ejecución (general)



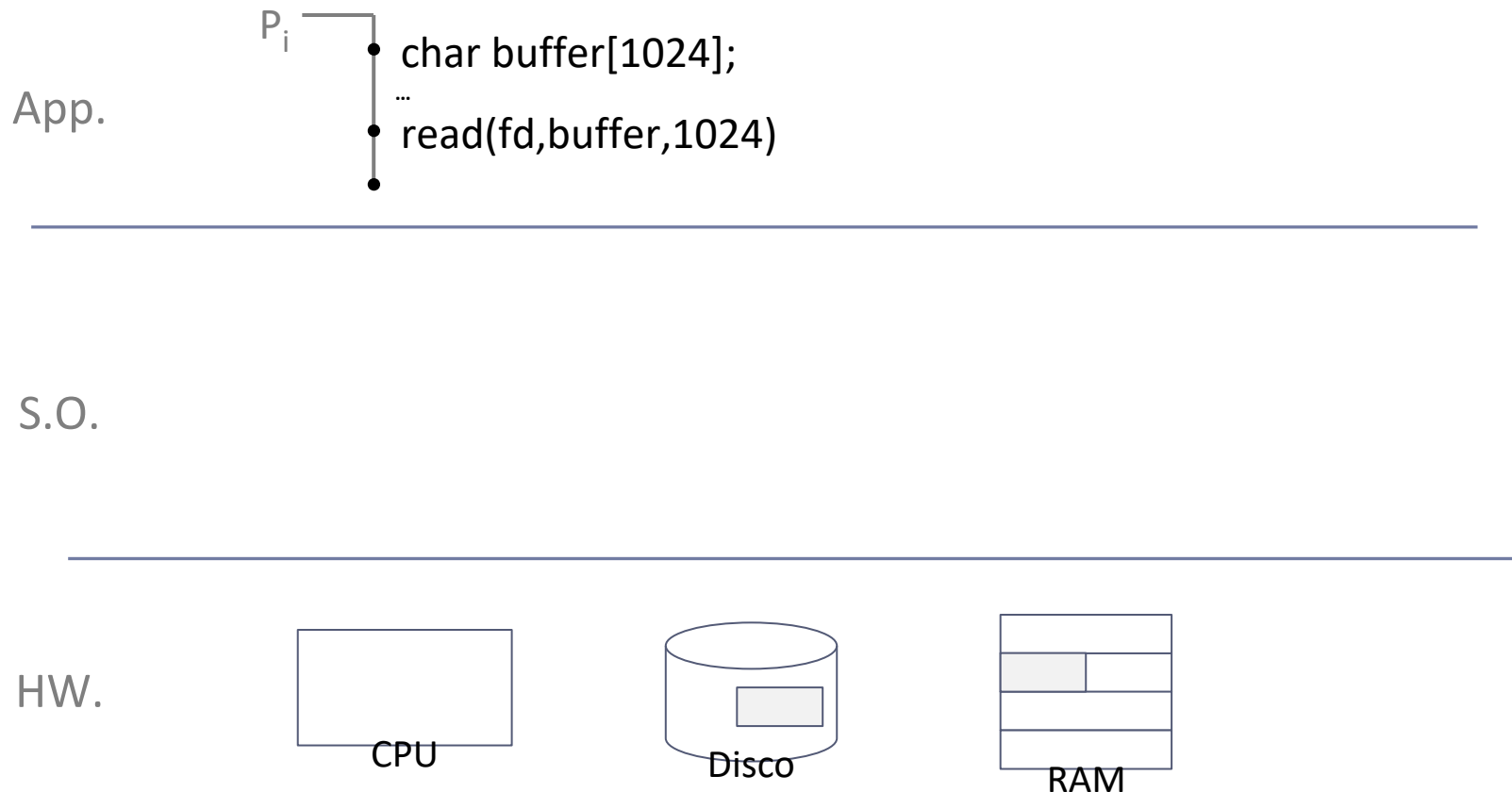
Programación orientada a eventos

resumen



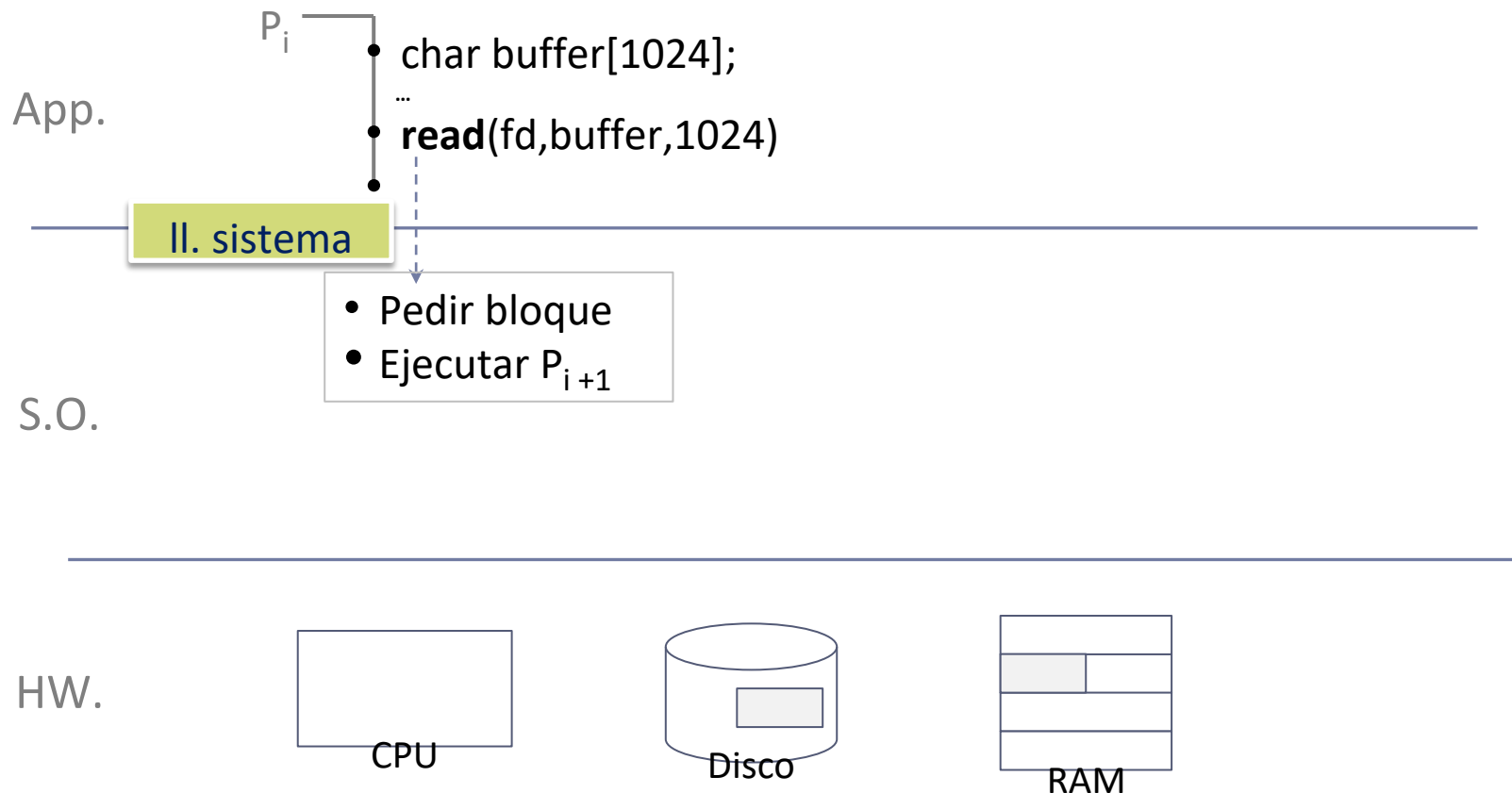
Ejecución asíncrona

ejemplo simplificado



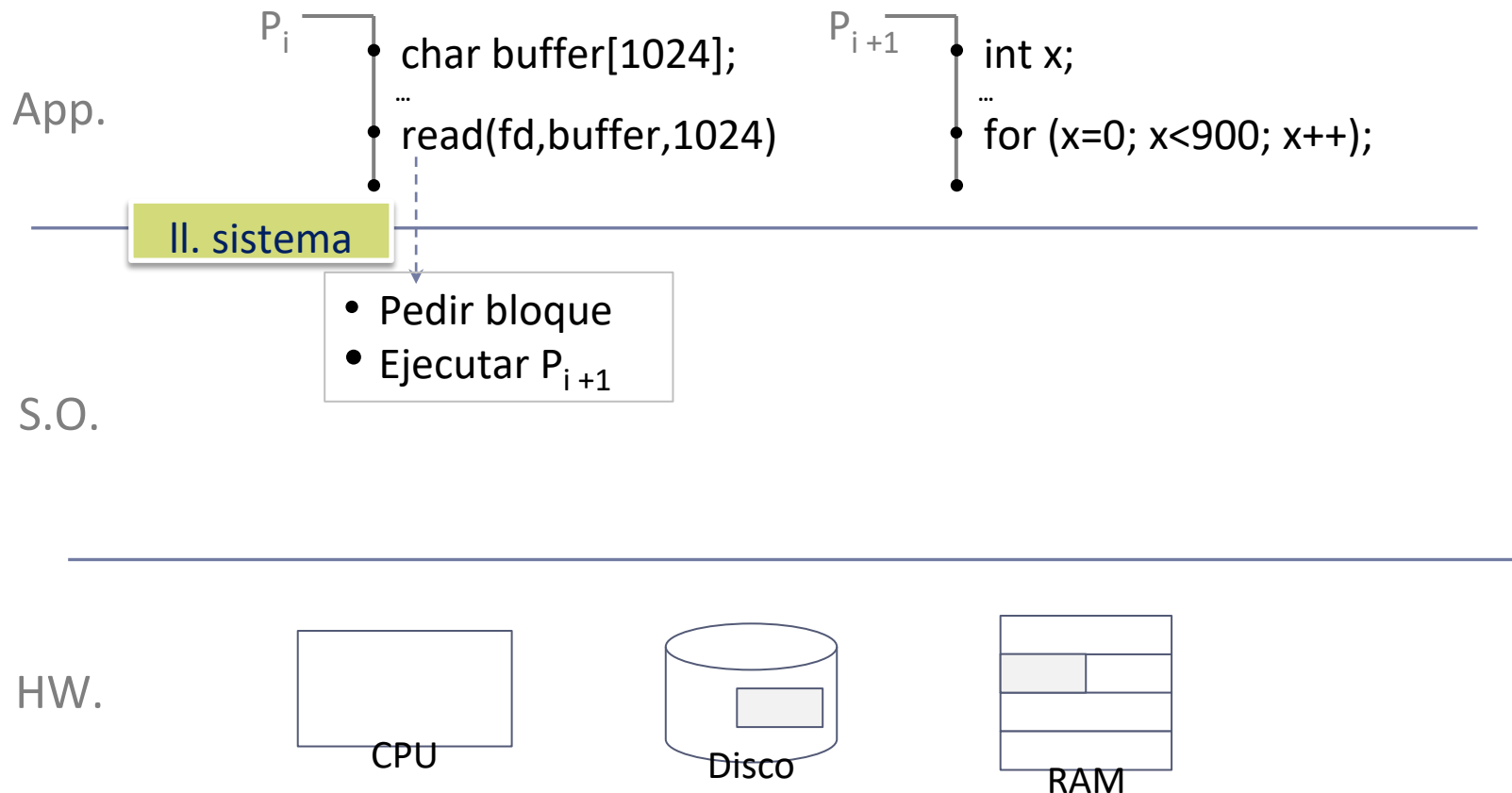
Ejecución asíncrona

ejemplo simplificado



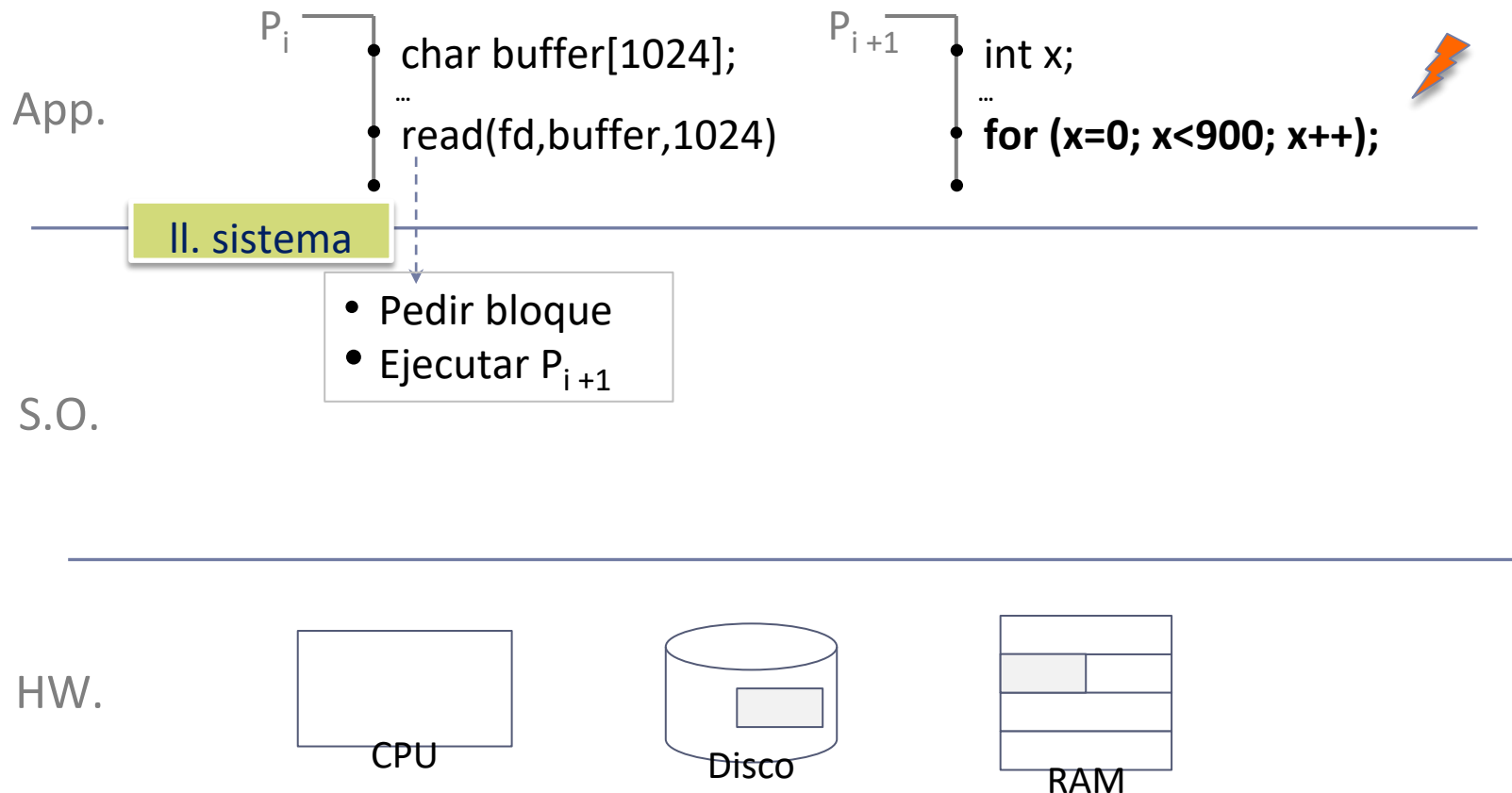
Ejecución asíncrona

ejemplo simplificado



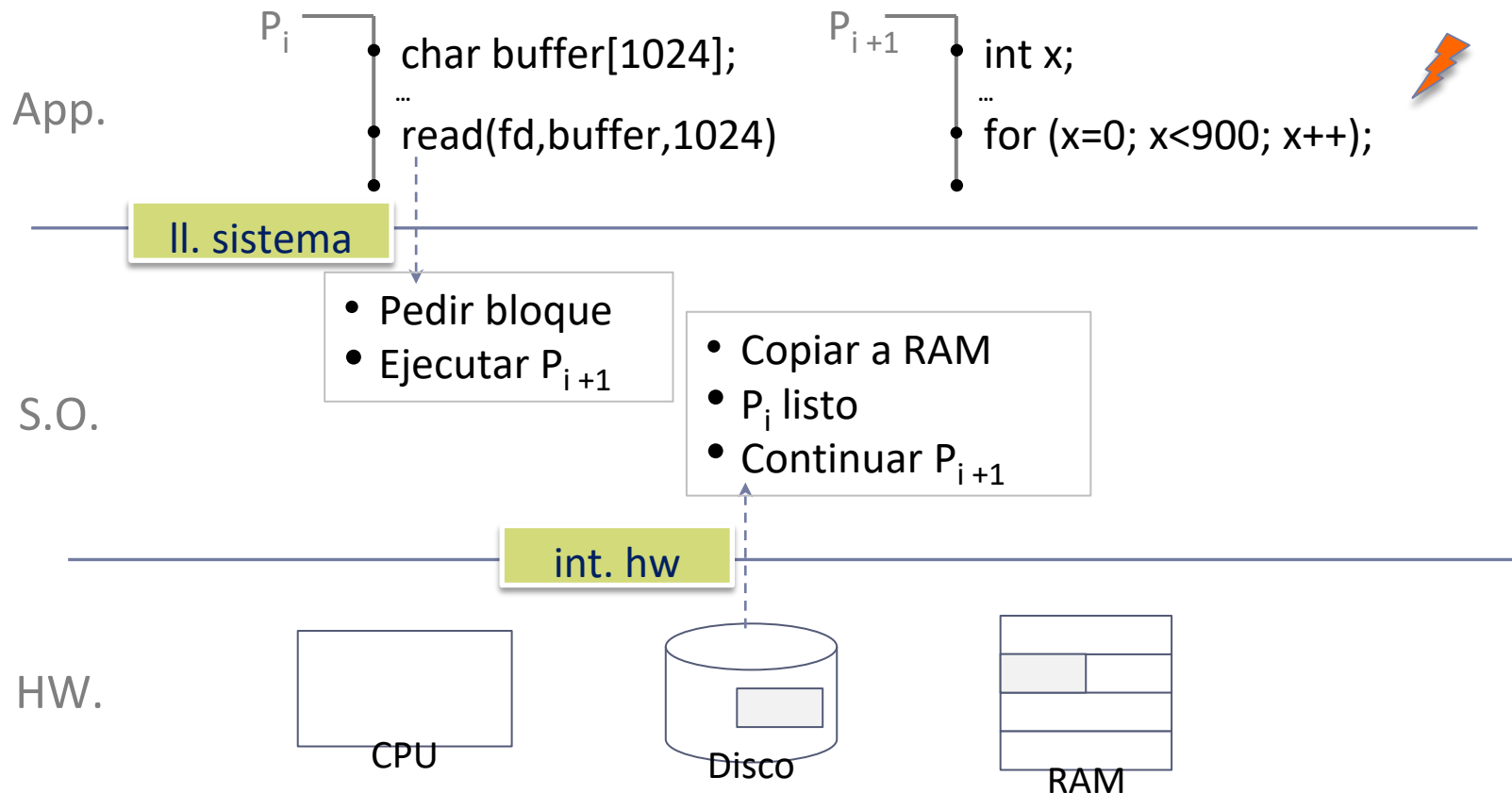
Ejecución asíncrona

ejemplo simplificado



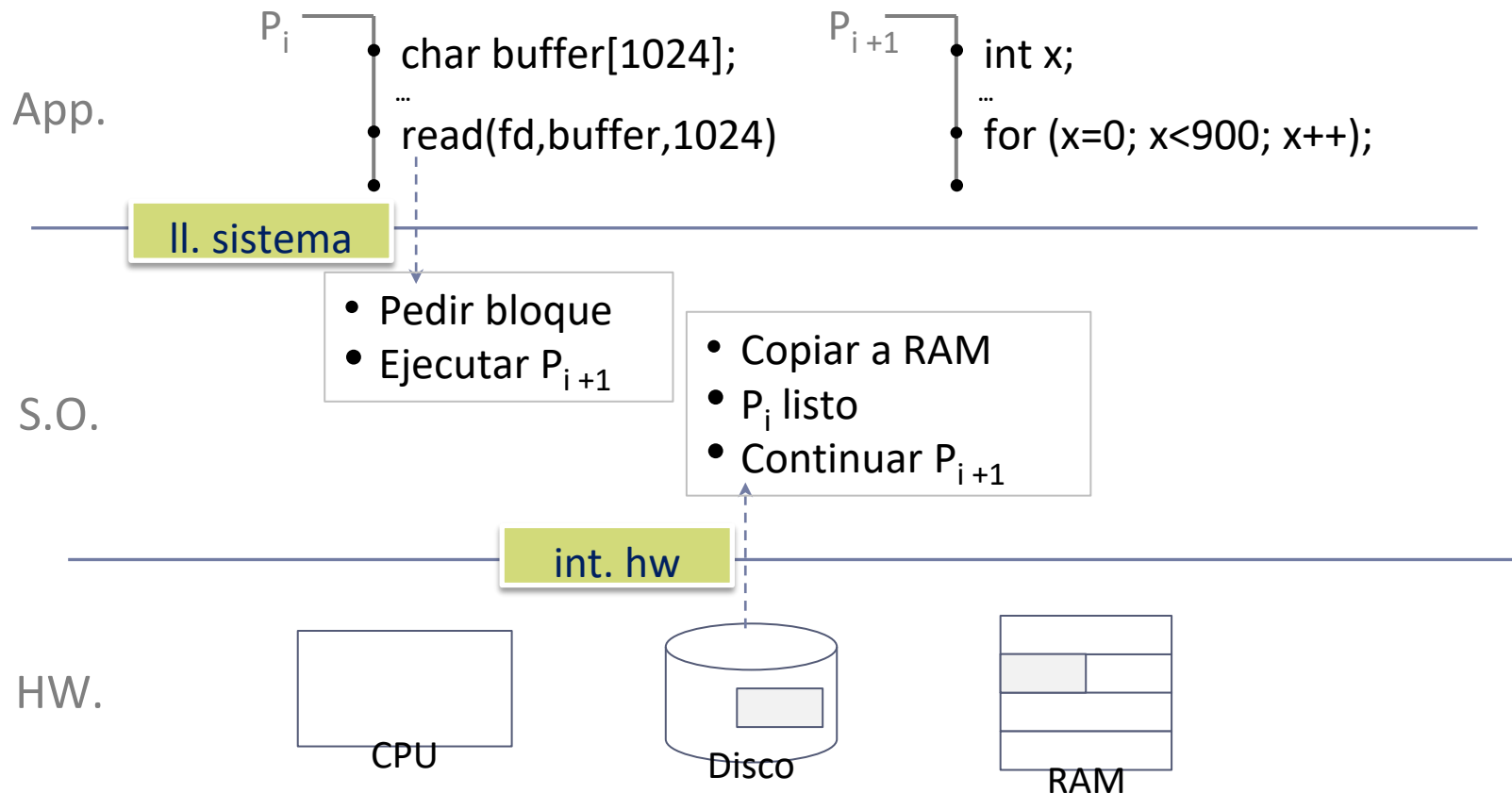
Ejecución asíncrona

ejemplo simplificado



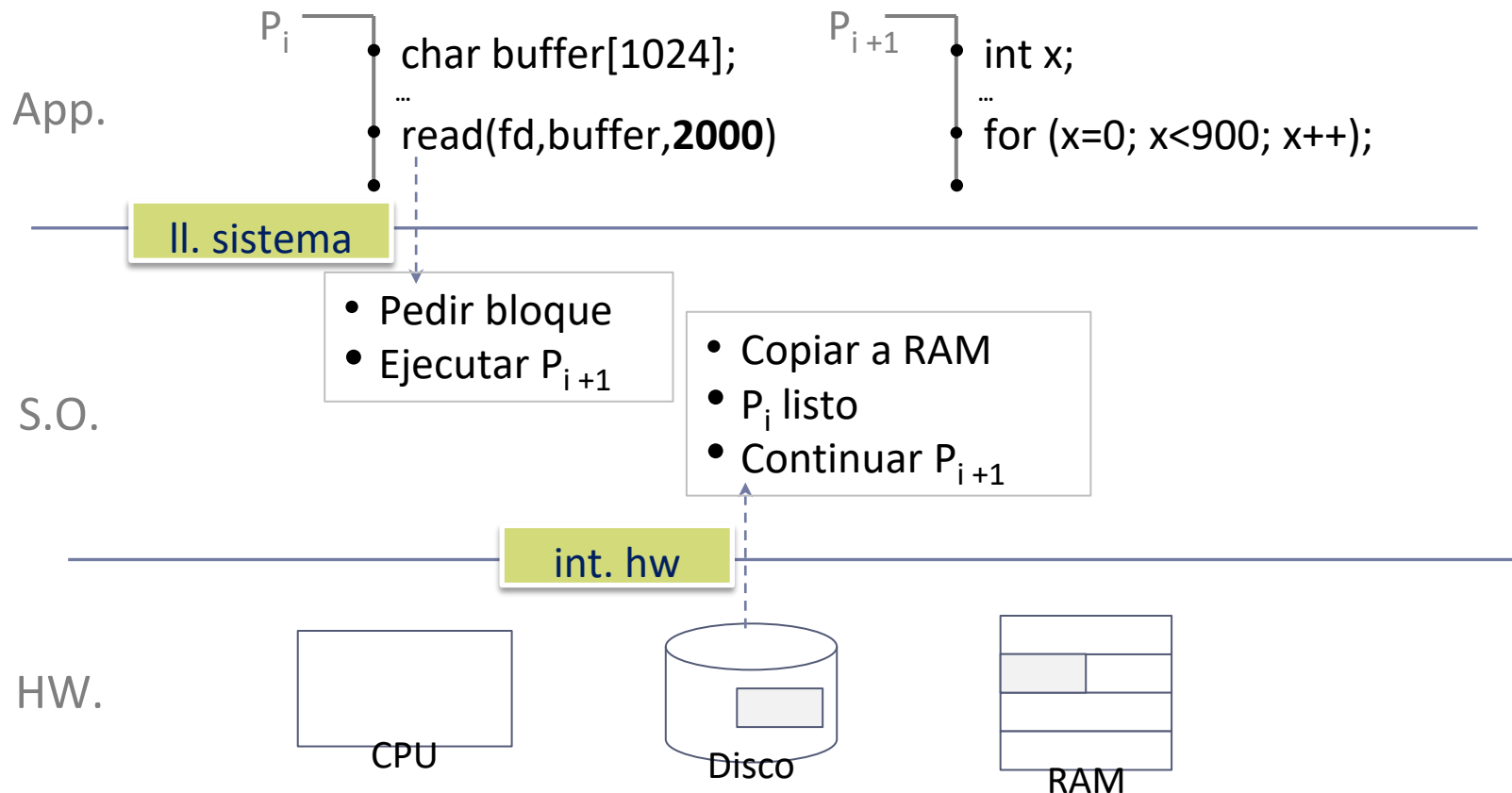
Ejecución asíncrona

ejemplo simplificado



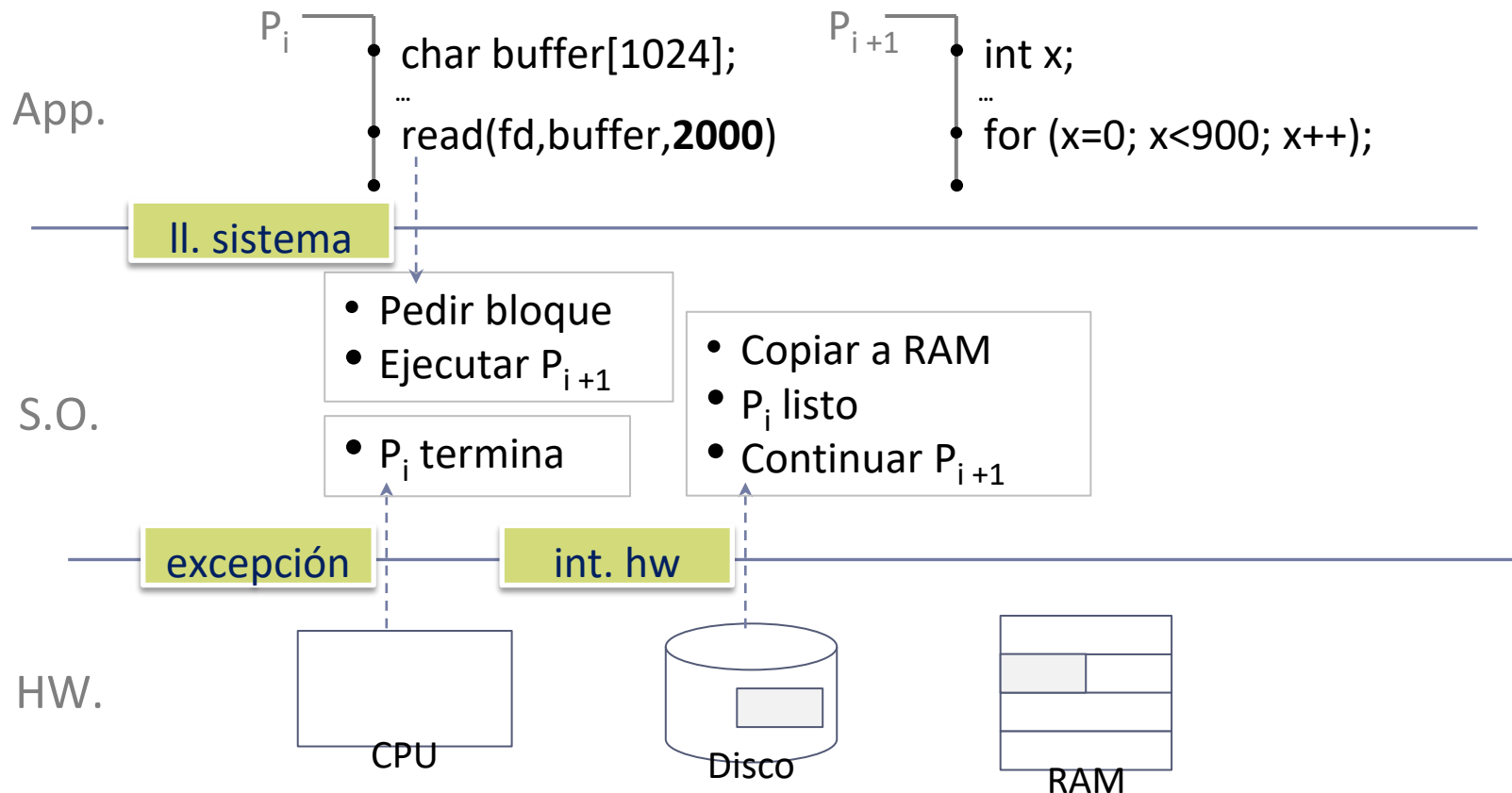
Ejecución asíncrona

ejemplo simplificado



Ejecución asíncrona

