Grupo ARCOS
Universidad Carlos III de Madrid

Lección 1
Introducción

Diseño de 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

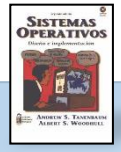
1. Carretero 2007:
   1. Cap. 2

La sugerencia del

Carretero 2007:

Sec. 2.2.- Arranque el Computador

Recomendada



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

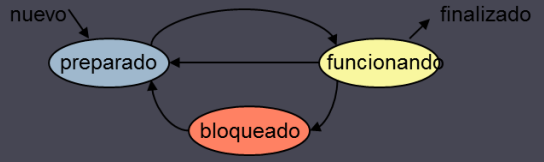
A recordar…

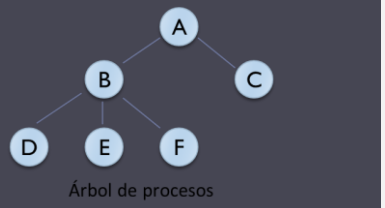
1. Estudiar la teoría asociada.
   - Estudiar el material asociado a la bibliografía: las transparencias solo no son suficiente.
2. Repasar lo visto en clase.
   - Realizar el cuaderno de prácticas progresivamente.
3. Ejercitar las competencias.
   - Realizar las prácticas progresivamente.
   - Realizar todos los ejercicios posibles.

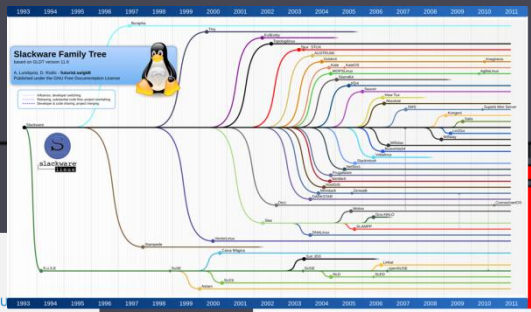
Contenidos

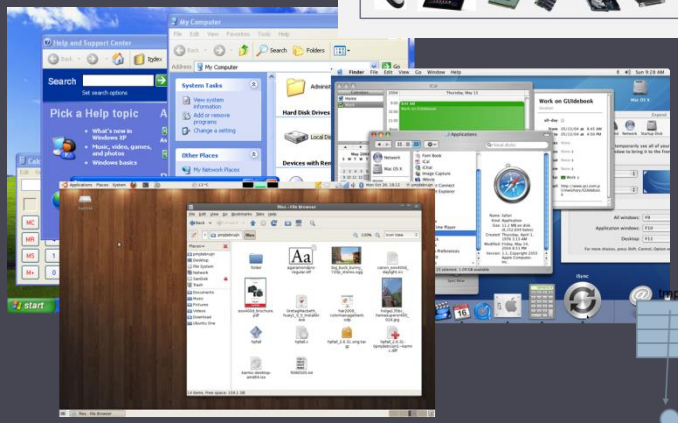
1. Qué es un sistema operativo.
   1. Definiciones
   2. Principales funciones
   3. Características
2. Cómo es por dentro.
   1. Principales objetivos
   2. Estructura
   3. Ejecución asíncrona
3. Cómo es por fuera.
   1. Núcleo
   2. Módulos

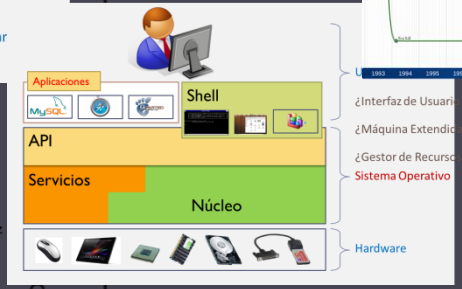
¿Qué es un Sistema Operativo?

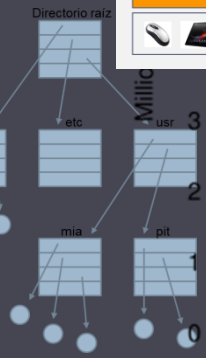












Contenidos

1. Definición
2. Principales Funciones.
3. Características.

¿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?

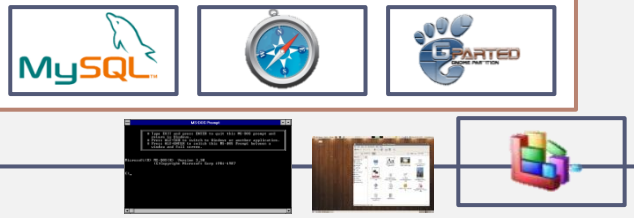
- Sistema operativo: software destinado a permitir la comunicación del usuario con un ordenador y gestionar sus recursos de manera cómoda y eficiente.
  - Complejidad para encubrir el hardware subyacente (en el S.O.)
  - Programas del sistema para la gestión de los recursos.

Contenidos

1. Definición.
2. Principales Funciones.
3. Características.

¿Qué tiene que hacer un S.O.?

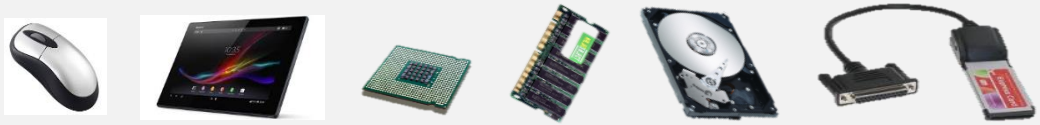
Aplicaciones



Usuario(s)

- Proporcionar Interfaz al Usuario (s)
- Máquina Extendida
  - Servicios, Interfaz del Programador, etc.
- Gestionar los recursos del equipo
  - CPU, Memoria, etc.

Sistema Operativo

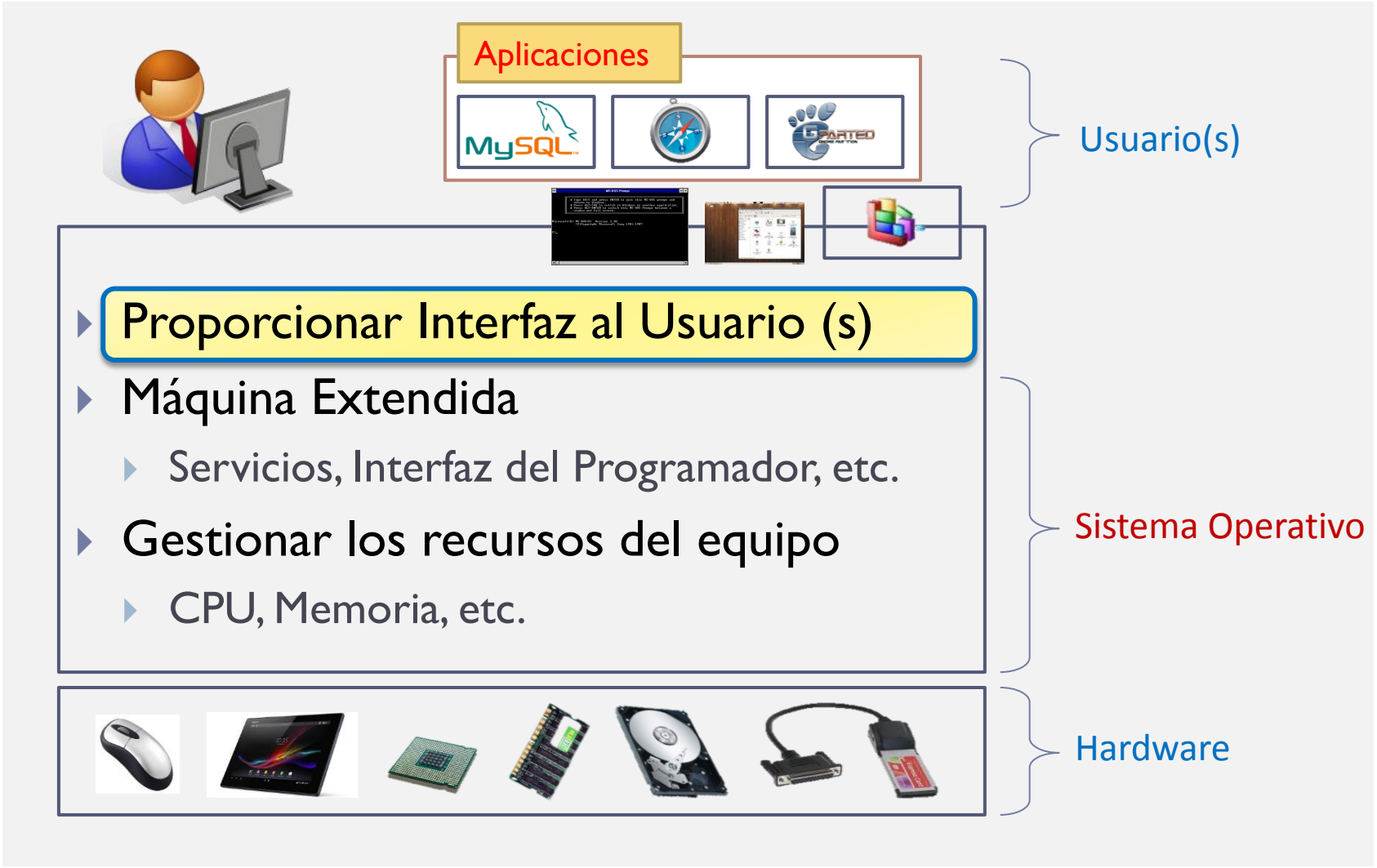


Hardware

Funciones del S.O.

- Interfaz de Usuario:
  - Ejecuta Órdenes de Usuario
  - Proporciona los resultados de esas órdenes
- Gestor de Recursos:
  - Asignación de Recursos
  - Contabilidad de los Recursos
  - Administración de los Recursos
  - Protección y ¿Seguridad?
- Máquina Extendida
  - Ejecuta, Administra y Gestiona Servicios para los Programas
  - Proporciona nuevos Recursos: canales de E/S, archivos, gestión de memoria y de errores, etc.
  - Proporciona Funcionalidades de Alto Nivel para todos los recursos

¿Qué tiene que hacer un S.O.?



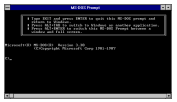
Interfaz de Usuario: Función

- El Interfaz de Usuario tiene dos funciones:
  - Ejecución de Órdenes del Usuario.
  - Proporcionar los Resultados de esas Órdenes.

Interfaz de Usuario: Tipos

Interfaz de Usuario:

- Suele utilizar una interfaz gráfica (Graphics User Interface, GUI)



Interfaz de Usuario avanzado:

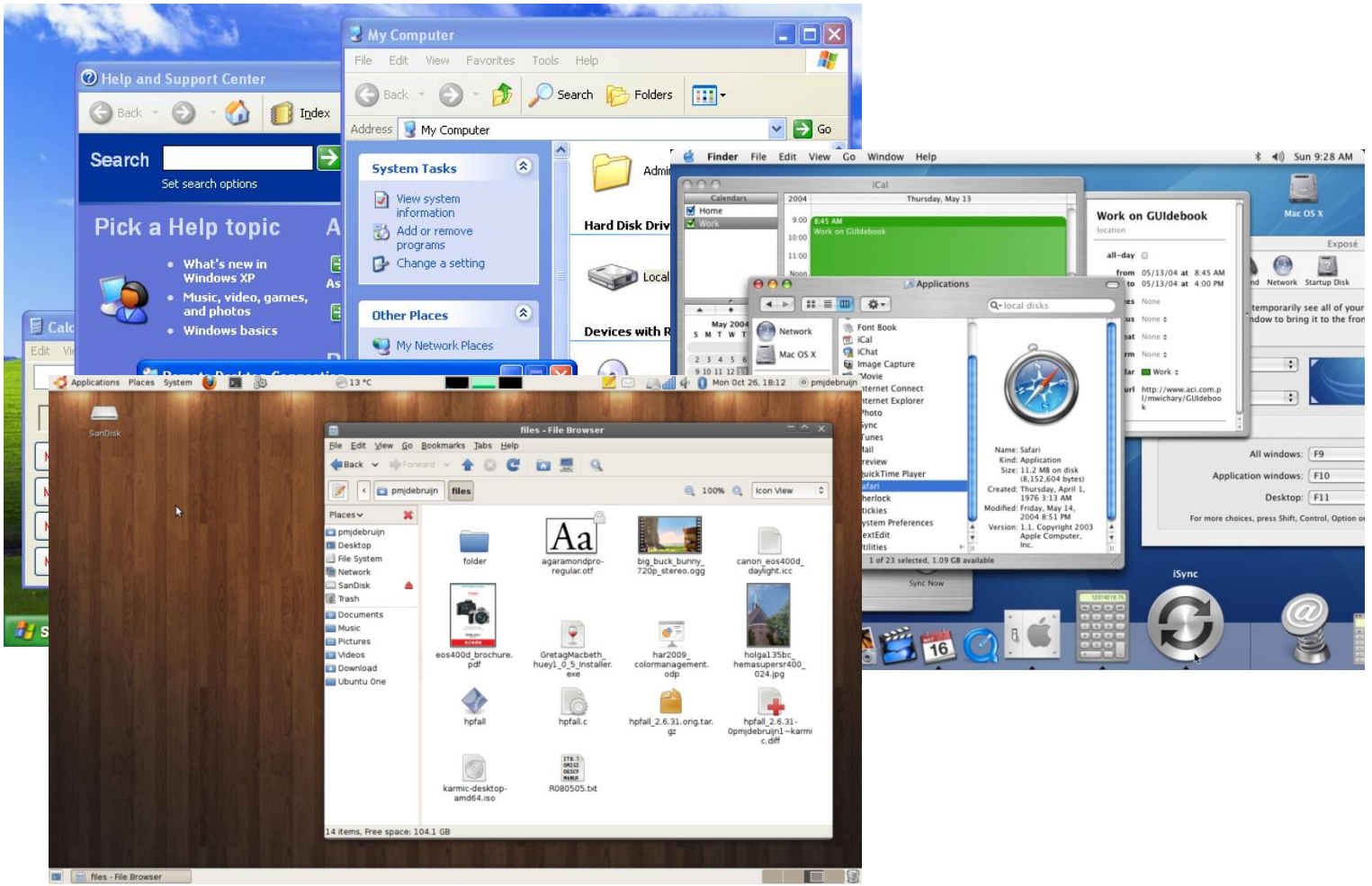
- Se utiliza una Línea de Mandatos o Línea de Comandos (Command Line Interpreter, CLI)