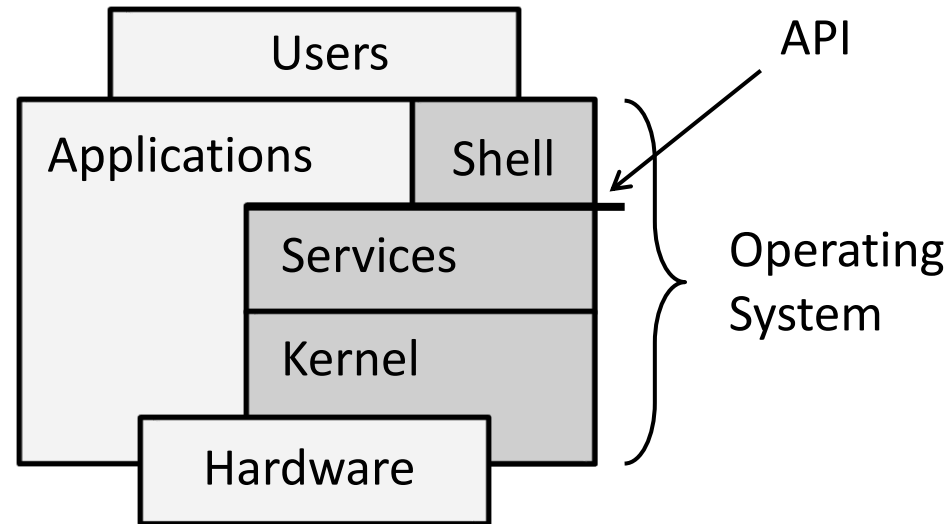
ejemplo simplificado



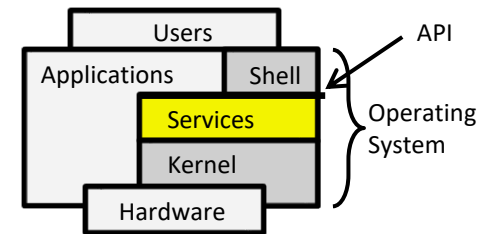
Contenidos

1. ¿Qué es un sistema operativo?
2. Evolución de los sistemas operativos.
3. Características.
4. Tareas de un sistema operativo.
5. Funcionamiento básico.
6. **Estructura del sistema operativo.**
7. Arranque del sistema operativo.

Niveles del sistema operativo



Los servicios



- ▶ En general, hay 5 tipos de llamada al sistema:
 - ▶ Control de procesos
 - ▶ Manipulación de archivos
 - ▶ Manipulación de dispositivos
 - ▶ Monitorización
 - ▶ Comunicación
- ▶ Interfaces propietarias vs. estándar:
 - ▶ Interfaz POSIX
 - ▶ Interfaz Win32

Estructura del Sistema Operativo

- ▶ Tipos generales de estructuras de S.O.:

- ▶ Monolítico
- ▶ Estructurados
 - ▶ En capas
 - ▶ Microkernel

- ▶ Estructuras específicas disponibles:

- ▶ Módulos
- ▶ Máquinas virtuales

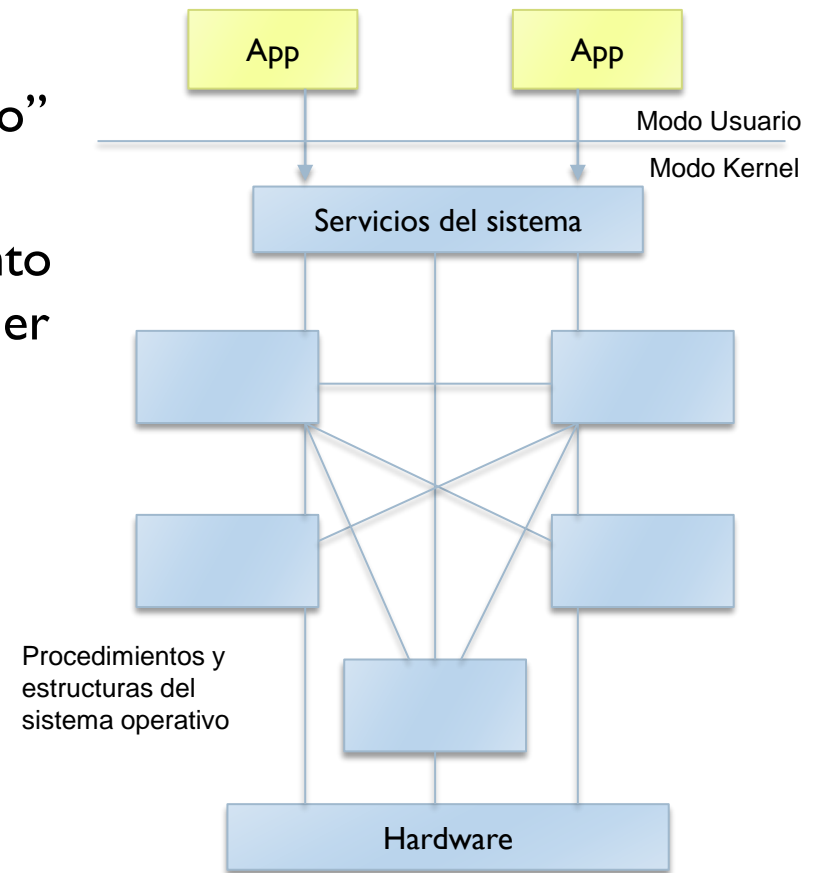
- ▶ Estructuras en algunos sistemas operativos:

- ▶ Linux
- ▶ Windows 2000

Estructura del Sistema Operativo

Monolítico (macrokernel)

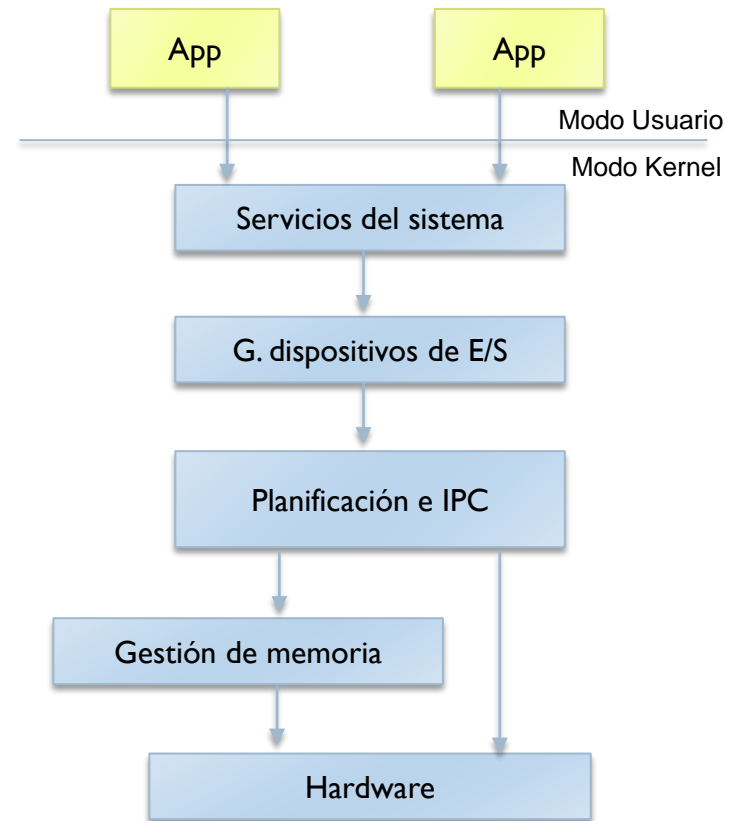
- ▶ Código enlazado como un único ejecutable, se ejecuta en modo “núcleo” en único espacio de direcciones.
- ▶ No estructurado: desde cualquier punto del código se puede acceder a cualquier variable o función de otra parte del núcleo (*kernel*)
- ▶ **[I]** muy difícil de mantener, muy sensible a errores
- ▶ Ejemplo:
 - ▶ MS-DOS, Linux