Interfaz del Desarrollador:

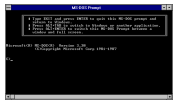
- Utilizado por Diseñadores y Programadores
- Utiliza funciones de Librería del Sistema

```
ret = close (filedesc);
```

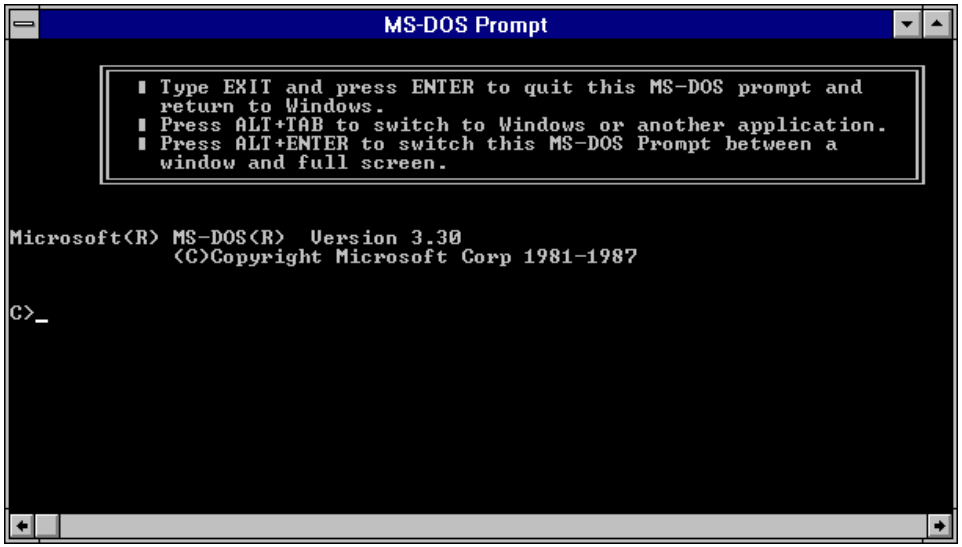
Interfaces gráficas (GUI)

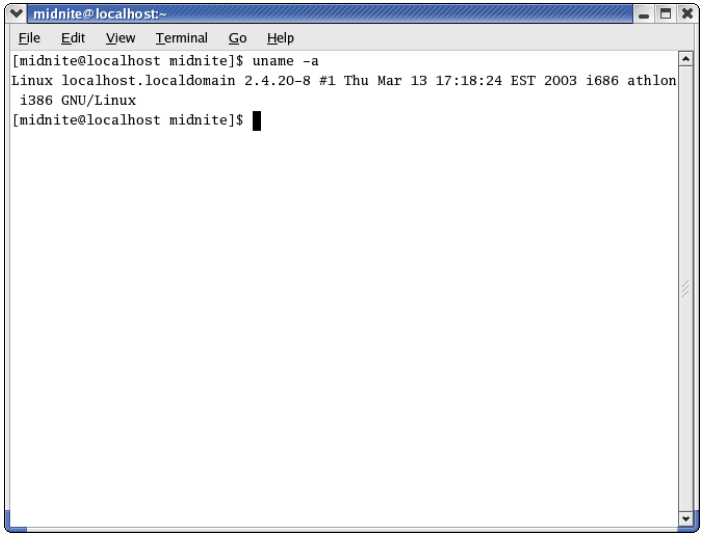


http://www.guidebookgallery.org/screenshots/commandprompt
http://www.guidebookgallery.org/screenshots/full



Intérprete de mandatos (CLI)

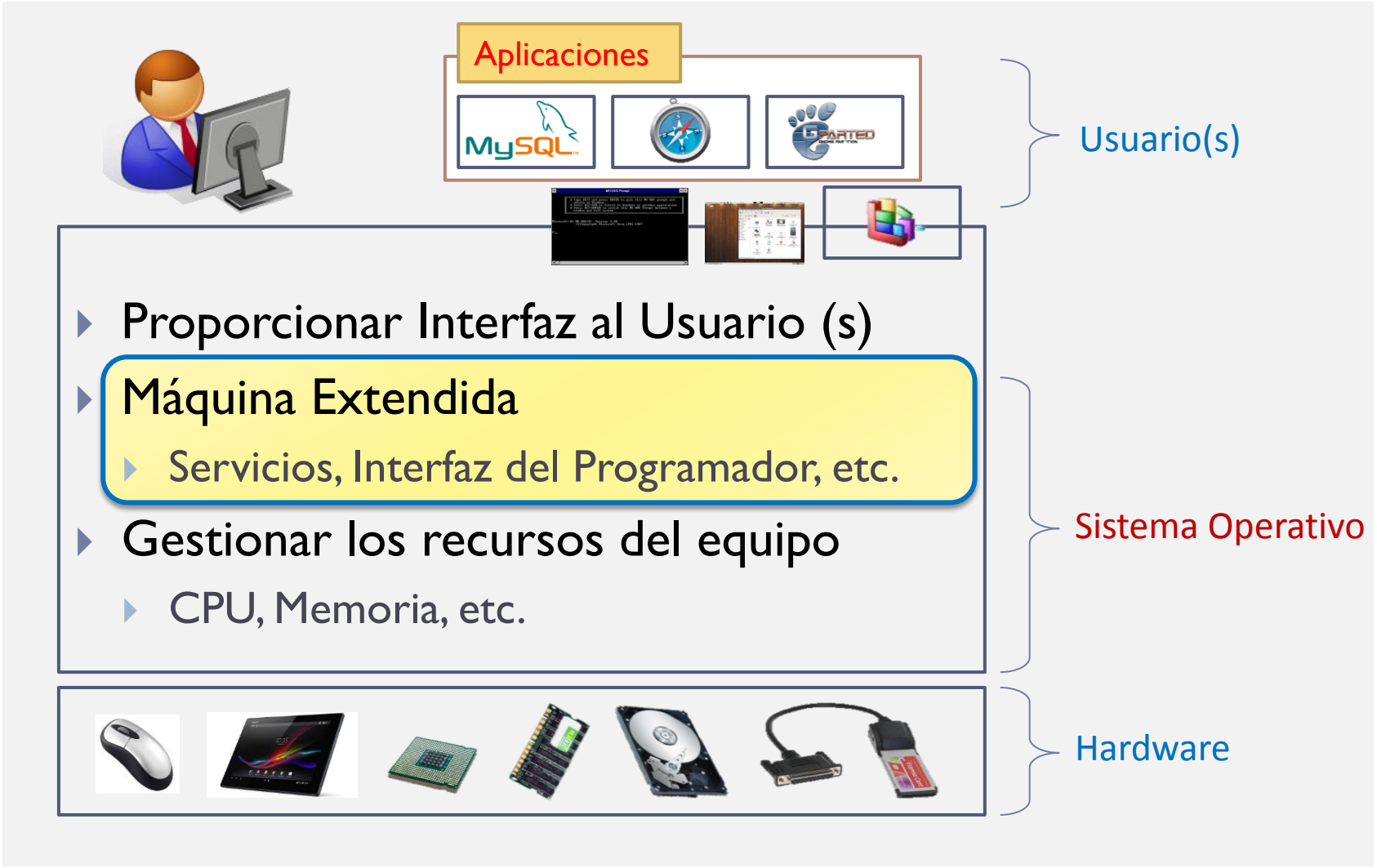




http://www.guidebookgallery.org/screenshots/commandprompt
http://www.guidebookgallery.org/screenshots/full

http://www.guidebookgallery.org/screenshots/commandprompt

¿Qué tiene que hacer un S.O.?



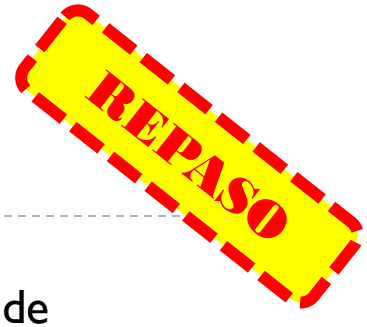
Máquina Extendida

- La Máquina Extendida se encarga de:
  - La ejecución de los Procesos.
  - La ejecución de las Órdenes de E/S (a Alto Nivel).
  - La realización de las Operaciones sobre Archivos.
  - La detección y tratamiento de Errores.

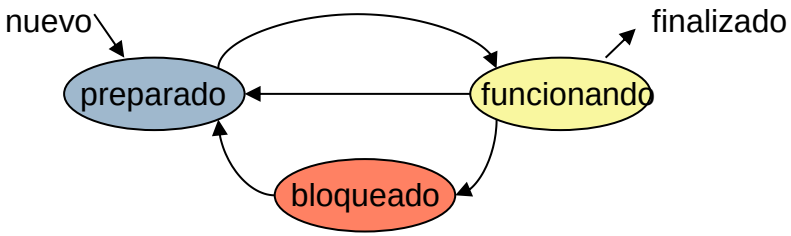
- En resumen, abarca la Ejecución, Administración y Gestión de los Servicios que proporciona el S. O. a todos los Procesos.

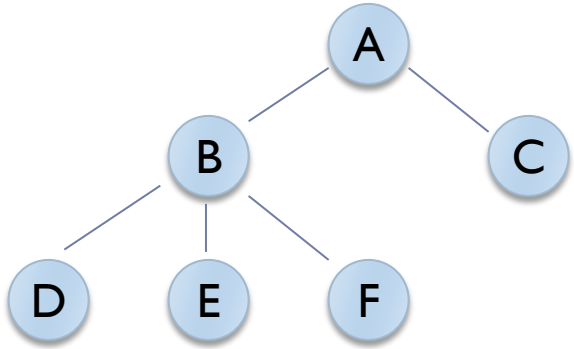
Abstracciones fundamentales

Procesos



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





Árbol de procesos

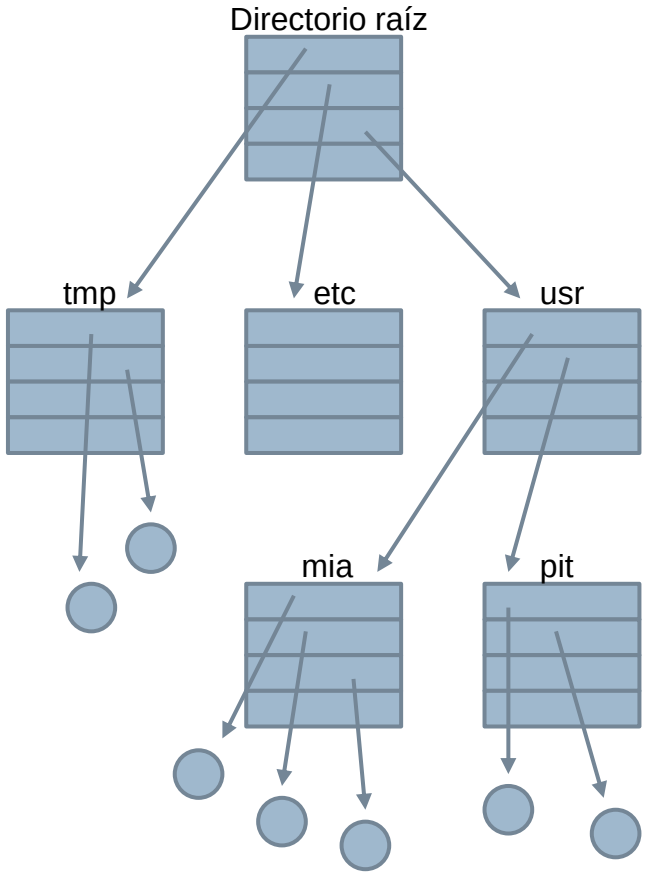
- Un Programa es algo estático
- Un Proceso es algo dinámico
- Un Proceso = Un Programa + Datos de Entrada + Información de Estado

https://www.microsoft.com/resources/sharedsource/windowsacademic/curriculumresourcekit.mspx

Abstracciones fundamentales
Archivos

REPASO

- 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

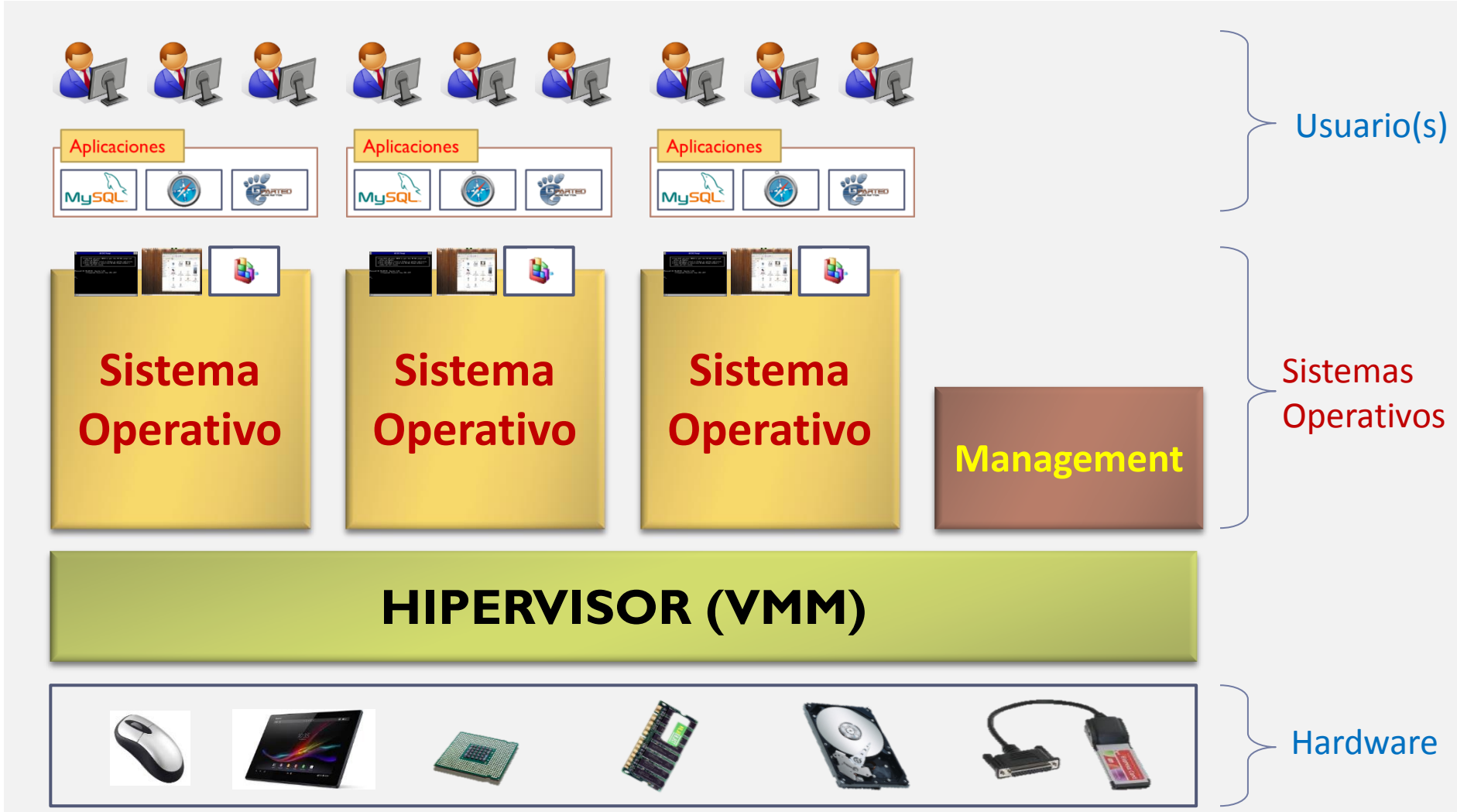


https://www.microsoft.com/resources/sharedsource/windowsacademic/curriculumresourcekit.mspx