Estructura del Sistema Operativo

Por capas

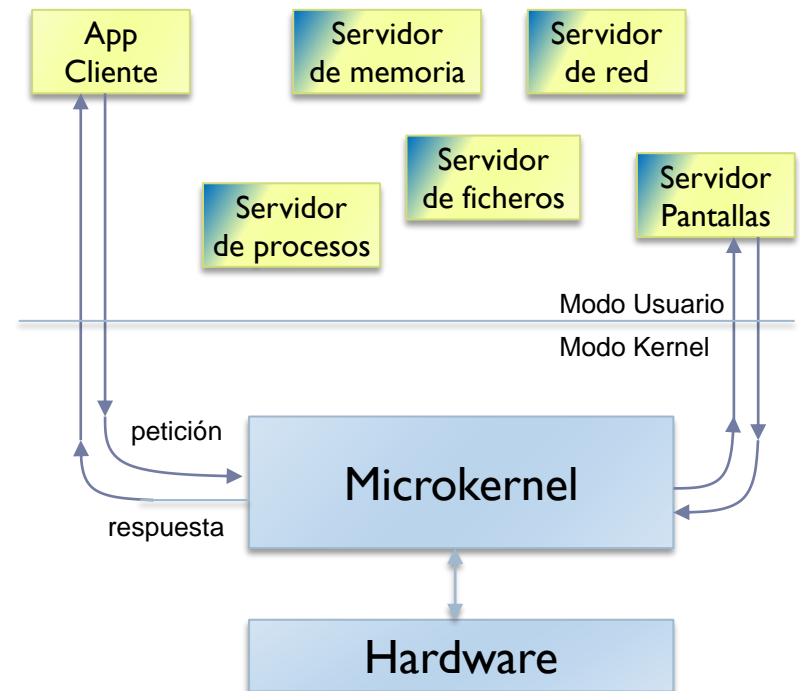
- ▶ Binario monolítico aunque codificado estructurado de forma lógica en capas.
- ▶ Cada capa proporciona acceso únicamente a la interfaz de niveles inferiores.
- ▶ **[I]** varias llamadas entre capas, distribución de funcionalidad.
- ▶ Ejemplo:
 - ▶ **THE** (Dijkstra)
 - ▶ **Multics**, que añadió a la noción de capa la idea de anillos de privilegios



Estructura del Sistema Operativo

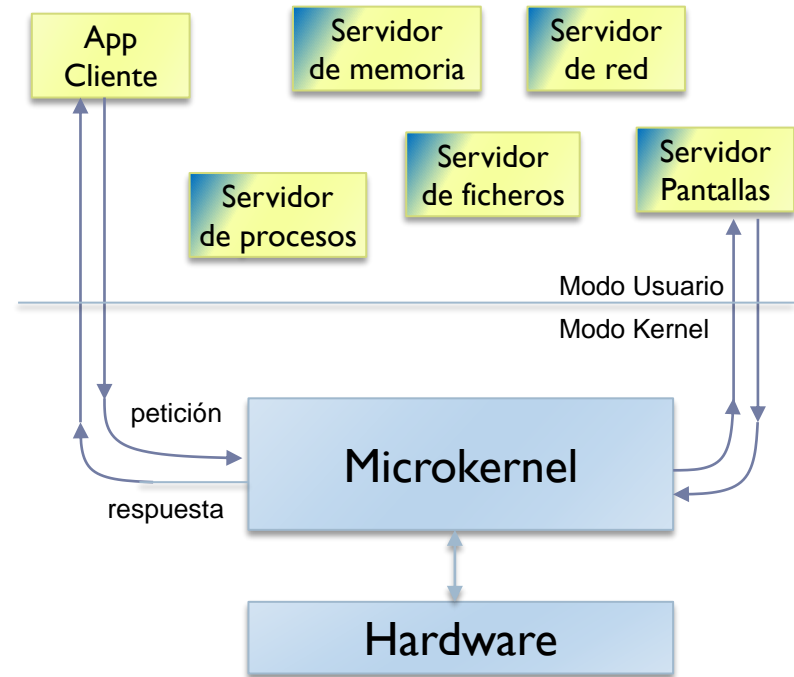
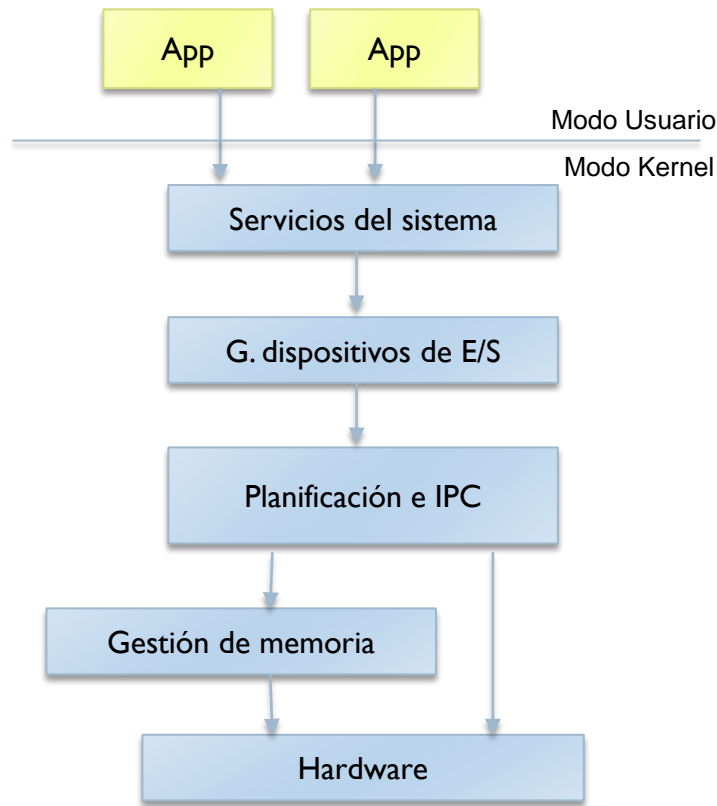
Microkernel

- ▶ Además de estructurado, los principales componentes se ejecutan como procesos servidores, fuera del kernel.
- ▶ El microkernel tiene:
 - ▶ Planificación y gestión de procesos.
 - ▶ Gestión de memoria virtual básica.
 - ▶ Comunicación entre procesos básica.
- ▶ [V/I] Flexible, seguro, extensible pero sobrecarga en ejecución.
- ▶ Ejemplo:
 - ▶ Match, QNX, Minix, L4, etc.



Estructura del Sistema Operativo

Microkernel vs Capas/Monolítico



Microkernel:

- ▶ **[v]** más fiable y estructurado
- ▶ **[!]** pérdida de rendimiento
 - ▶ cambio de modo kernel a usuario son costosos
- ▶ **[!]** posibles problemas de seguridad
 - ▶ Ej.: *man in the middle*

Estructura del Sistema Operativo

- ▶ Tipos generales de estructuras de S.O.:

- ▶ Monolítico
- ▶ Estructurados
 - ▶ En capas
 - ▶ Microkernel

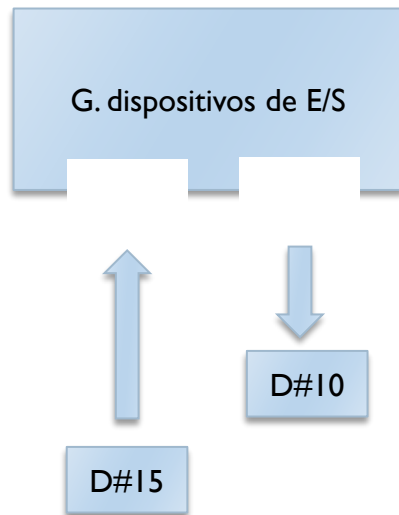
- ▶ Estructuras específicas disponibles:

- ▶ Módulos
- ▶ Máquinas virtuales

- ▶ Estructuras en algunos sistemas operativos:

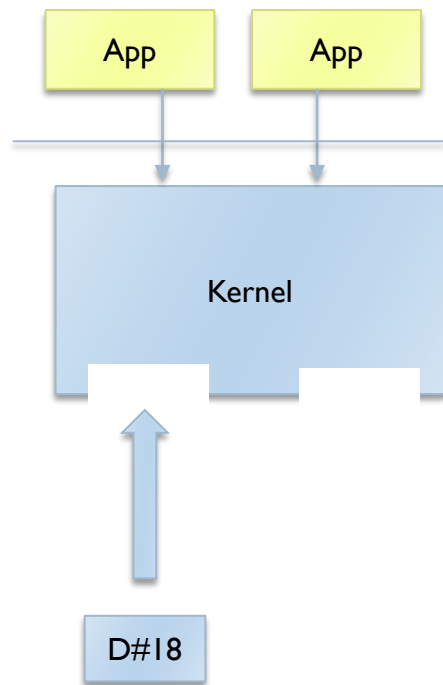
- ▶ Linux
- ▶ Windows 2000

Módulos (1 / 2)



- ▶ Los primeros *kernels* tenían que:
 - ▶ Incluir código para **todos los posibles dispositivos**
 - ▶ Ser **recompilados** por cada nuevo dispositivo añadido
- ▶ Los módulos inicialmente se desarrollaron para permitir la **inclusión condicional de controladores de dispositivos** (*drivers*)
 - ▶ Los módulos ofrecen **añadir dinámicamente código de un driver pre-compilado**.
 - ▶ Pueden verse como **las librerías dinámicas** para el kernel (DLL).
 - ▶ El módulo puede **descargarse** cuando el dispositivo deje de usarse.

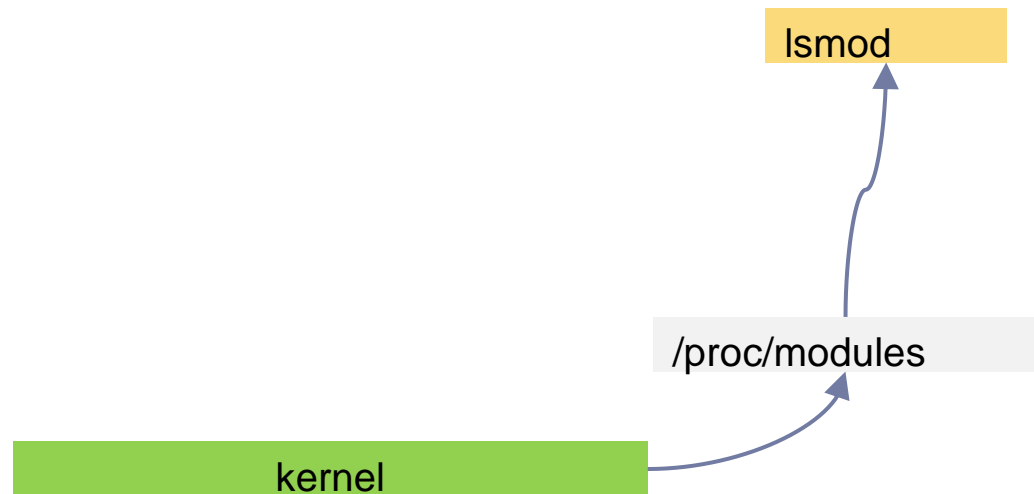
Módulos (2/2)



- ▶ La gran mayoría de **sistemas operativos modernos** tienen un kernel que permite el **uso de módulos**:
 - ▶ Linux, Solaris, BSD, Windows, etc.
- ▶ Los módulos se utilizan no solo para los drivers de los dispositivos, actualmente **también se utilizan para añadir** otro tipos de **funcionalidad**:
 - ▶ El kernel de Linux lo utiliza extensivamente para sistemas de ficheros, protocolos de red, llamadas al sistema, etc.

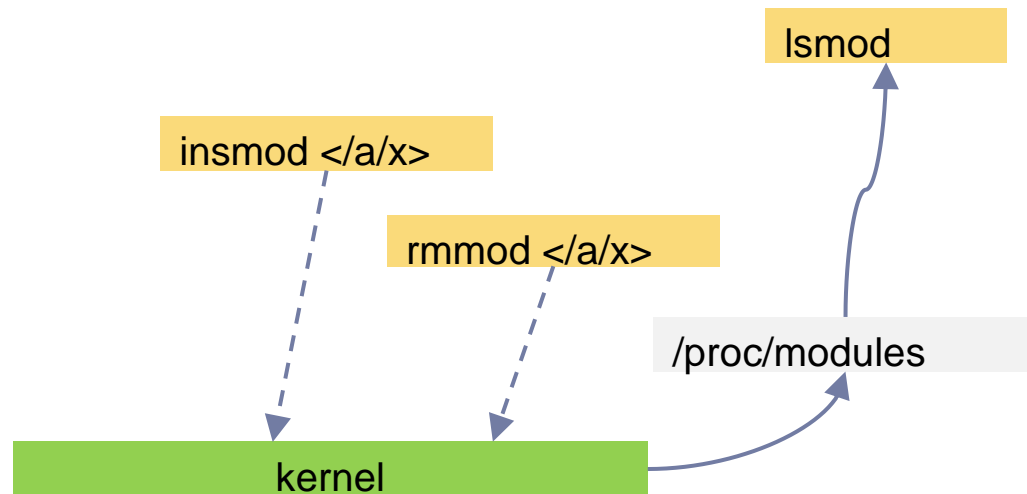
Gestión básica de módulos

Linux: listar



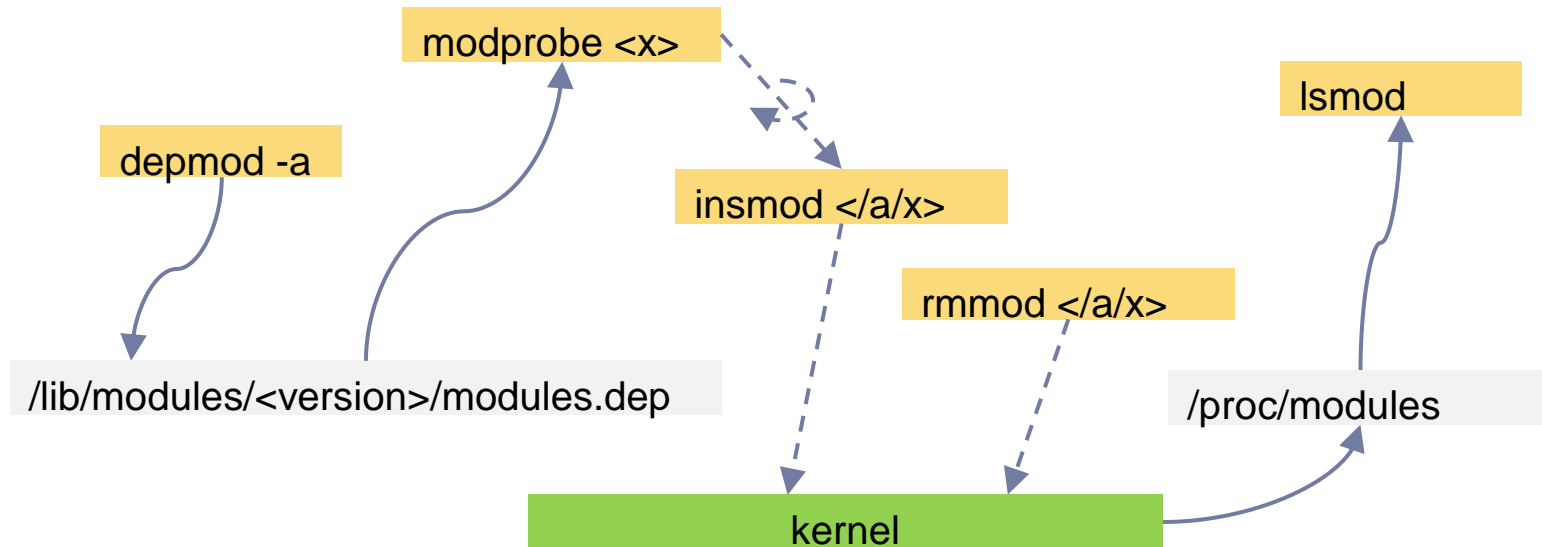
Gestión básica de módulos

Linux: añadir/quitar manualmente un módulo



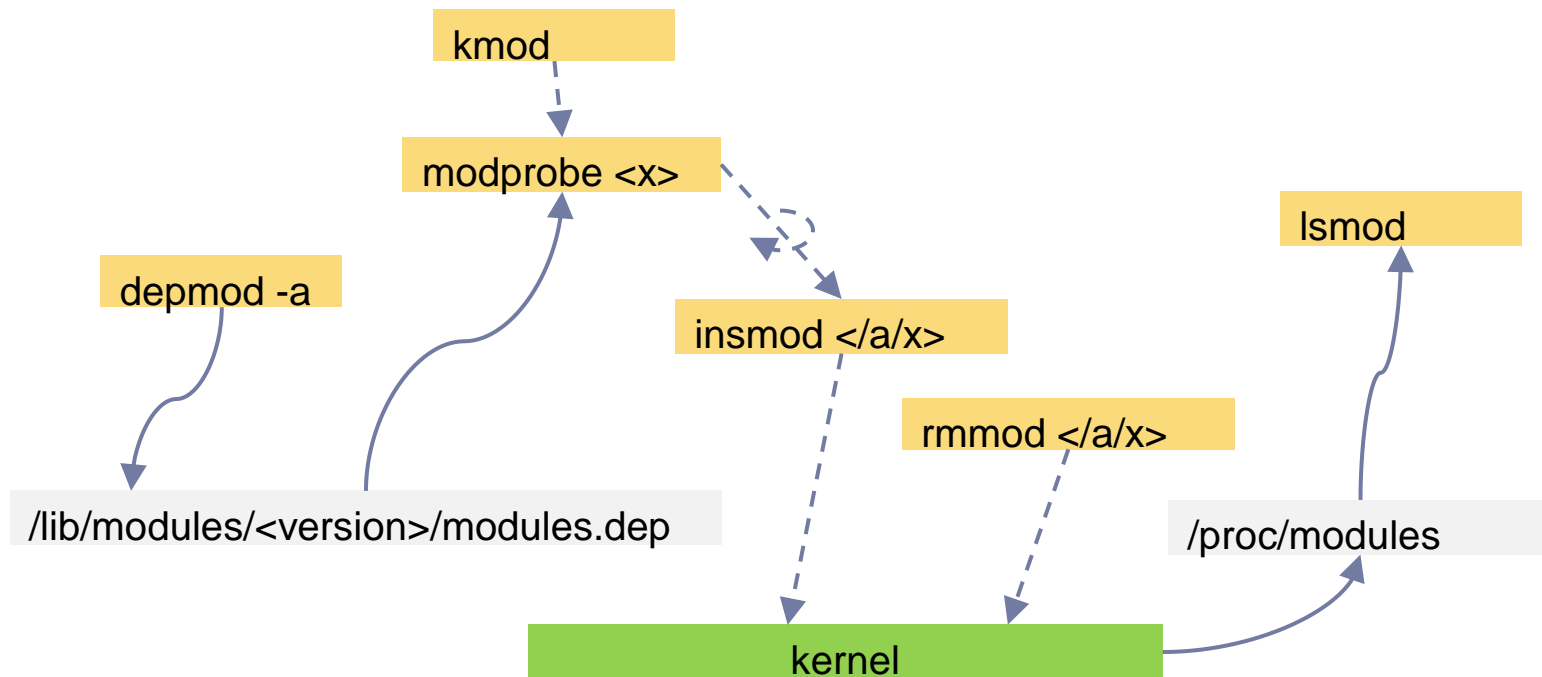
Gestión básica de módulos

Linux: añadir con dependencias



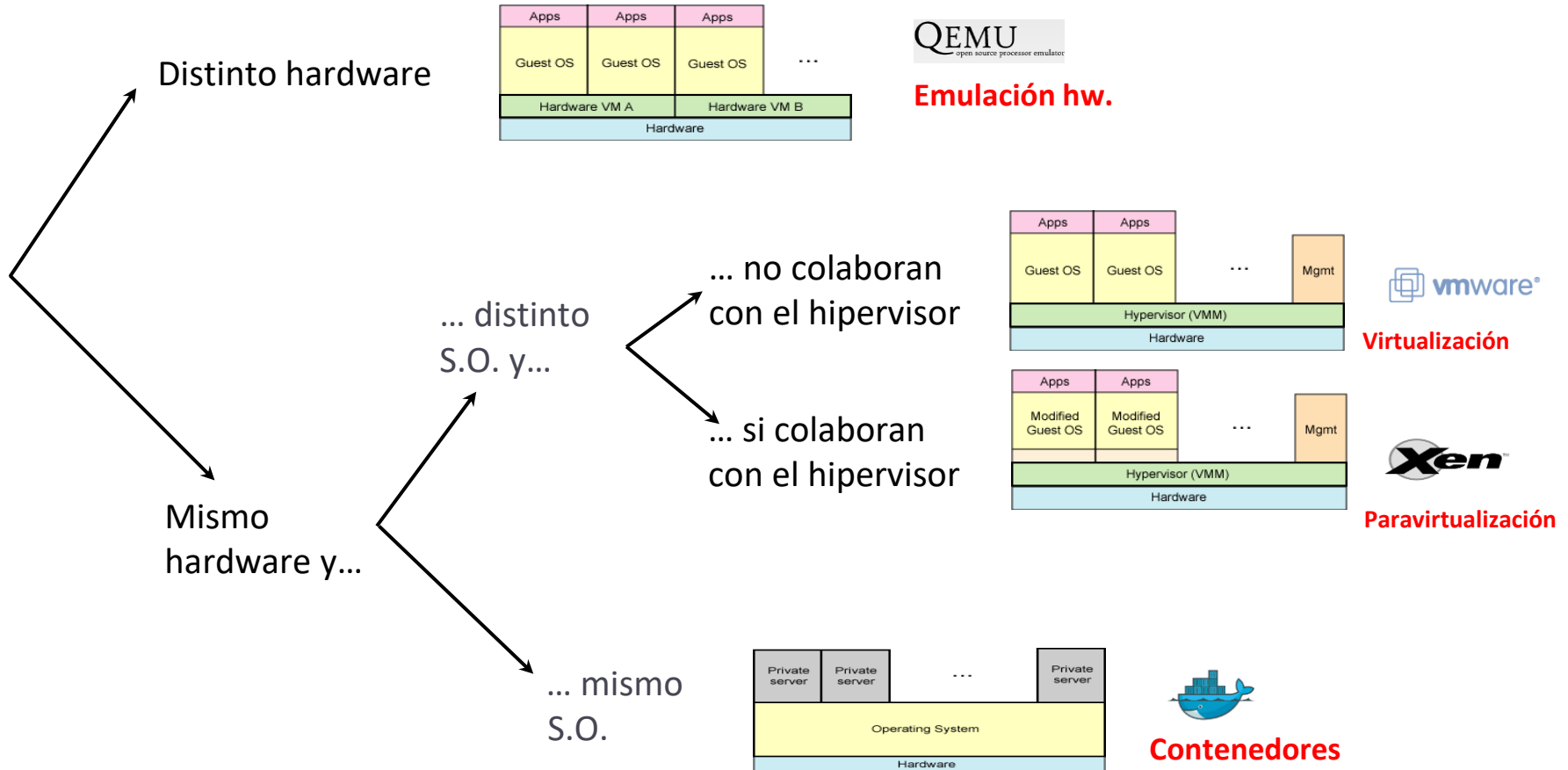
Gestión básica de módulos

Linux



Máquinas virtuales

repaso



Estructura del Sistema Operativo

- ▶ Tipos generales de estructuras de S.O.:

- ▶ Monolítico

- ▶ Estructurados

- ▶ En capas

- ▶ Microkernel

- ▶ Estructuras específicas disponibles:

- ▶ Módulos

- ▶ Máquinas virtuales

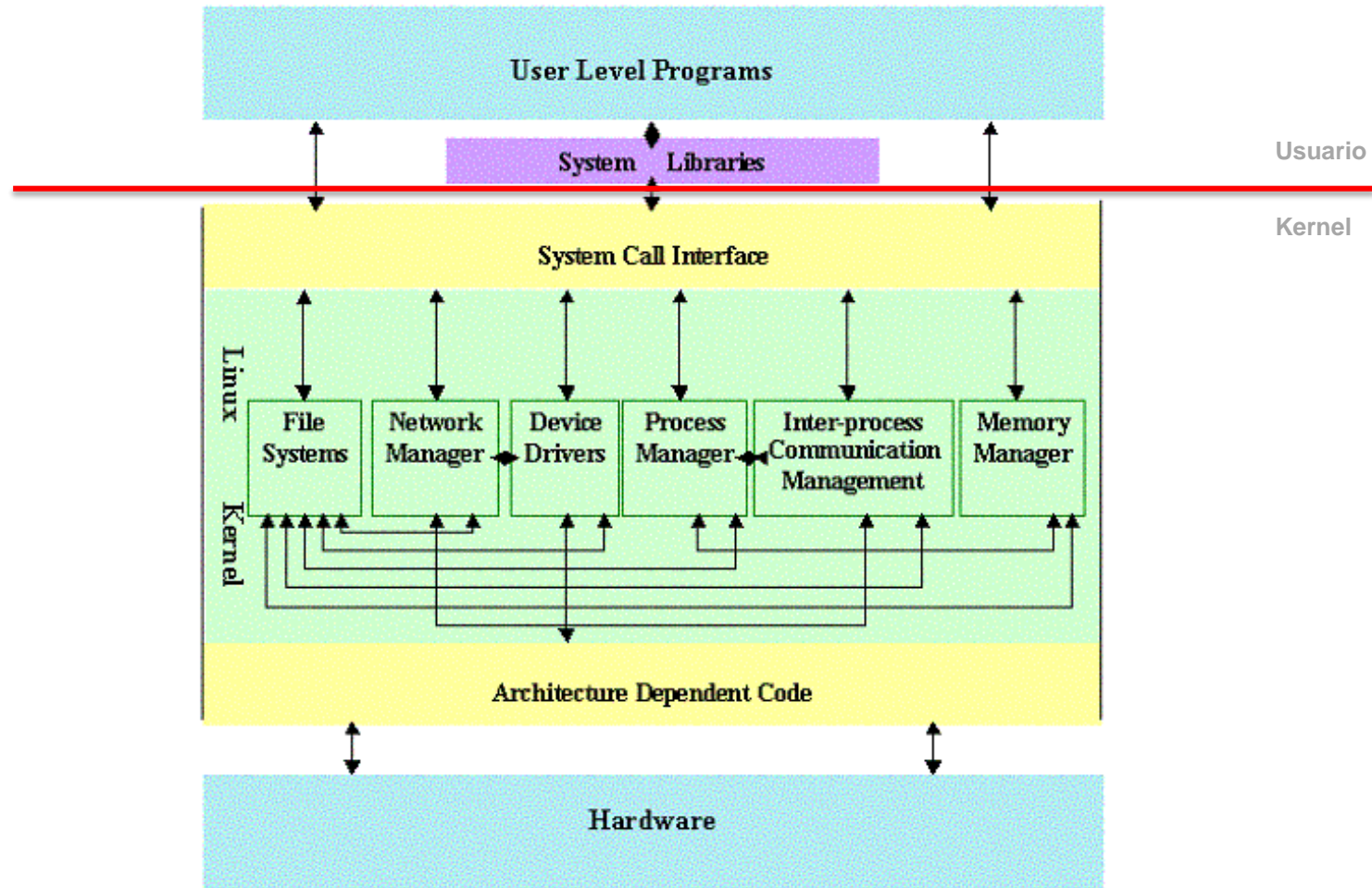
- ▶ Estructuras en algunos sistemas operativos:

- ▶ Linux

- ▶ Windows 2000

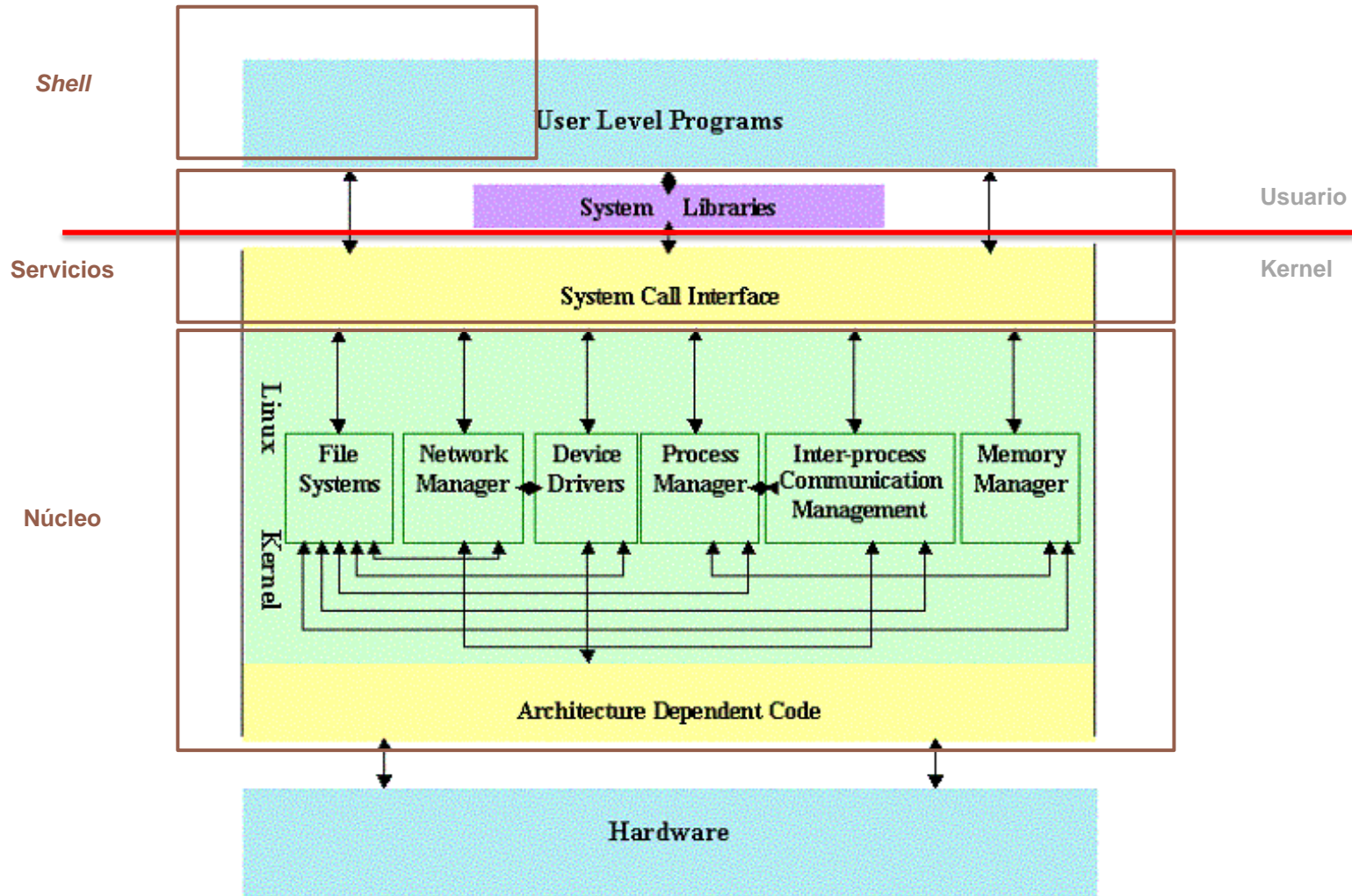
Estructura del Sistema Operativo

Linux (versión simplificada)



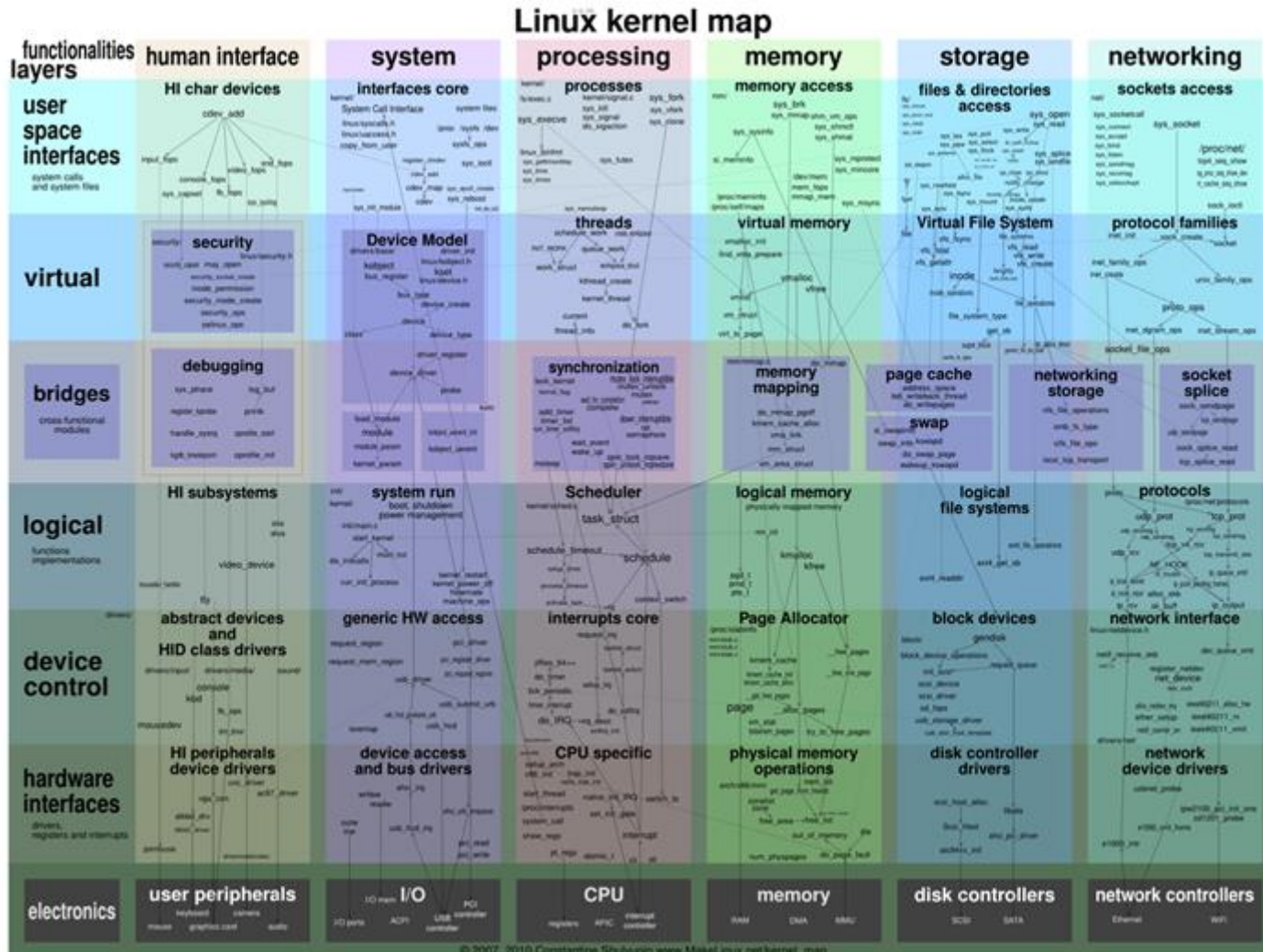
Estructura del Sistema Operativo

Linux (versión simplificada)



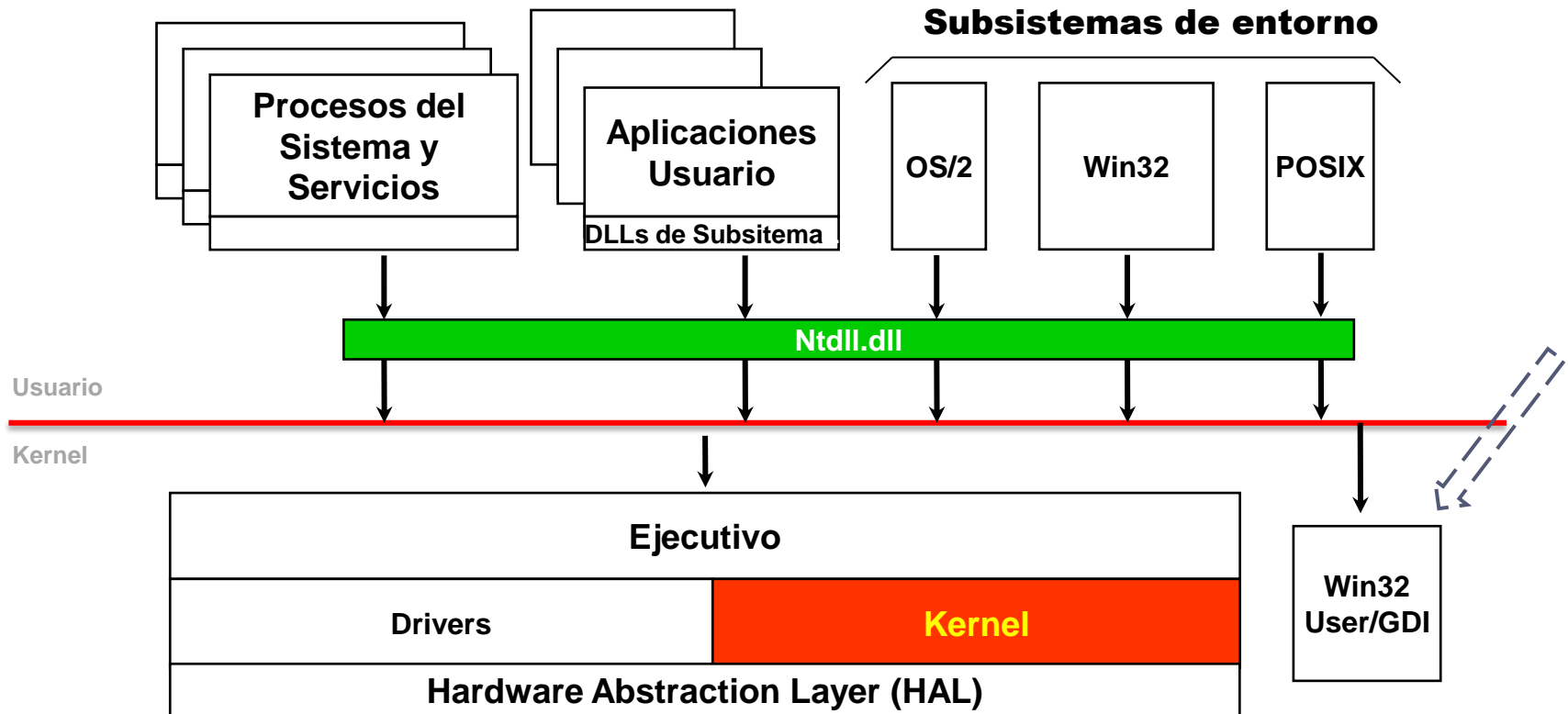
Estructura del Sistema Operativo

Linux (versión 'menos' simplificada)