Máquina Virtual

- El S.O. virtualiza ciertos elementos del hardware:
  - ¿Por qué no virtualizarlo todo?
- IBM ha usado esta idea en sus mainframes desde principio de la década de los 70.
- Un Hipervisor (VMM) 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»

Máquina Virtual



Máquina Extendida vs Máquina Virtual

Máquina Extendida vs Máquina Virtual

Máquina Virtual

Distinto HW

- Emulación del HW

QEMU open source processor emulator

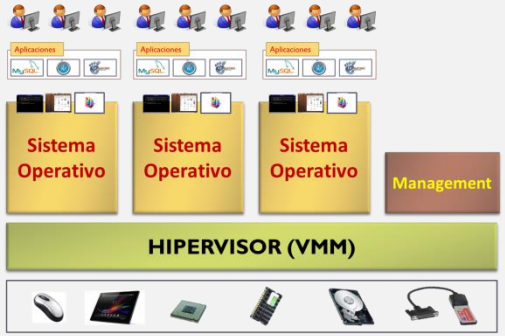
Mismo HW

Distinto S.O.

No Colabora con el Hipervisor

- Virtualización Completa

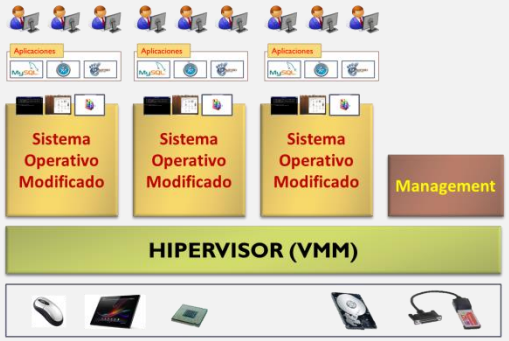
vmware



Colabora con el Hipervisor

- Para-Virtualización

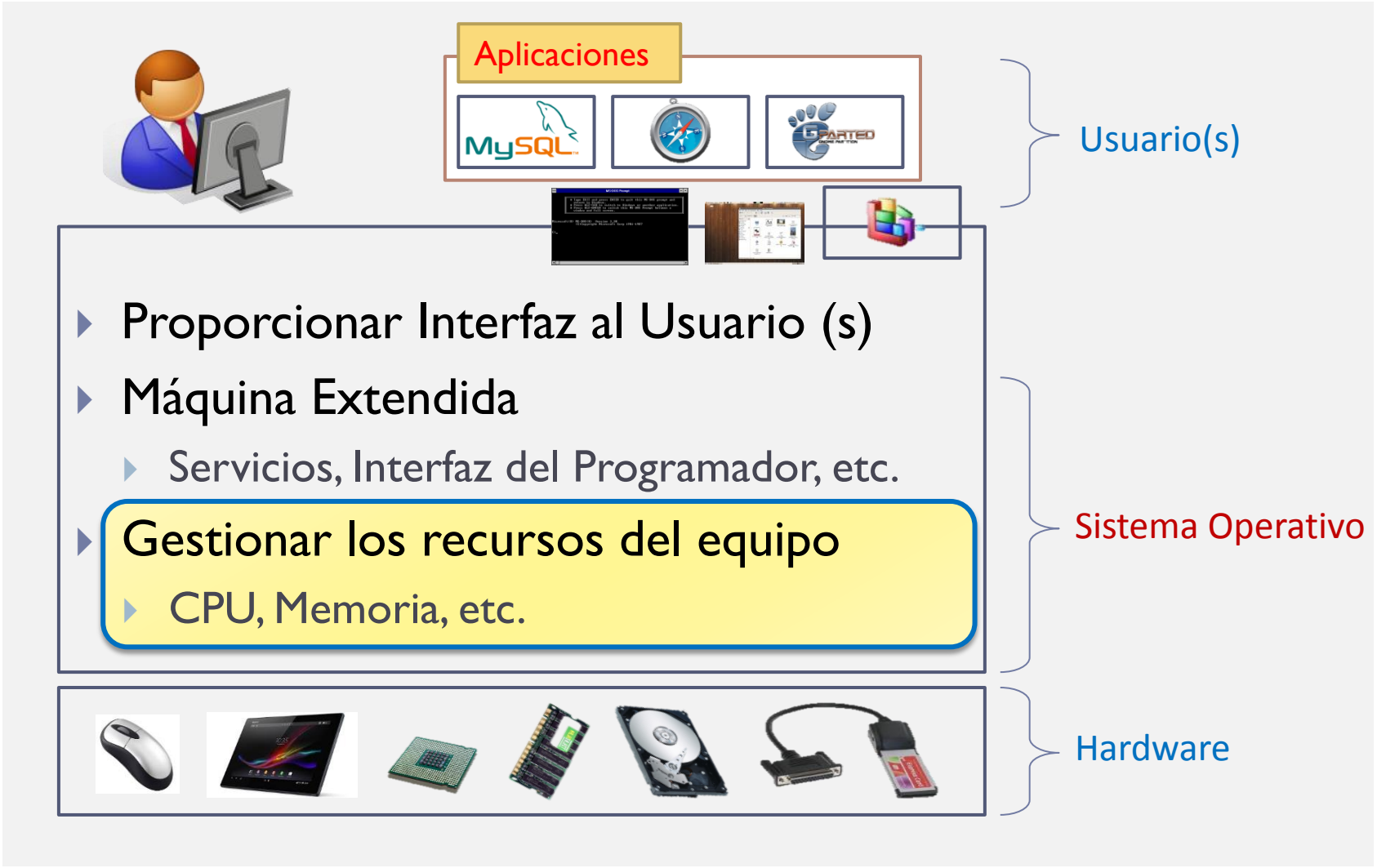
Xen



Mismo S.O.

- Contenedores

¿Qué tiene que hacer un S.O.?



Gestión de Recursos: ¿Qué gestiona?

- Gestión de Procesamiento
  - 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

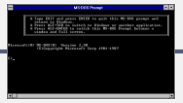
Gestión de Recursos: ¿Qué es gestionar?

- Asignación de Recursos
- Contabilidad de los recursos
  - Estado, Asignación, etc.
- Protección y ¿¿Seguridad??
- Administración de Recursos
  - Determinación de la Asignación
  - Expropiación / Recuperación de Recursos
  - Aplicación de las políticas a los mecanismos internos del S.O.

Resumen

Aplicaciones



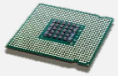


Usuario(s)

- Proporcionar Interfaz al Usuario (s)
- Máquina Extendida
  - Servicios, Interfaz del Programador, etc.
- Gestionar los recursos del equipo
  - CPU, Memoria, etc.

Sistema Operativo






Hardware

Contenidos

1. Definición.
2. Principales Funciones.
3. Características.

Las principales características de un S.O.

1. Versátil
    1. Portabilidad
2. Adaptativo
3. Multidisciplinar
4. Complejo
5. Delicado

Versátil

Supercomputador

Unix, Linux, …

Mainframe

OS/360, z/OS, …

Miniordenadores y PC

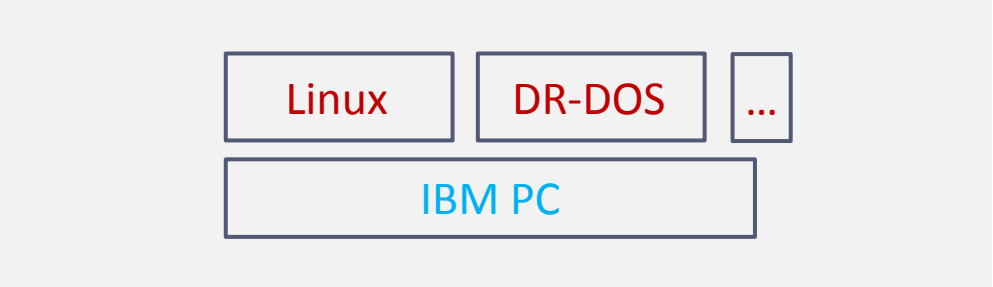
Unix, MacOs, Windows, …

Empotrados

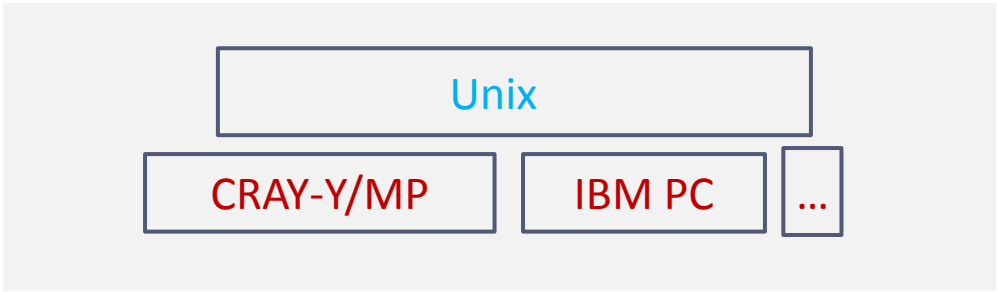
VxWorks, QNX, LynxOS,
Android, iOS,
Windows Embedded, …

1) Versátil

- Mismo equipo, diferentes SSOO: IBM PC



- Mismo SO, diferentes equipos: Unix

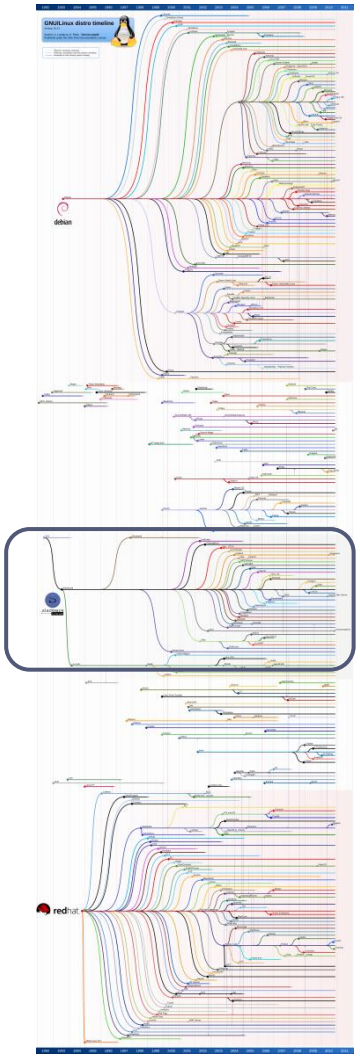


Portabilidad

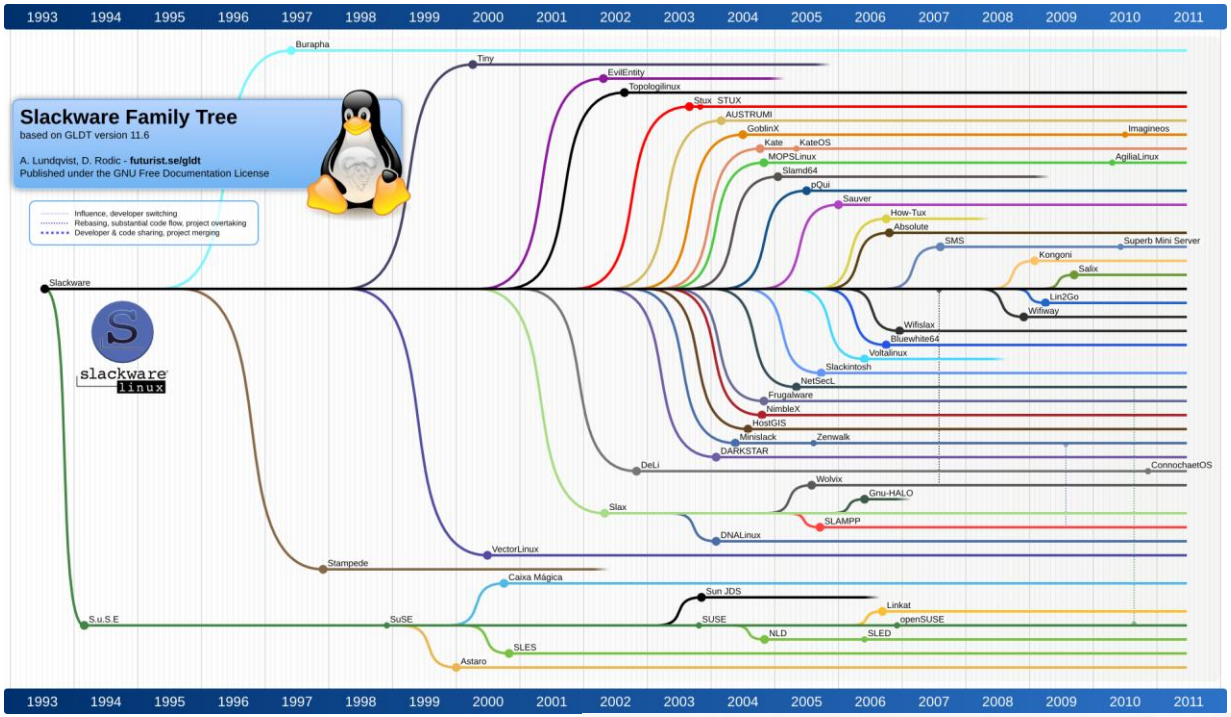
2) Continuos cambios para adaptarse

- Está sometido a continuos cambios para adaptarse a:
  - Las nuevas demandas de los usuarios:
    - Reconocimiento de voz, entrada multitáctil, etc.
  - La evolución o nuevo tipo de hardware:
    - Controladores para todo tipo de nuevos dispositivos
    - Sistemas multicore, virtualización, etc.
  - La integración de 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