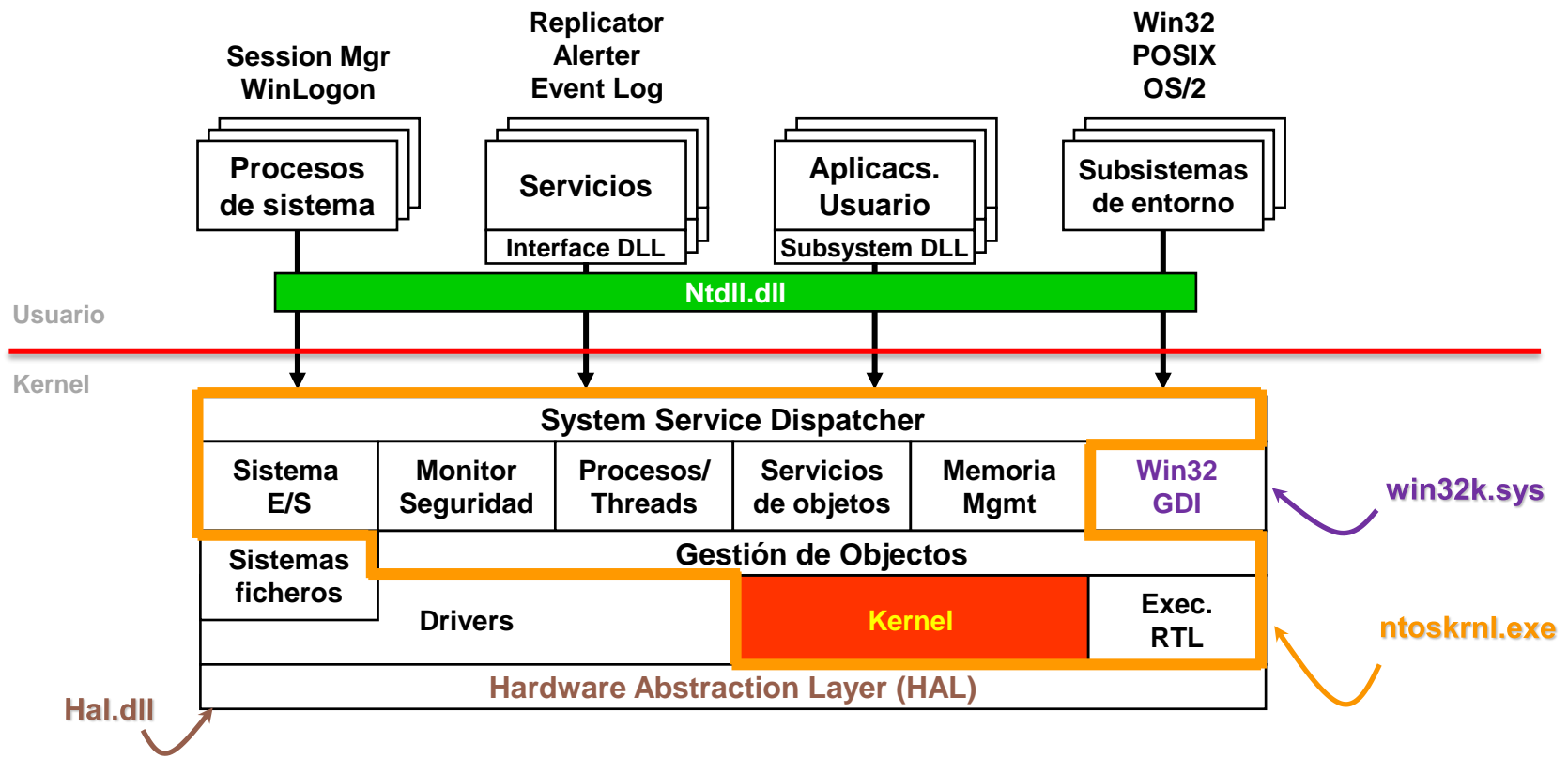
Estructura del Sistema Operativo

Windows 2000 (visión simplificada)



Estructura del Sistema Operativo

Windows 2000



Estructura del Sistema Operativo

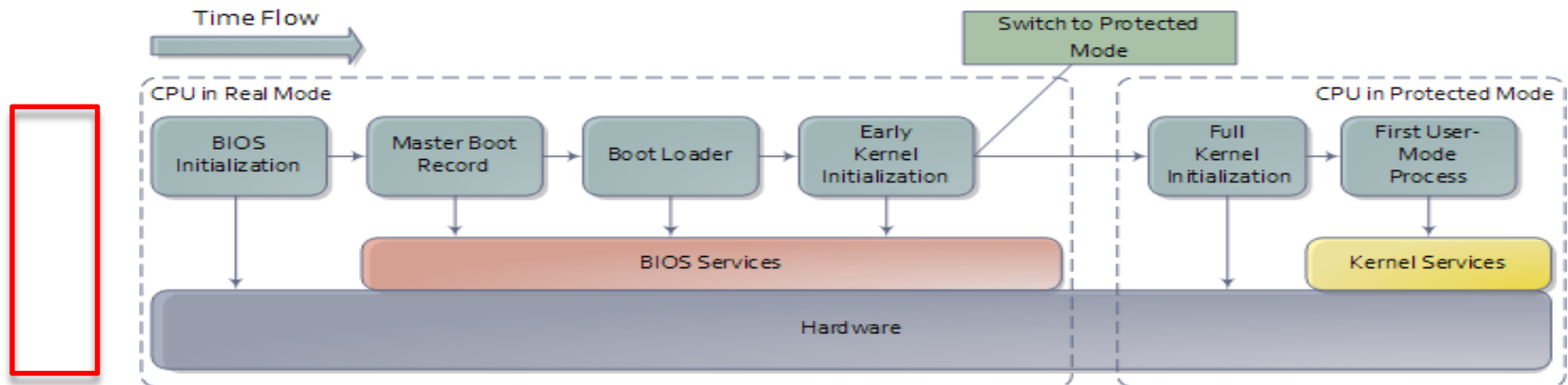
Virtualización en z/OS

- ▶ Parte del Sistema operativo OS/360 y evoluciona hasta z/OS
- ▶ Construido para los mainframe System/360 de IBM.
 - ▶ Codificado en el ensamblador del System/360
 - ▶ Principalmente orientado a proceso por lotes (batch)
 - ▶ Job Control Language (JCL)
 - ▶ Lenguaje de script para ejecutar programas por lotes (batch)
- ▶ Evolucionó progresivamente bajo los siguientes nombres:
 - ▶ MFT, MVT, SVS, MVS, MVS/XA, MVS/ESA, OS/390 y z/OS
- ▶ Operaciones típicas de mainframes:
 - ▶ Trabajos por lotes (batch)
 - ▶ Transacciones o queries on-line
 - ▶ **No soporta ningún trabajo interactivo**

Contenidos

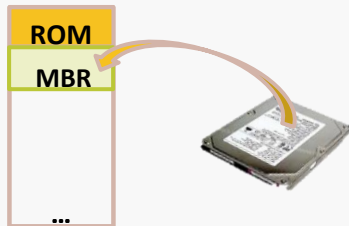
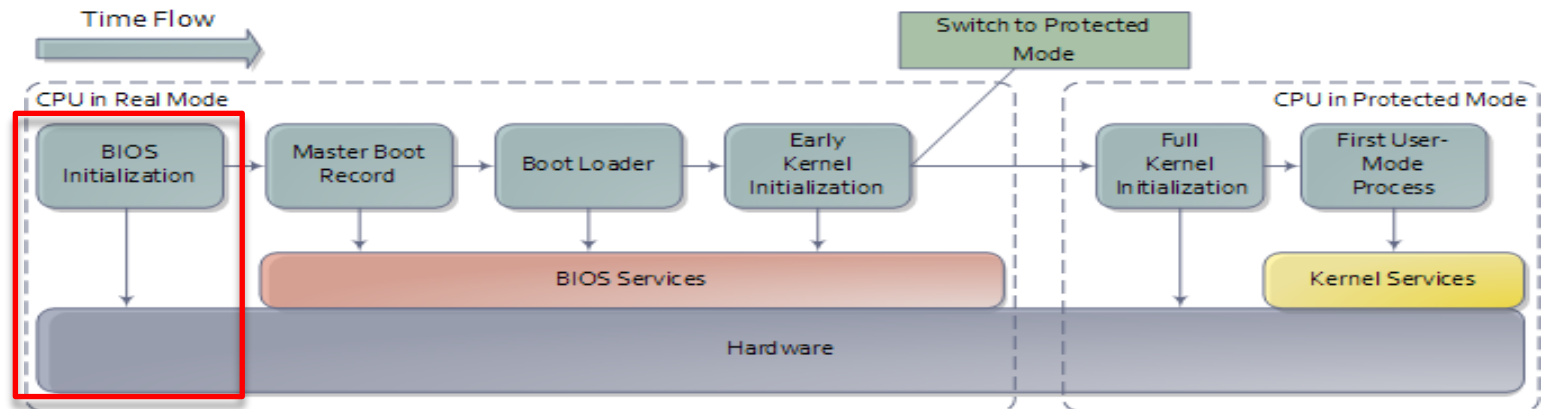
1. ¿Qué es un sistema operativo?
2. Evolución de los sistemas operativos.
3. Características.
4. Tareas de un sistema operativo.
5. Funcionamiento básico.
6. Estructura del sistema operativo.
7. **Arranque del sistema operativo.**

Proceso de arranque_{PC}



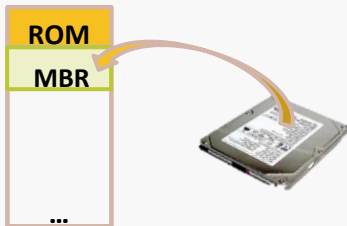
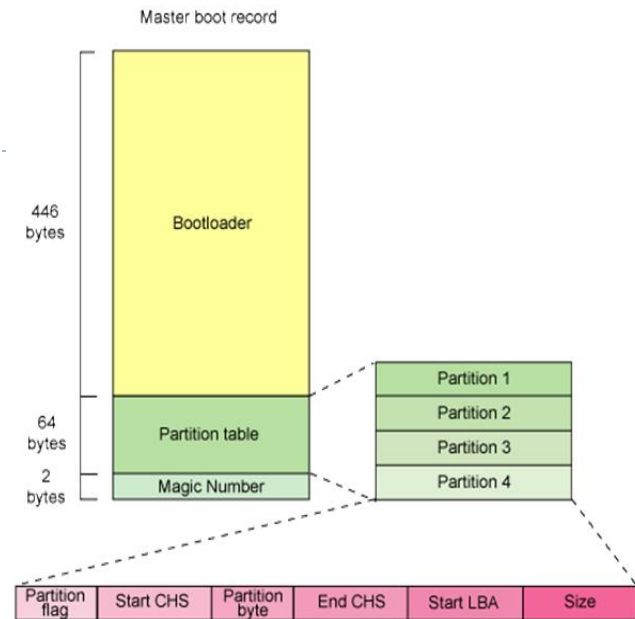
- El *Reset* carga en registros de CPU los valores iniciales
 - PC ← Dirección de arranque del cargador de la ROM (FFFF:0000)

Proceso de arranque_{PC}



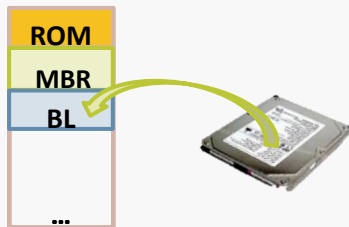
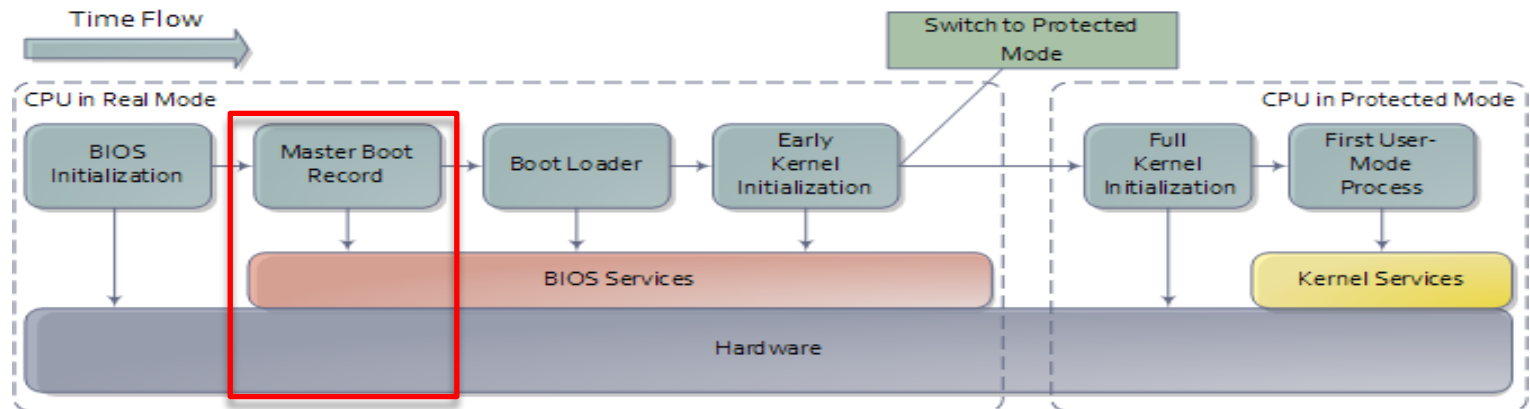
- Se ejecuta el **cargador de la ROM**
 - *Power-On Self Test (POST)*
 - Carga en memoria (0000:7C00) el *Master Boot Record*

Proceso de arranque ^{PC}



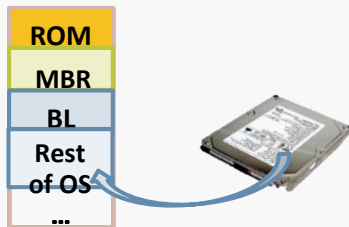
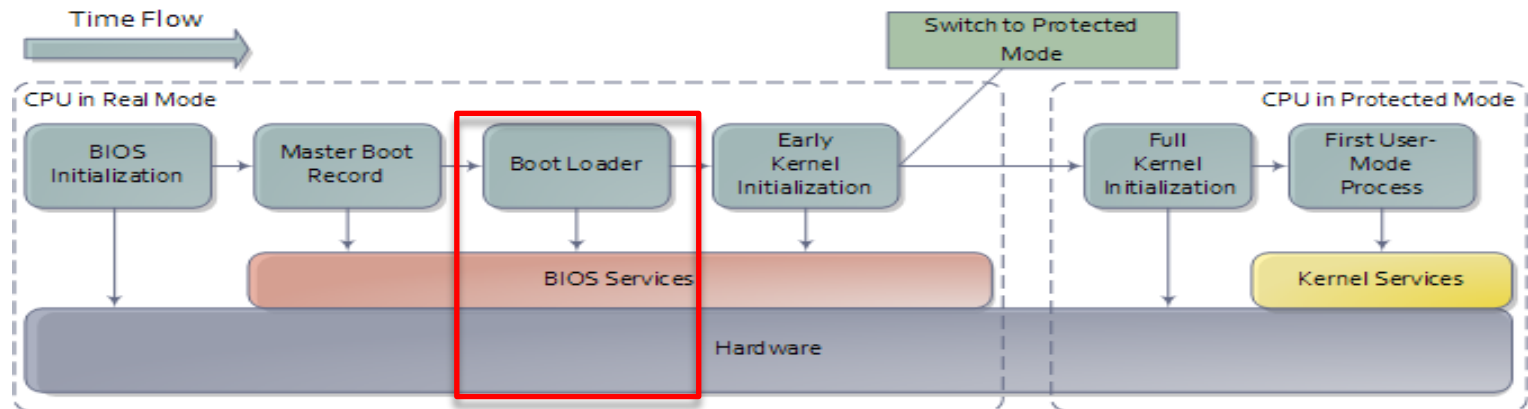
- Se ejecuta el **cargador de la ROM**
 - *Power-On Self Test* (POST)
 - Carga en memoria (0000:7C00) el **Master Boot Record**