- Slackware es la más antigua de todas y sigue en evolución:



- Otras distribuciones:

http://futurist.se/gldt/wp-content/uploads/gldt1012.png

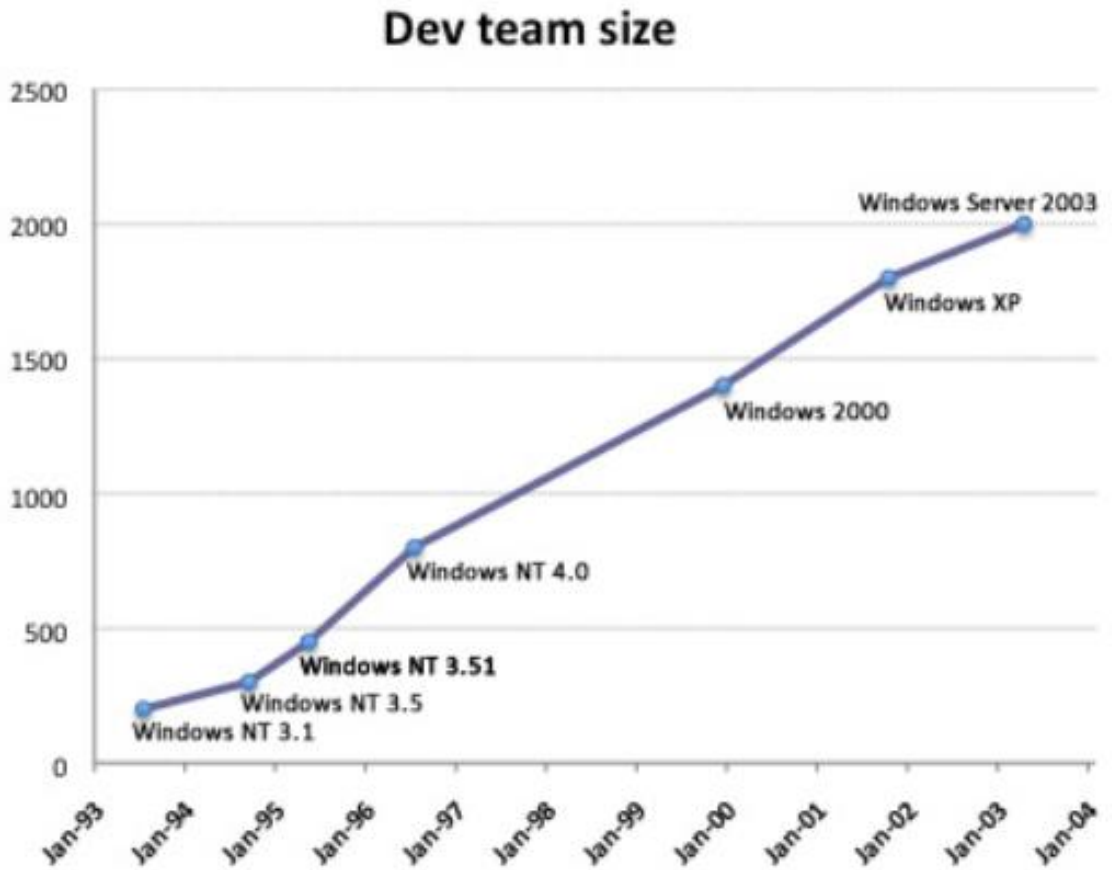
3) Multidisciplinar

- Software multidisciplinar que integra trabajos de diferentes áreas:
  - Interfaces de usuario, software de sistema, inteligencia artificial, seguridad, Ingeniería Software, etc.

4) Software Complejo

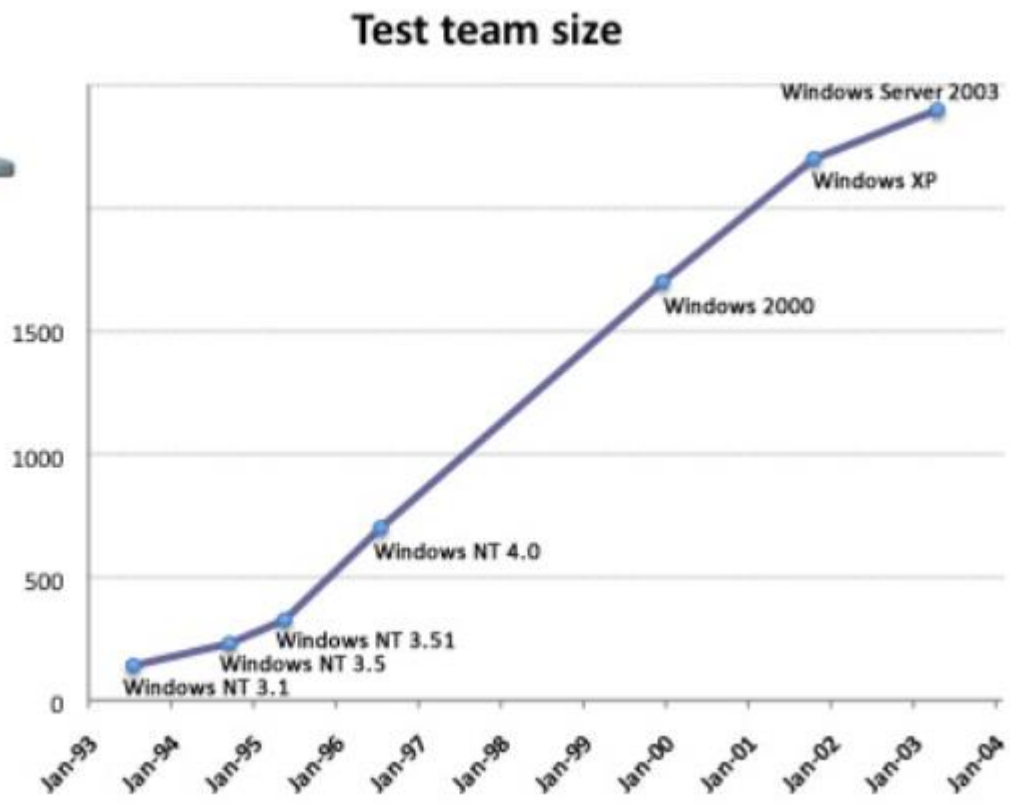
- Software complejo:
  - Muchas líneas de código.
  - Muchos equipos de trabajo.

Complejidad de los sistemas operativos
Windows



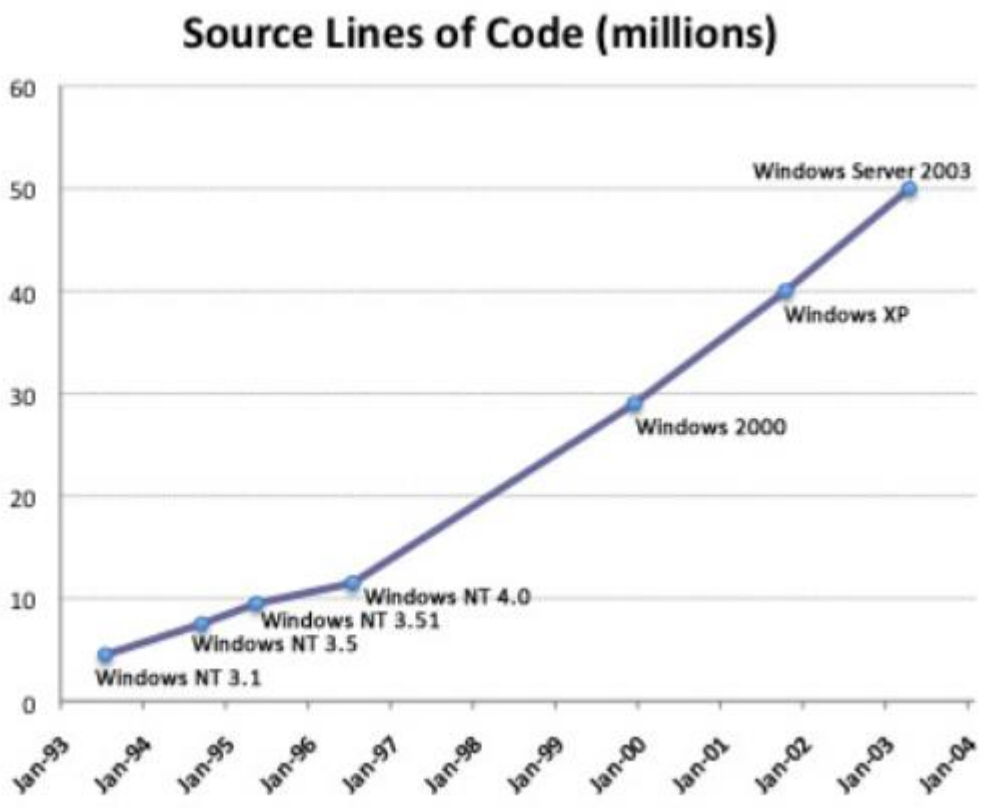
http://www.zdnet.co.uk/reviews/desktop-os/2010/11/20/a-quarter-century-of-windows-40090900/5/#top

Complejidad de los sistemas operativos
Windows



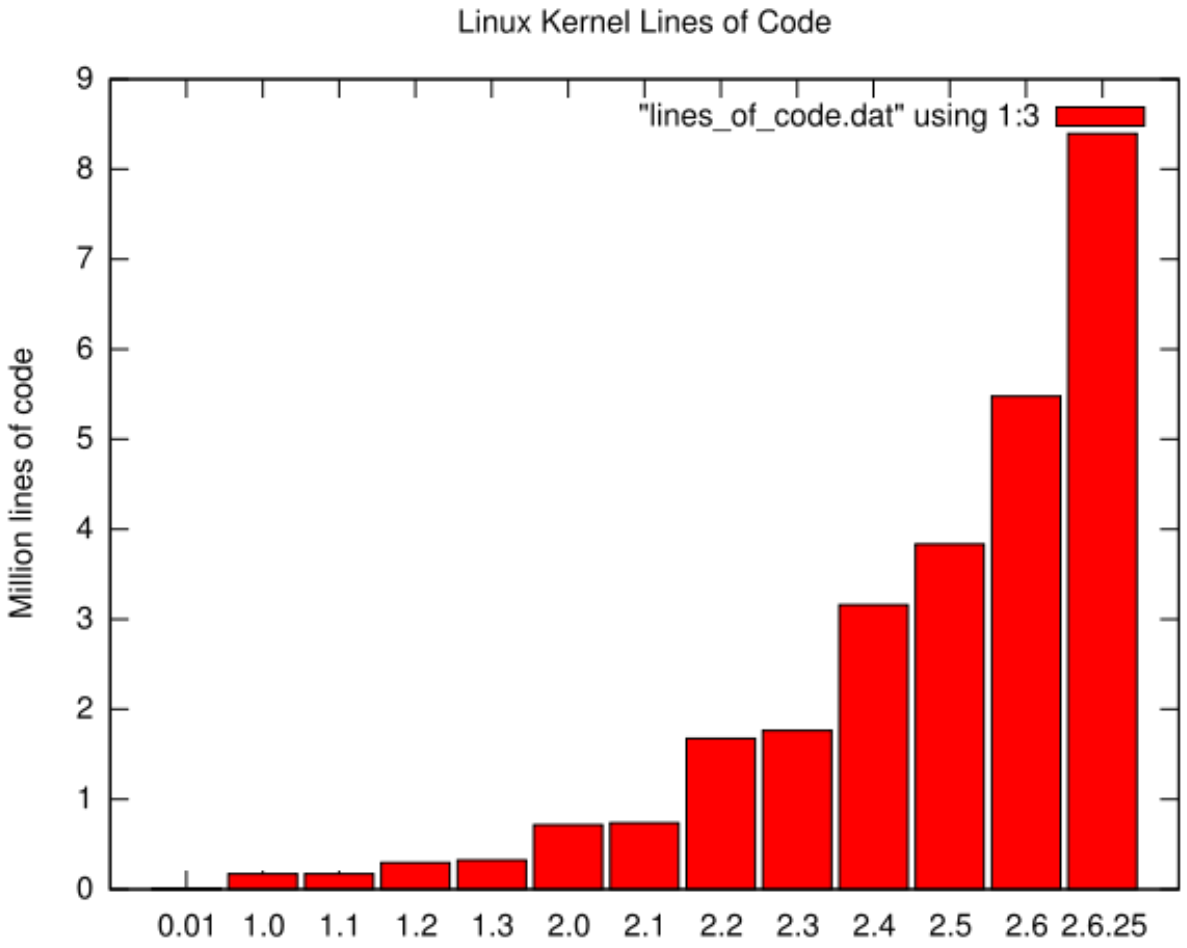
http://www.zdnet.co.uk/reviews/desktop-os/2010/11/20/a-quarter-century-of-windows-40090900/5/#top

Complejidad de los sistemas operativos
Windows



http://www.zdnet.co.uk/reviews/desktop-os/2010/11/20/a-quarter-century-of-windows-40090900/5/#top

Complejidad de los sistemas operativos
Linux



https://secure.wikimedia.org/wikipedia/de/wiki/Datei:Linux_kernel_loc.svg
http://blogs.computerworld.com/204_5_million_lines_of_code_equals_one_great_linux_distribution

5) 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) Delicado

- Software delicado:

(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.

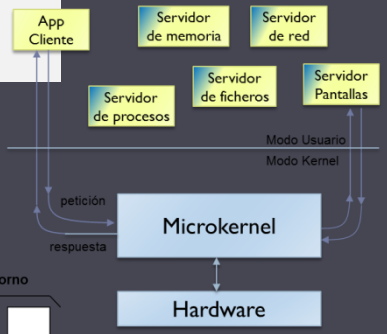
(b) Microsoft Applications: "about 10 - 20 defects per 1000 lines of code during in-house testing, and 0.5 defect per KLOC (KLOC IS CALLED AS 1000 lines of code) in released product (Moore 1992)." He attributes this to a combination of code-reading techniques and independent testing (discussed further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

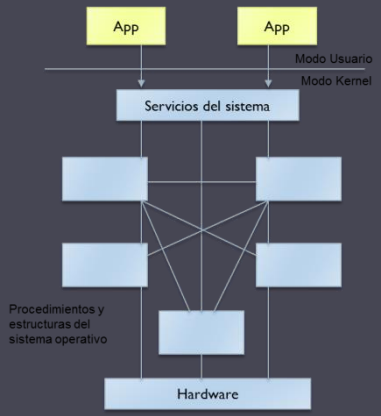
(c) "Harlan Mills pioneered 'cleanroom development', a technique that has been able to achieve rates as low as 3 defects per 1000 lines of code during in-house testing and 0.1 defect per 1000 lines of code in released product (Cobb and Mills 1990). A few projects - for example, the space-shuttle software - 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."

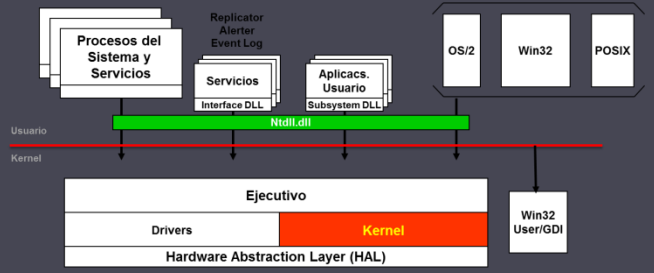
Las principales características de un S.O. **resumen**

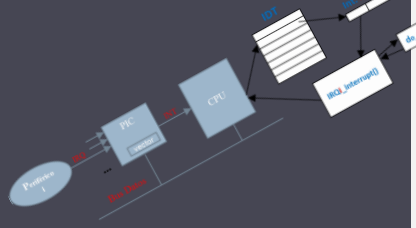
1. Versátil
    1. Portabilidad
2. Adaptativo
3. Multidisciplinar
4. Complejo
5. Delicado

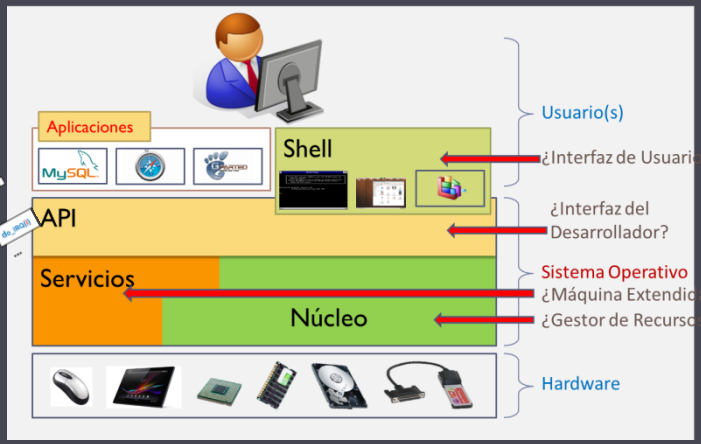
¿Cómo es por dentro?











Contenidos

1. Principales Objetivos de Diseño de un S.O.
2. Estructura del sistema operativo.
3. Ejecución Asíncrona.

Objetivos de Diseño de un S.O.

- 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
  - Código inteligible y documentado
- Extensibilidad

http://www.cc.gatech.edu/~pwh/

Objetivos de Diseño de un S.O.

- Algunos Objetivos de Diseño son Antagónicos entre sí:
  - Rendimiento (eficiencia y velocidad) vs Seguridad
- Por tanto, siempre "hay que seleccionar una terna de objetivos de diseño" (*Butter Lampson*)

http://www.cc.gatech.edu/~pwh/

Contenidos

1. Principales Objetivos de Diseño de un S.O.
2. Estructura del sistema operativo.
    1. Diseño Conceptual
    2. Diseño Arquitectural (Arquitecturas de S.O.)
3. Ejecución Asíncrona.

Contenidos

1. Principales Objetivos de Diseño de un S.O.
2. Estructura del sistema operativo.
    1. Diseño Conceptual
    2. Diseño Arquitectural (Arquitecturas de S.O.)
3. Ejecución Asíncrona.

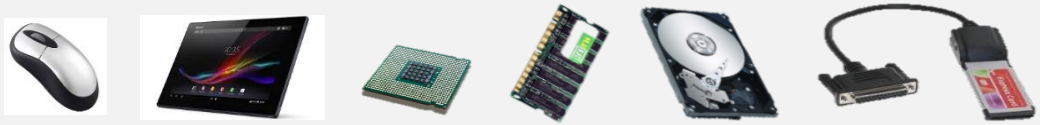
¿Qué tiene que hacer un S.O.?

Aplicaciones

Usuario(s)

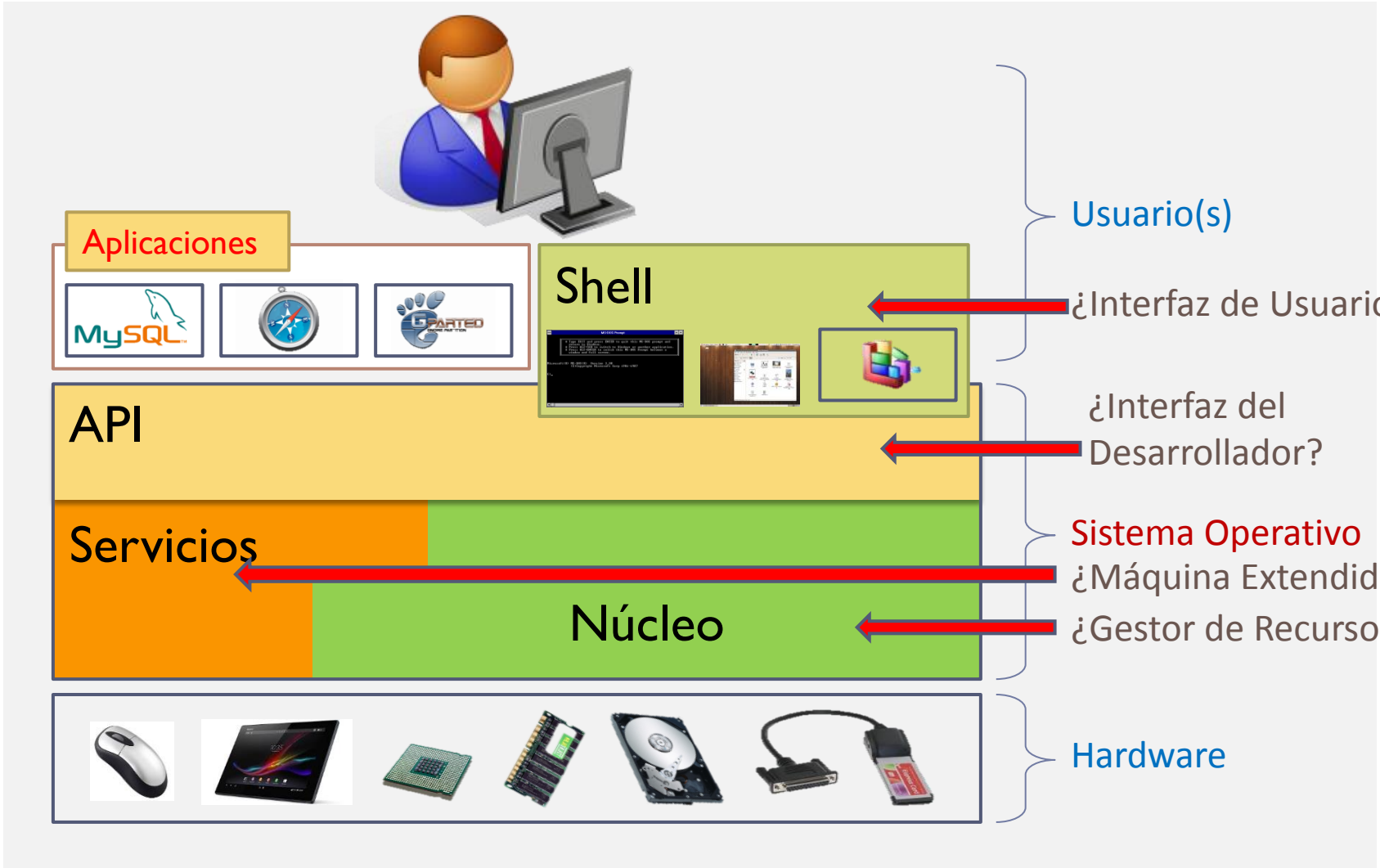
- Proporcionar Interfaz al Usuario (s)
- Máquina Extendida
- Gestionar los recursos del equipo

Sistema Operativo



Hardware

Diseño Conceptual de un S.O.



Diseño Conceptual de un S.O.

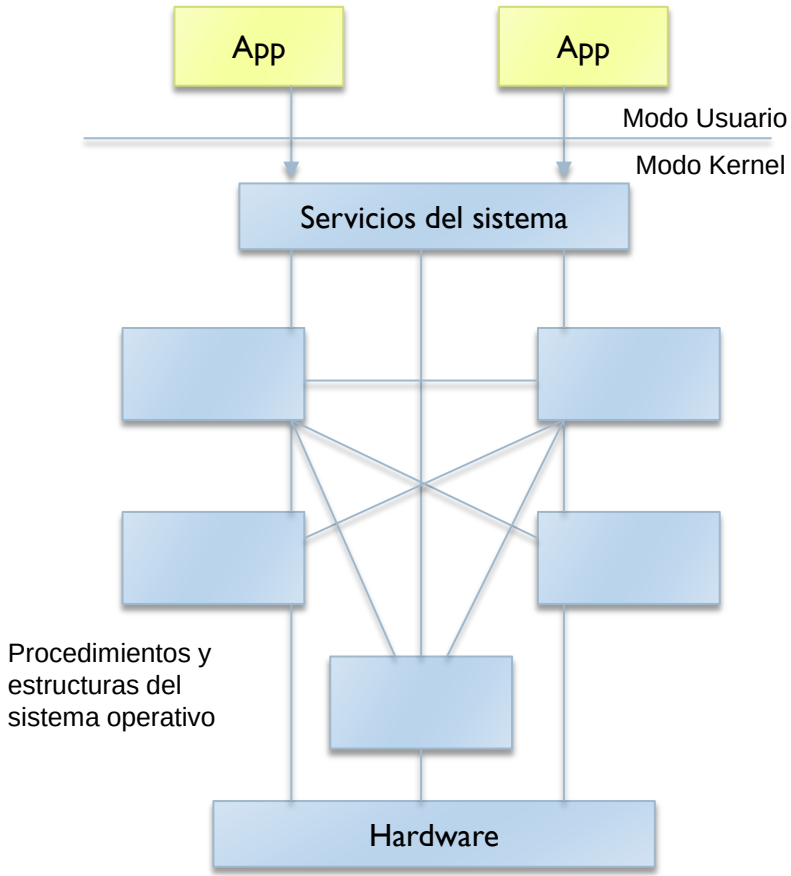
- Intérprete de Comandos (shell):
  - Ejecuta Órdenes de Usuario
  - Proporciona los resultados de esas órdenes
- Interface de Programación de Aplicaciones (API):
  - Proporciona las invocaciones a las Llamadas al Sistema
  - Gestiona el paso de Modo Usuario a Modo Kernel
- Servicios:
  - Implementa las Llamadas al Sistema (y su funcionalidad)
  - Gestiona los dispositivos Hardware
- Núcleo / Kernel / Core:
  - Gestor del S.O.
  - Gestiona Procesos e Interrupciones (INT)

Contenidos

1. Principales Objetivos de Diseño de un S.O.
2. Estructura del sistema operativo.
    1. Diseño Conceptual
    2. Diseño Arquitectural (Arquitecturas de S.O.)
3. Ejecución Asíncrona.