Proceso de arranque_{PC}



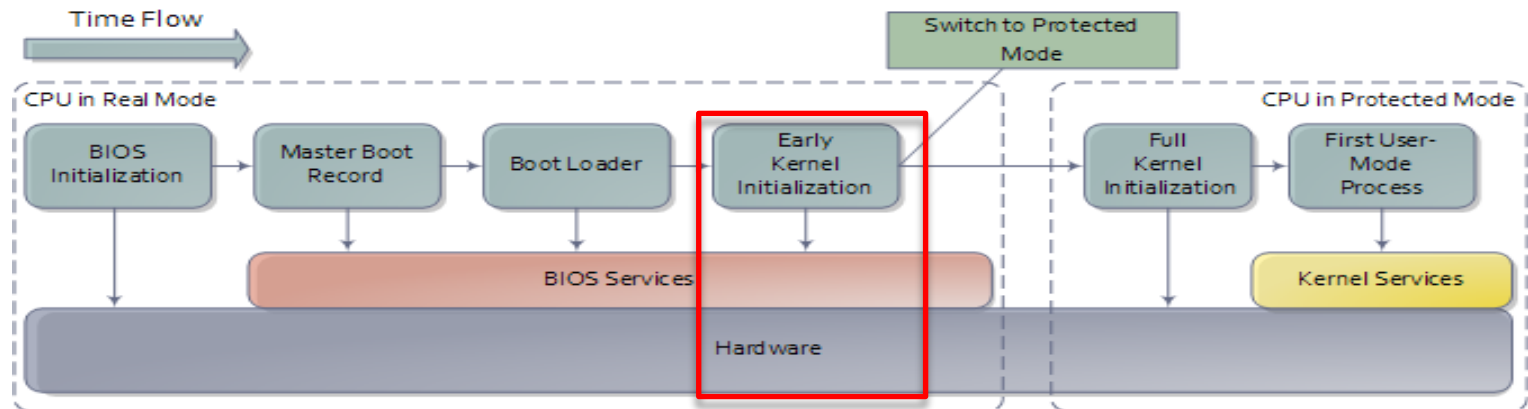
- Se ejecuta el *Master Boot Record*
 - (Es la primera parte del cargador del S.O.)
 - Busca una partición activa en la tabla de particiones
 - Carga el *Boot Record* en memoria desde esta partición

Proceso de arranque_{PC}



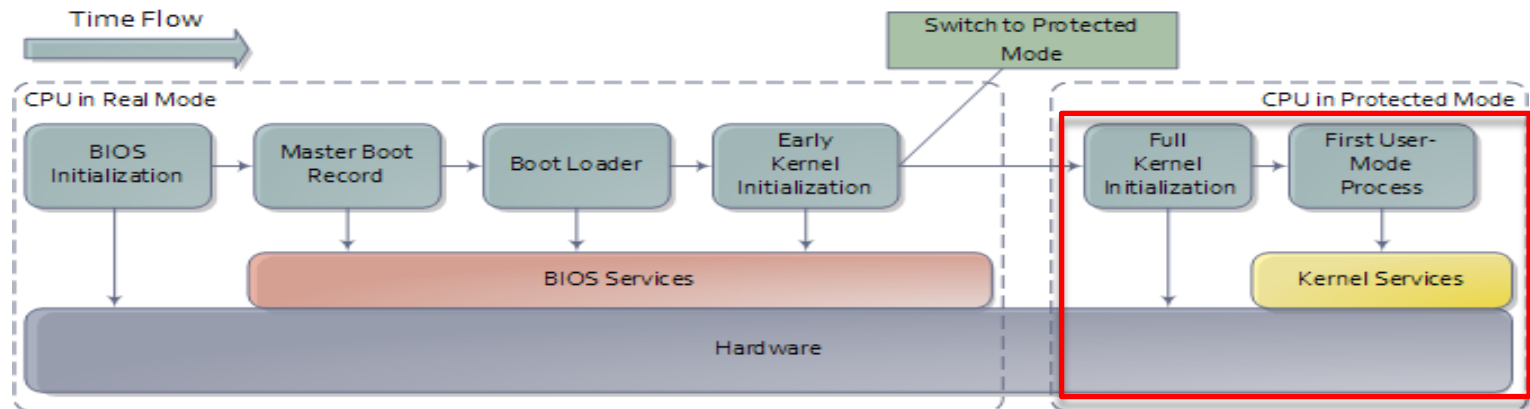
- Se ejecuta el *Boot Loader*
 - (Es la segunda parte del cargador del S.O.)
 - Podría presentar una lista de opciones de arranque...
 - El *boot loader* lleva a memoria la parte residente del sistema operativo (núcleo y módulos)

Proceso de arranque_{PC}



- Se ejecuta la **inicialización del kernel (1/2)**
 - Inicialización del hardware
 - Comprueba errores en los sistemas de ficheros
 - Establece las estructuras internas iniciales del sistema operativo
 - Pasa a modo protegido...

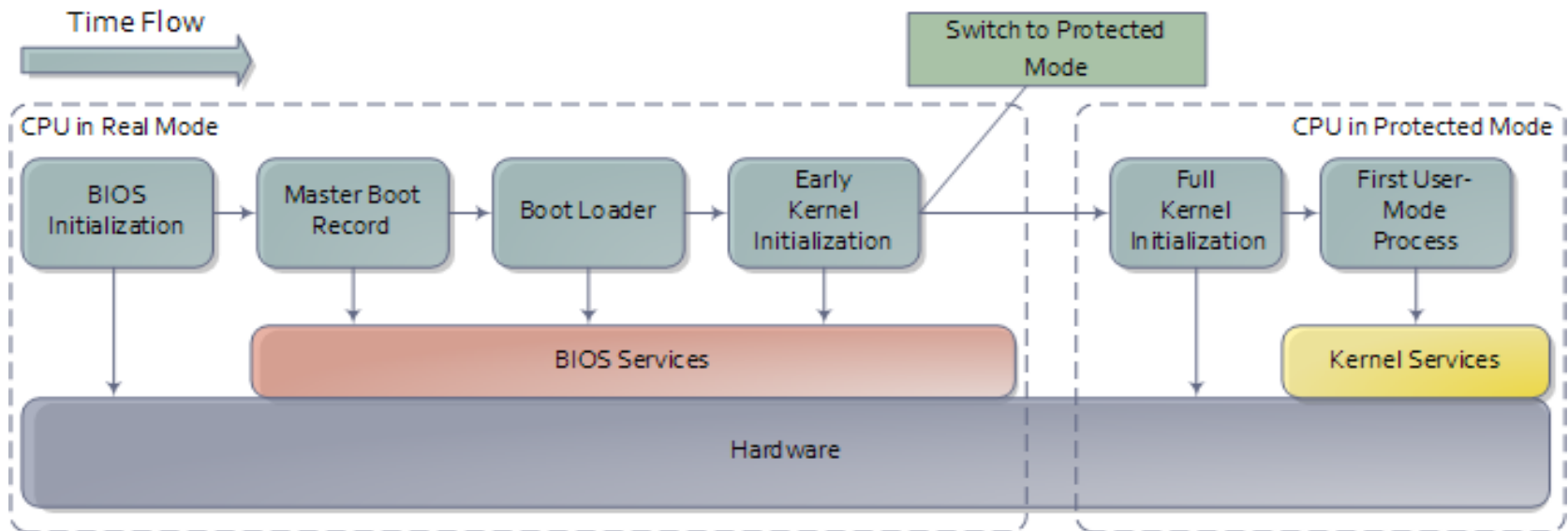
Proceso de arranque_{PC}



- Se ejecuta la **inicialización del kernel (2/2)**
 - Se establece el resto del S.O. en modo protegido
 - Se construye los procesos iniciales
 - Procesos de núcleo, servicios de sistema y terminales (*login*)

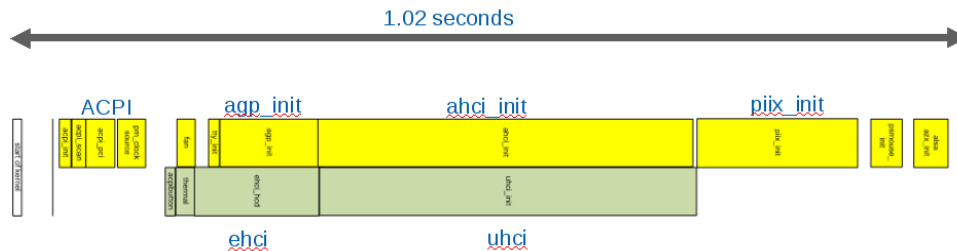
Proceso de arranque_{PC}

resumen

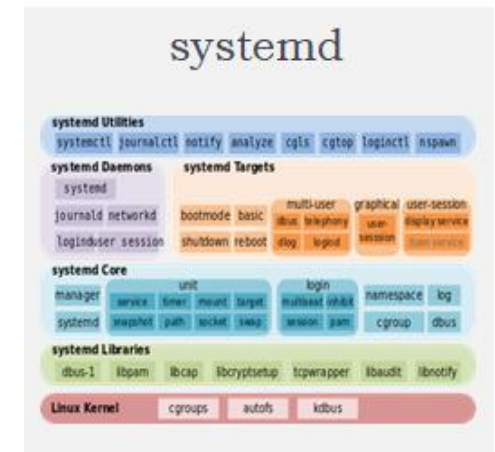
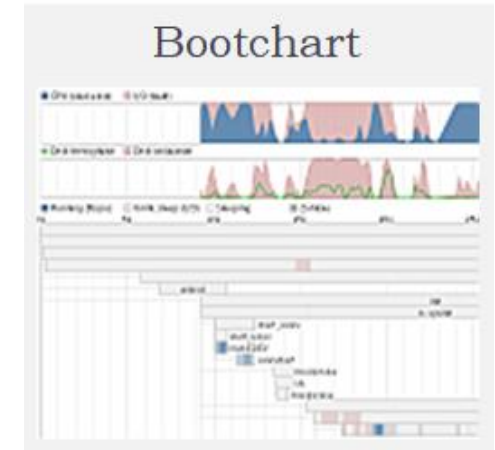
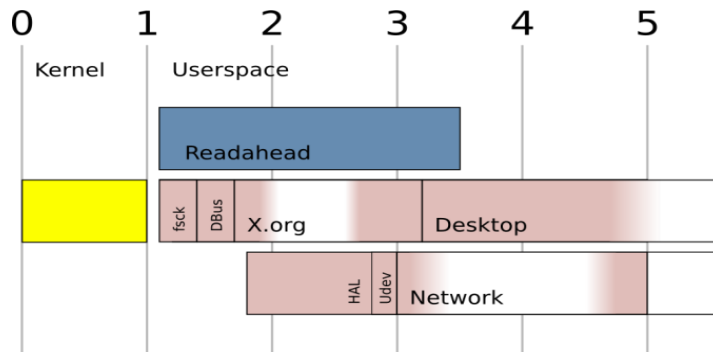


Acelerando el arranque en Linux

► Inicialización asíncrona del hardware



► Inicialización asíncrona de servicios



Parada del computador

- **Apagado en espera** (*standby*):
 - Suspensión del computador que alimenta la memoria principal pero apaga la mayoría de elementos.
- **Hibernación:**
 - Se guarda estado de la memoria principal a memoria secundaria y se apaga el computador.
- **Apagado completo:**
 - Para acelerar la ejecución el sistema operativo mantiene información en memoria no actualizada a disco.
 - Al apagar hay que volcar dicha información a disco y terminar la ejecución de todos los procesos.
 - Si no se hace volcado (apagado brusco)
 - Pérdida de información.
 - Sistema de ficheros en estado inconsistente.

Instalación del SO

- ▶ El medio de distribución contiene un sistema operativo con distintos perfiles de configuración (SO Genérico en ISO, DVD, etc.).
- ▶ Durante la instalación se genera el sistema operativo copiando los elementos necesarios basados en las características de configuración de la máquina (SO Específico).
 - ▶ Hardware existente: modelo de CPU, cantidad de memoria, dispositivos detectados, etc.
 - ▶ Opciones del sistema operativo deseadas por el usuario/a.
- ▶ Alternativas:
 - ▶ Compilación del código fuente.
 - ▶ Creación de tablas en tiempo de ejecución para generar en cada arranque del SO el perfil a usar.

Lección 1

Introducción a los sistemas operativos

Grupo ARCOS

Sistemas Operativos

Ingeniería Informática

Universidad Carlos III de Madrid