Arquitectura del S.O.
Monolítico (macrokernel)

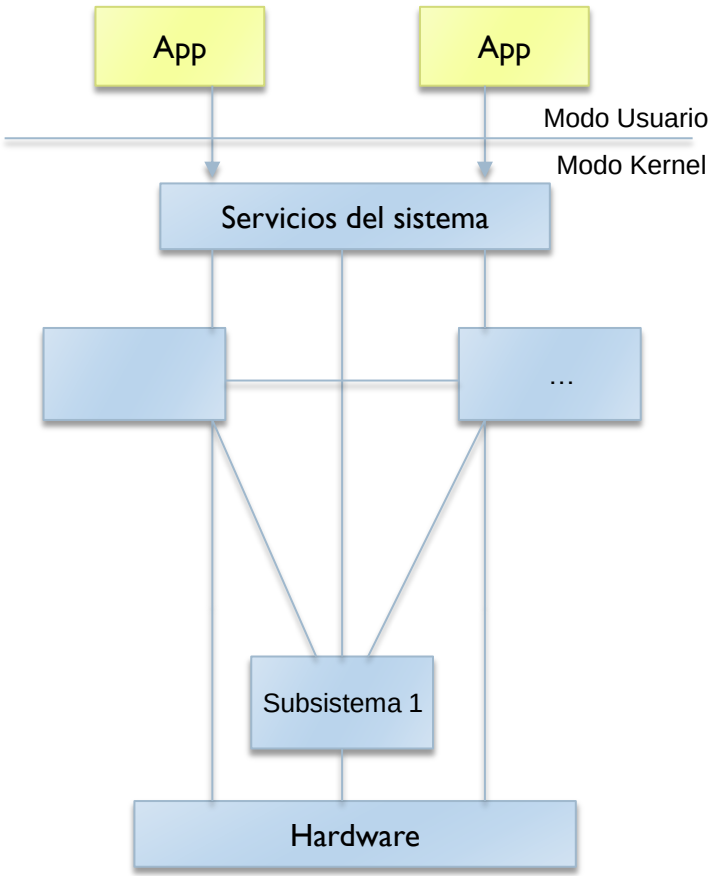
- Sistema monolítico.
- No estructurado.
- Desde cualquier punto del código se puede acceder a cualquier variable o función de otra parte del núcleo (*kernel*)
- Ejemplo:
  - MS-DOS, DR-DOS, UNICS



En monolítico:
- [v] muy rápido en ejecución
- [i] muy difícil de mantener y muy sensible a errores

Arquitectura del S.O.
En subsistemas

- Sistema monolítico, compuesto de subsistemas lógicos que ofrecen interfaces bien definidas como puntos de entrada.
- Se agrupan procedimientos y estructuras de datos relacionadas.
- Ejemplo:
  - Linux

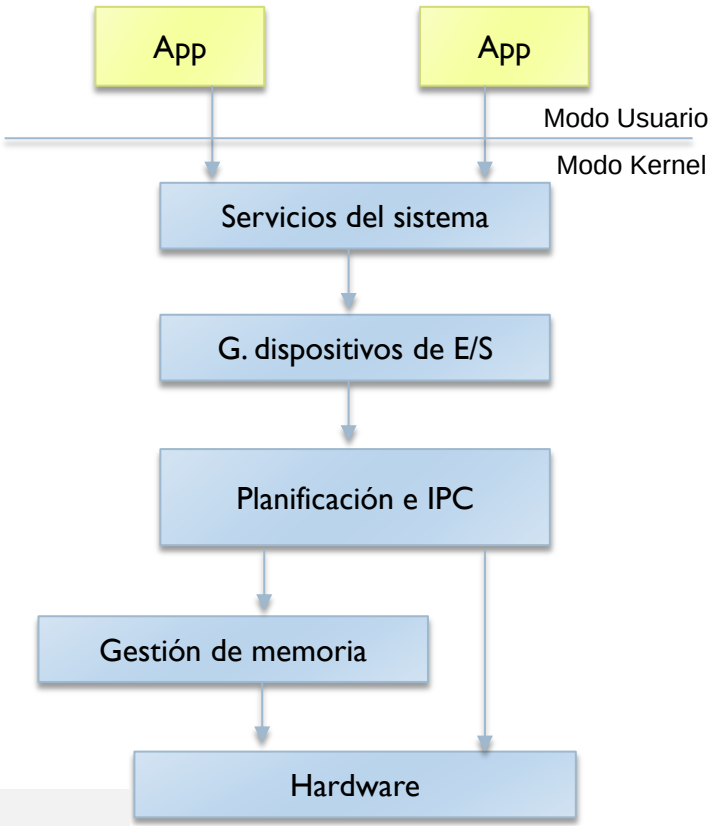


En subsistemas:
- [v] diseño algo más claro
- [i] bastante sensible a errores y difícil de mantener

Arquitectura del S.O.

Por capas

- Binario monolítico aunque codificado estructurado de forma lógica en capas.
- Cada capa sólo se relaciona con las adyacentes:
  - Proporciona SERVICIOS a capa superior
  - Requiere SERVICIOS del interfaz de la capa del nivel inferior
- Ejemplo:
  - THE (Dijkstra)
  - Multics, que añadió a la noción de capa la idea de anillos de privilegios
  - OS/2 (IBM)

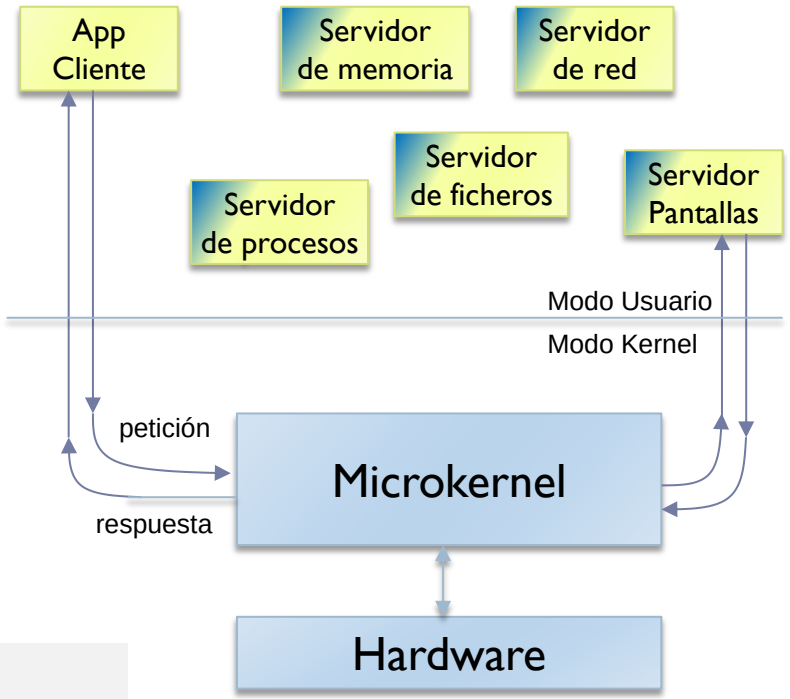


Por Capas:

- [v] modularidad y ocultación (encapsulación) de la información
- [i] sobrecarga de procesamiento por el tratamiento de los SERVICIOS

Arquitectura del S.O.
Microkernel

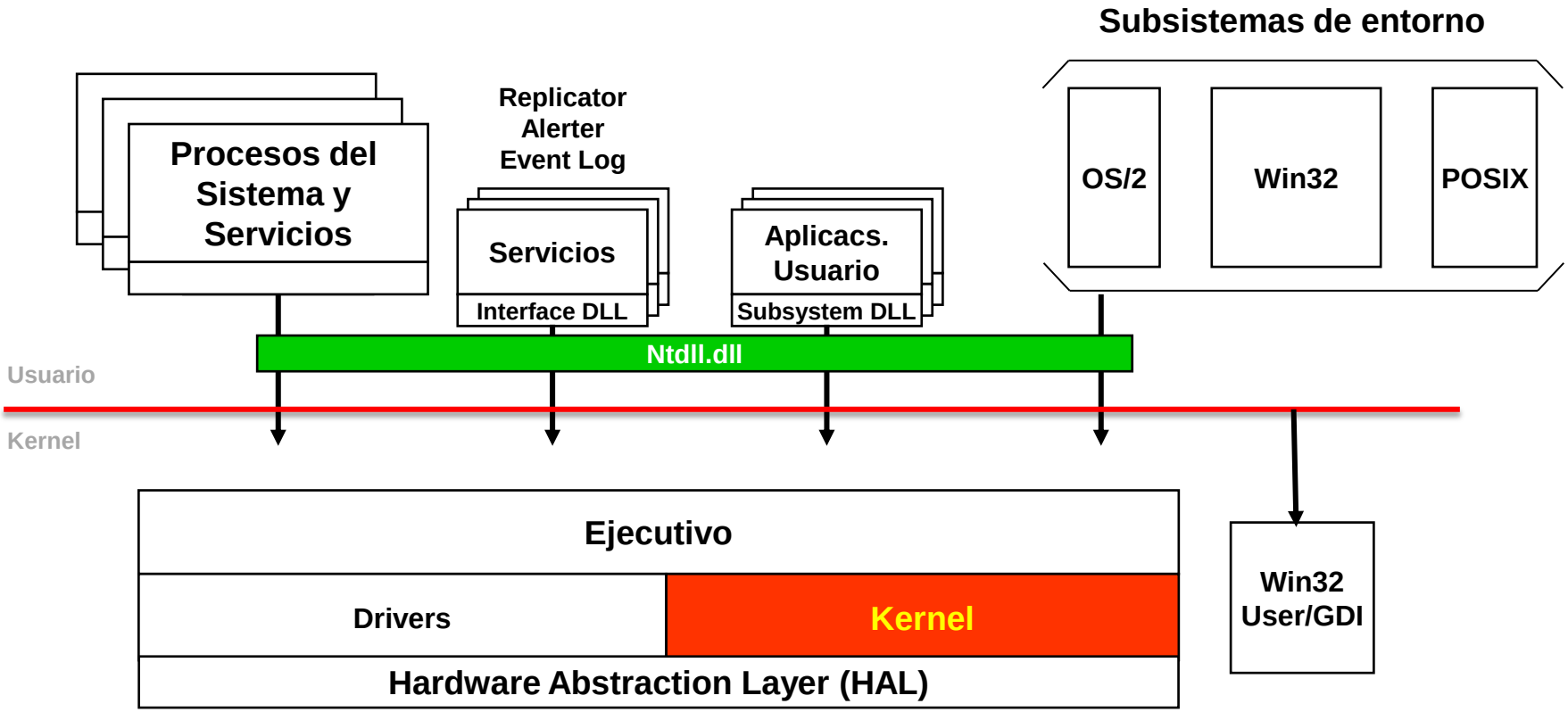
- Además de estructurado, los principales componentes se ejecutan como procesos servidores, fuera del kernel.
- El microkernel tiene:
  - Planificación y gestión de procesos.
  - Gestión de memoria virtual básica.
  - Comunicación entre procesos básica.
- Ejemplo:
  - Match, QNX, Minix, L4, etc.



Por Microkernel:
- [v] más fiable y estructurado
- [i] sobrecarga pérdida de rendimiento por sobrecarga de procesamiento y menor seguridad

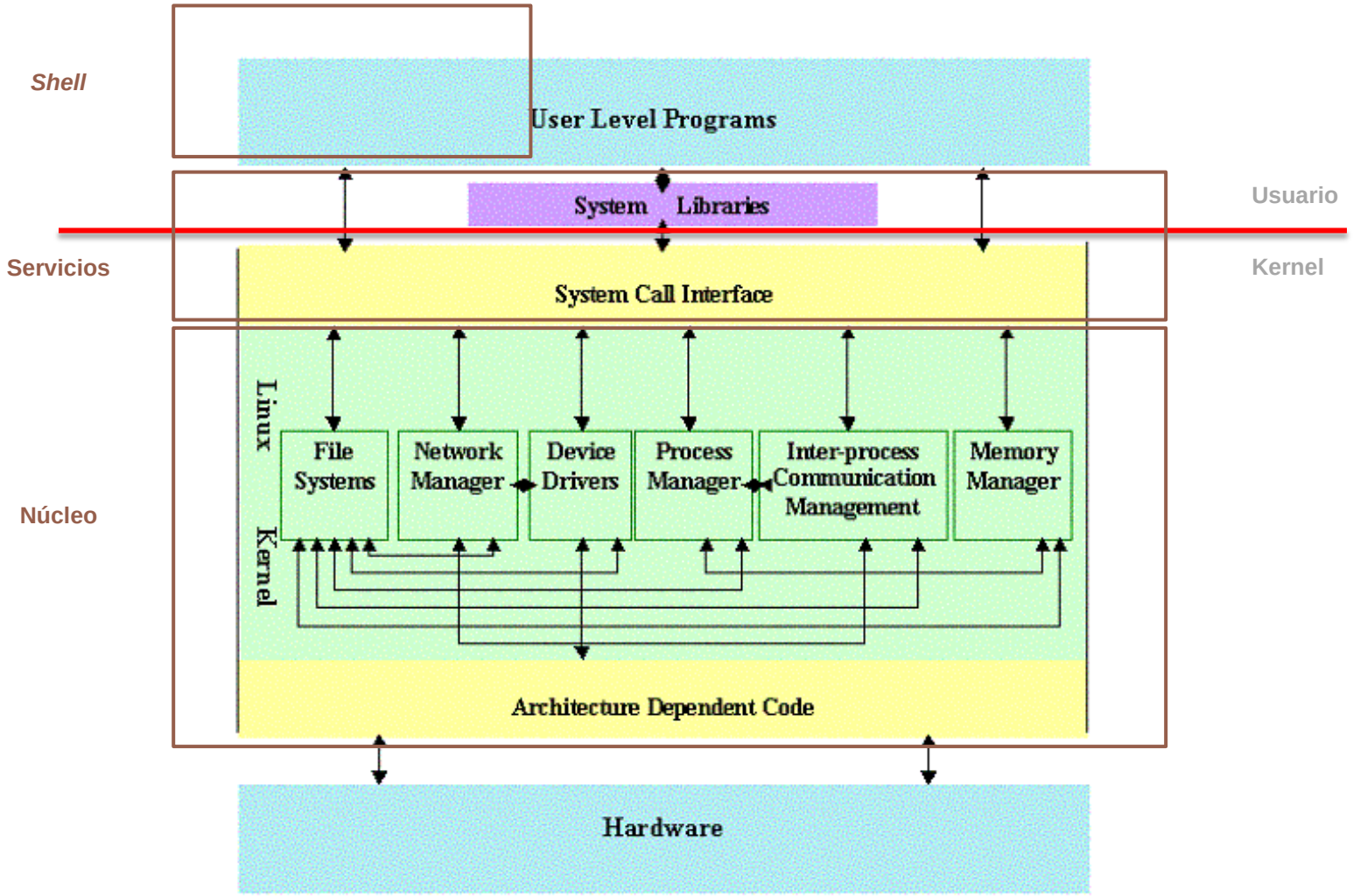
Arquitecturas del SS.OO. Reales

Arquitectura de la Familia Windows 2000



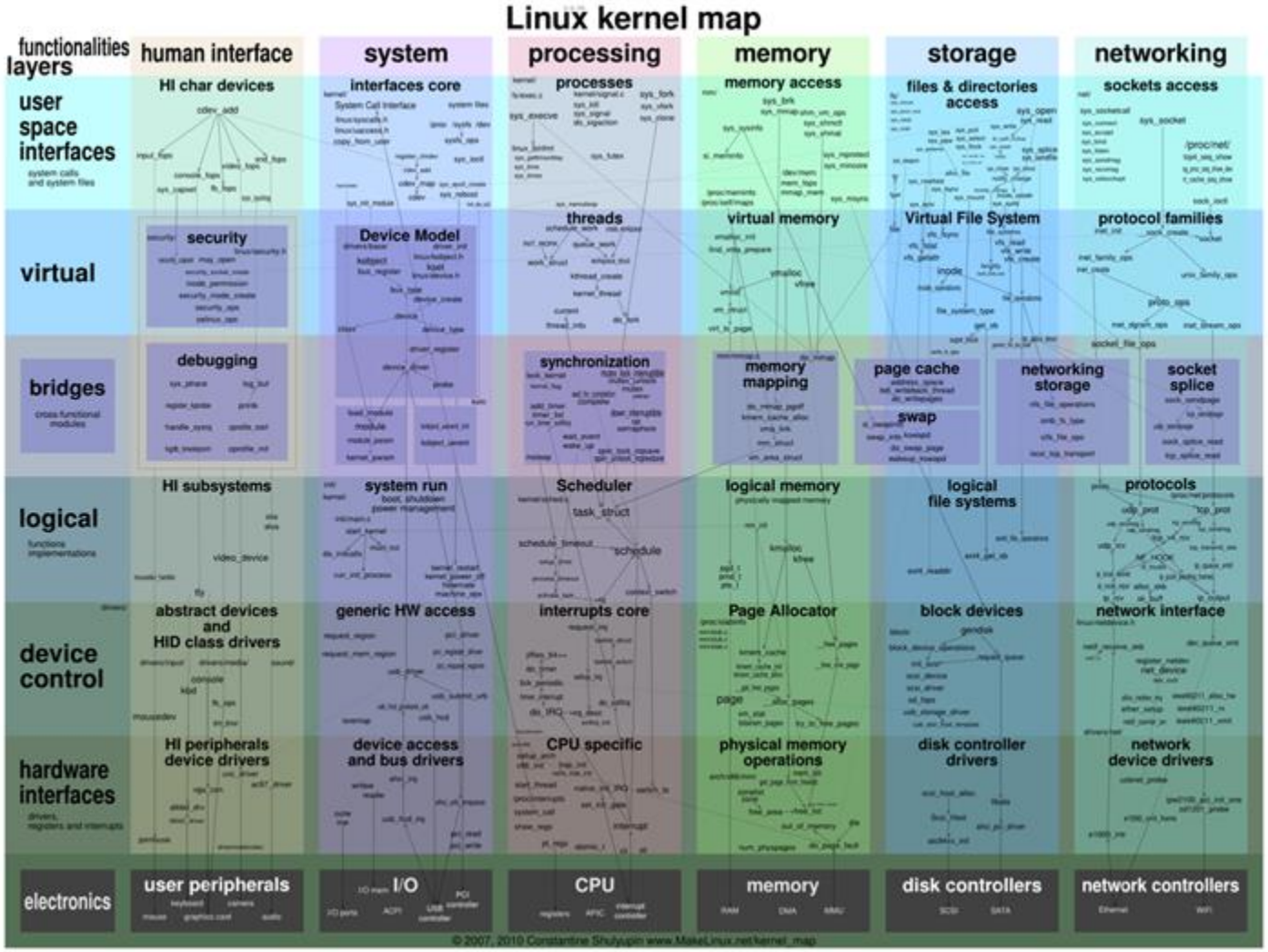
Arquitecturas de SS.OO. Reales

Linux (Esquema Simplificado)



SS.OO. Reales

Linux (versión 'menos' simplificada)

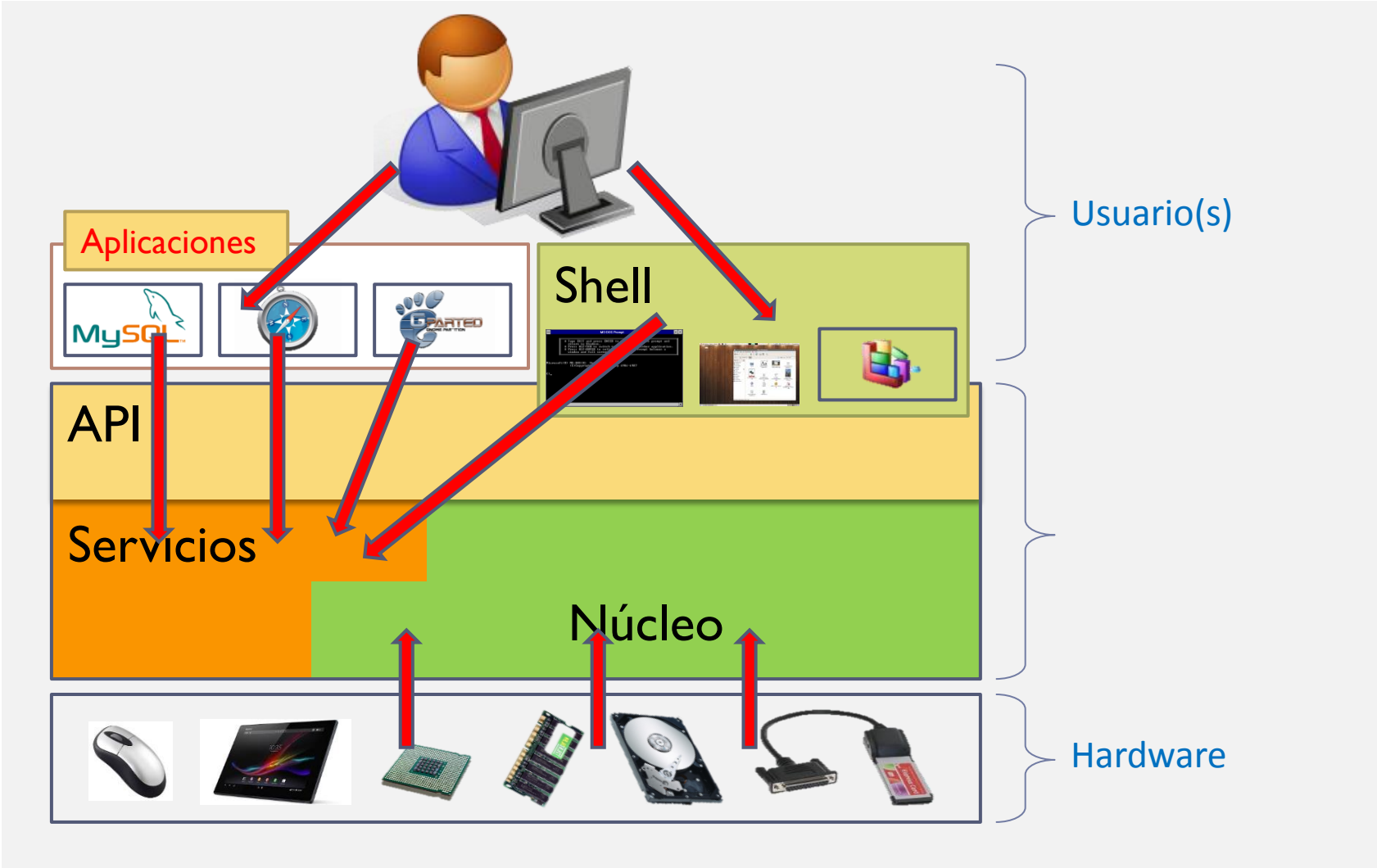


http://www.makelinux.net/kernel_map.shtml

Contenidos

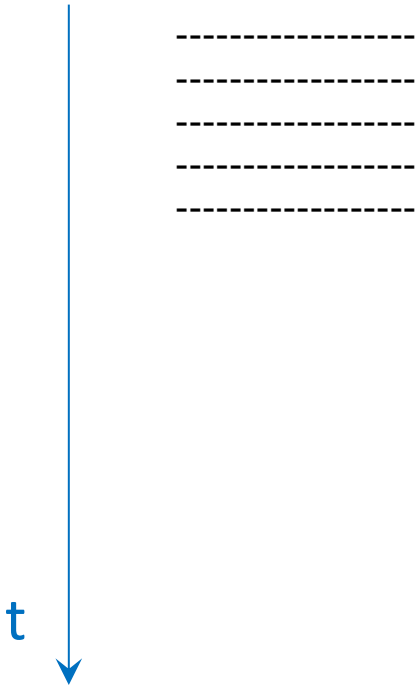
1. Principales Objetivos de Diseño de un S.O.
2. Estructura del sistema operativo.
3. Ejecución Asíncrona.

Comunicación interna en S.O.



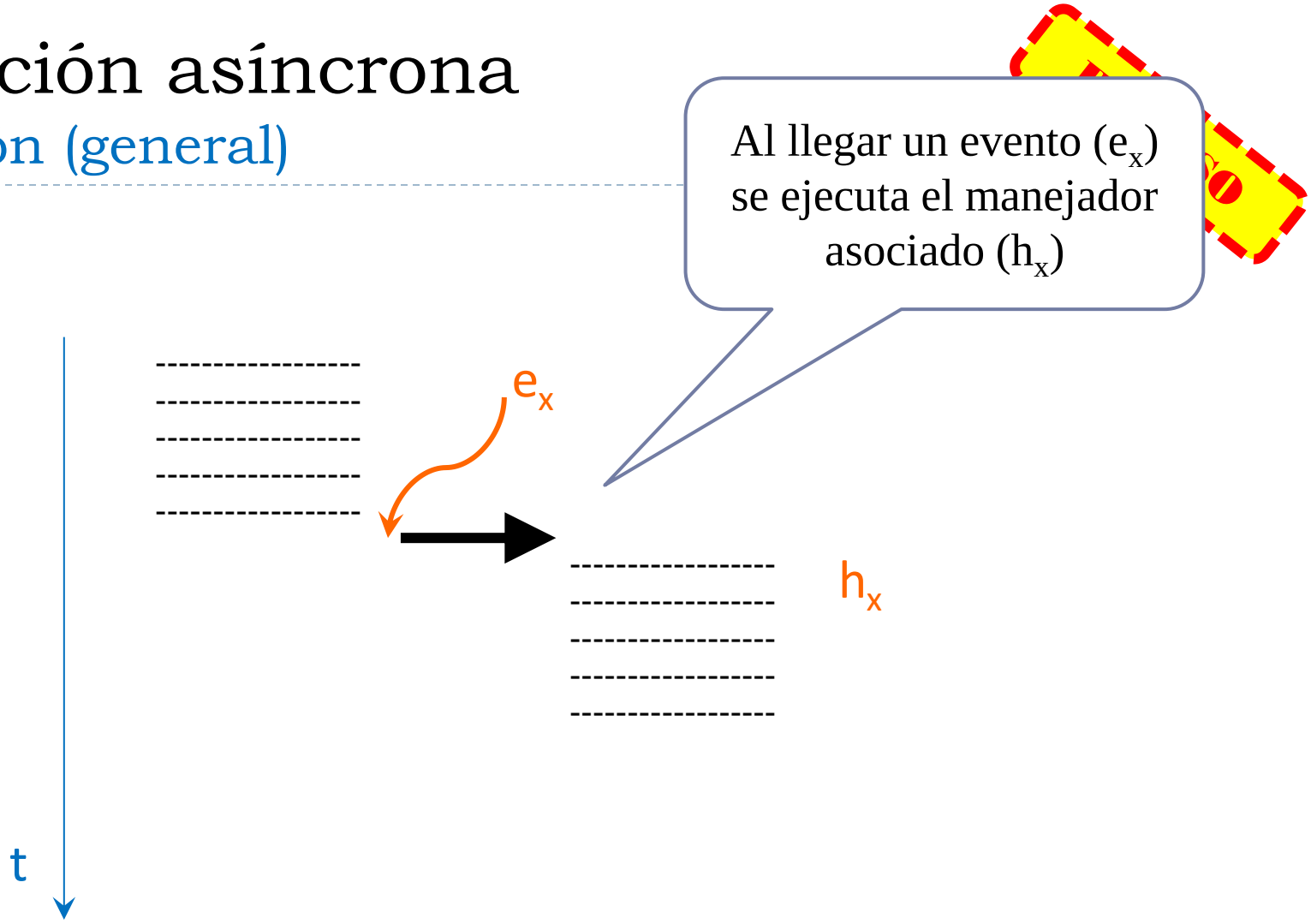
Ejecución asíncrona

ejecución (general)



Ejecución asíncrona

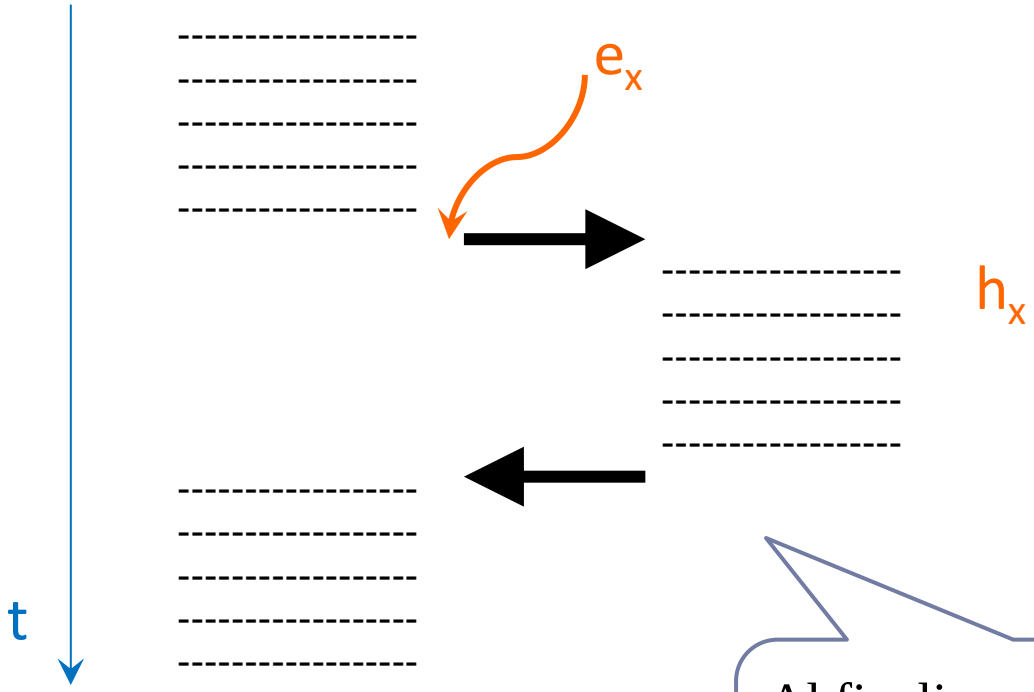
ejecución (general)



Ejecución asíncrona

ejecución (general)

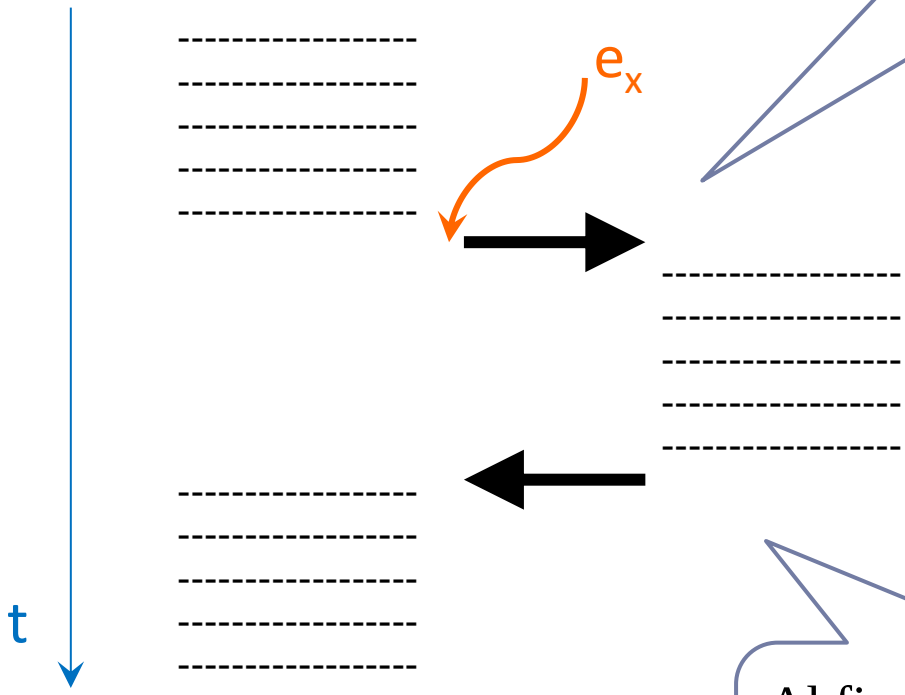
REPASO



Al finalizar el manejador, se continúa la ejecución por donde fue interrumpida

Ejecución asíncrona

ejecución (general)



Al llegar un evento (e_x) se ejecuta el manejador asociado (h_x)

Al finalizar el manejador, se continúa la ejecución por donde fue interrumpida

Ejecución asíncrona
código (general)

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

1) Asociar el manejador (**`handler1`**) al evento.

Ejecución asíncrona
código (general)

```
void handler1 ( … )
{
}

…
```

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

2) Codificar la función manejador (**handler1**) que tratará el evento.

1) Asociar el manejador (**handler1**) al evento.

Ejecución asíncrona
código (general)

```
int global1;
…
```

```
void handler1 ( … )
{
}

…
```

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

3) Dado que utilizan el mismo espacio todas las funciones del Kernel, es más eficiene usar Constantes y Variables Globales.

2) Codificar la función manejador (**handler1**) que tratará el evento.

1) Asociar el manejador (**handler1**) al evento.

Ejemplo de ejecución asíncrona

Señales (interrupciones software)

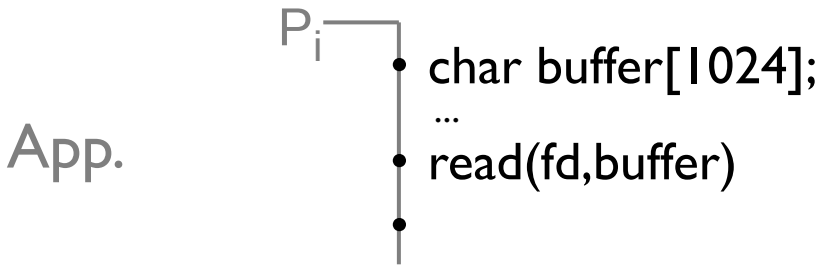
signal.h

```
#include<stdio.h>
#include<signal.h>
#include<unistd.h>

void sig_handler (int signo)
{
    if (signo == SIGINT)
        printf("received SIGINT\n");
}

int main(void)
{
     if (signal(SIGINT, sig_handler) == SIG_ERR)
         printf("\ncan't catch SIGINT\n");

      sleep(60); // simula un proceso largo.

     return 0;
}
```

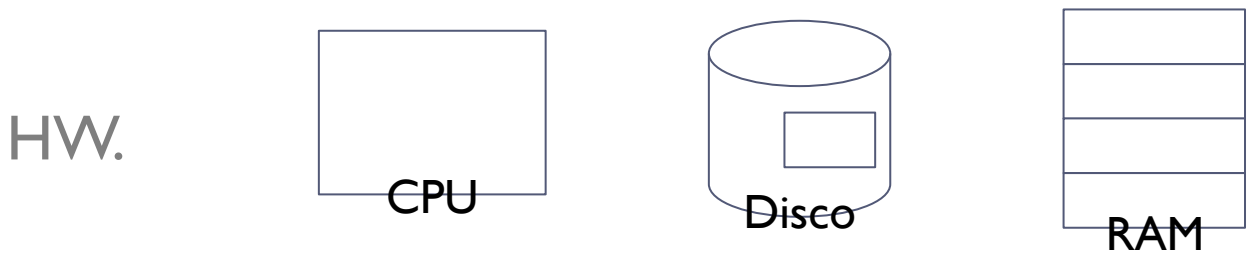

http://www.thegeekstuff.com/2012/03/catch-signals-sample-c-code/

Ejecución asíncrona
ejemplo simplificado

Tema 2

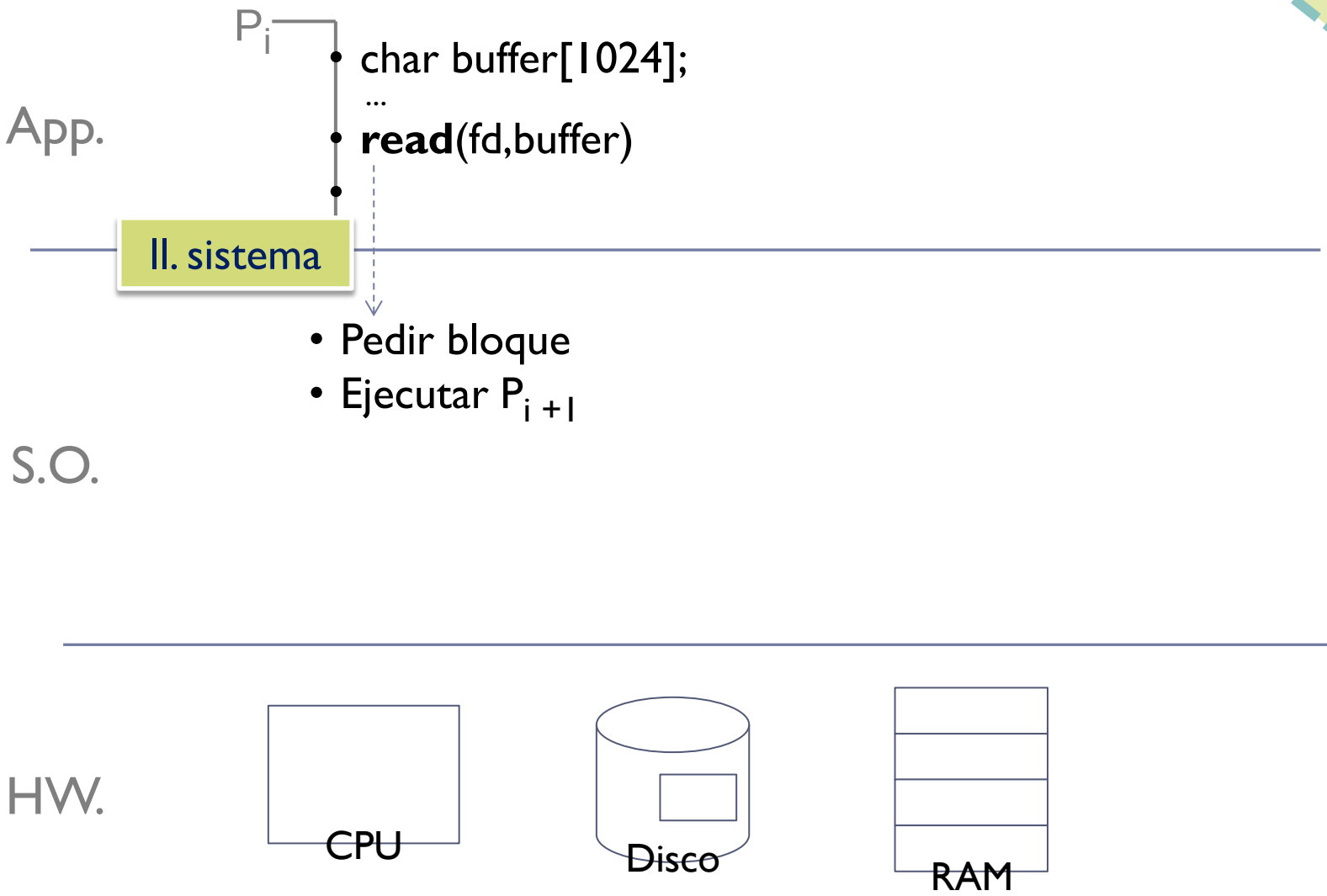


S.O.



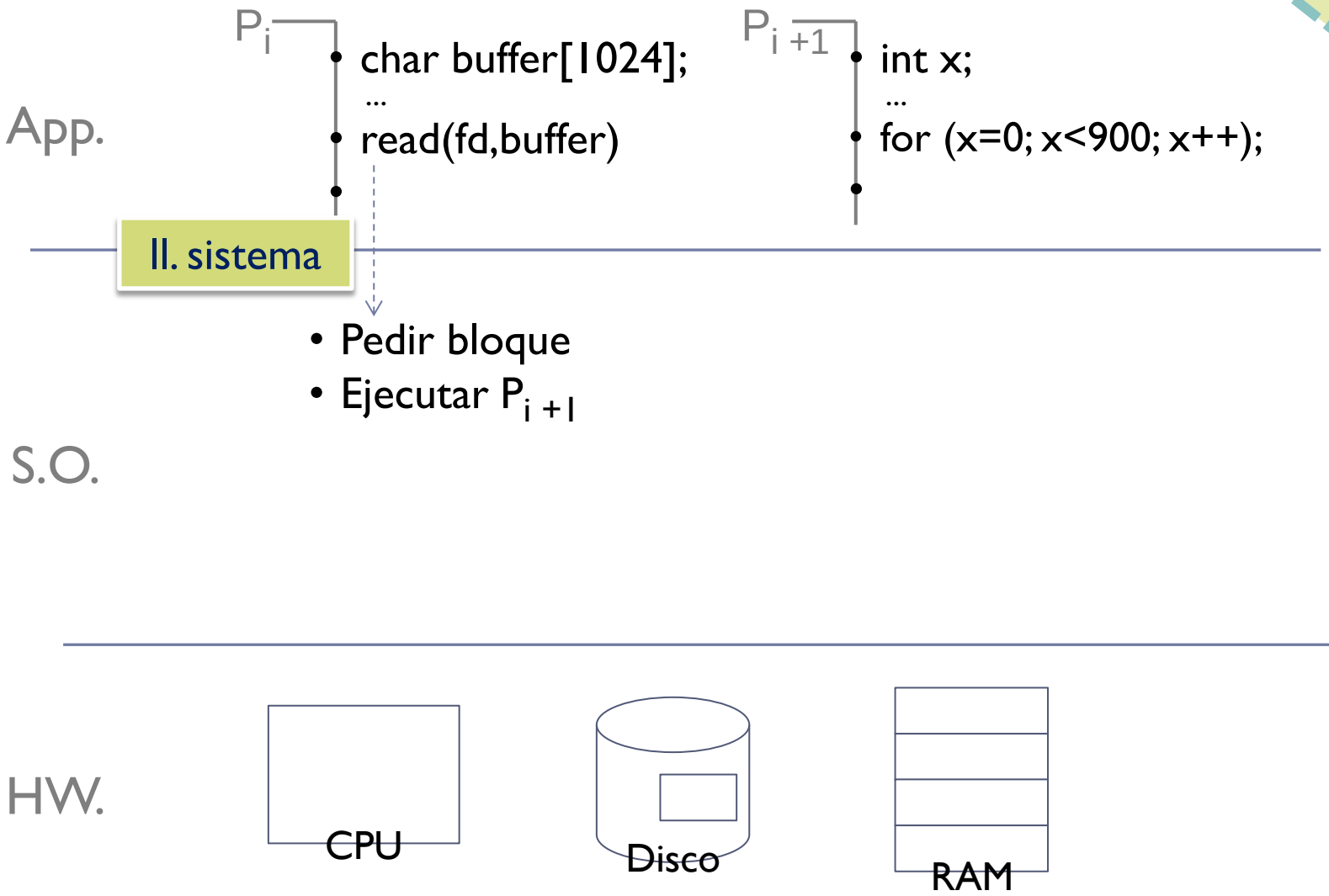
Ejecución asíncrona
ejemplo simplificado

Tema 2



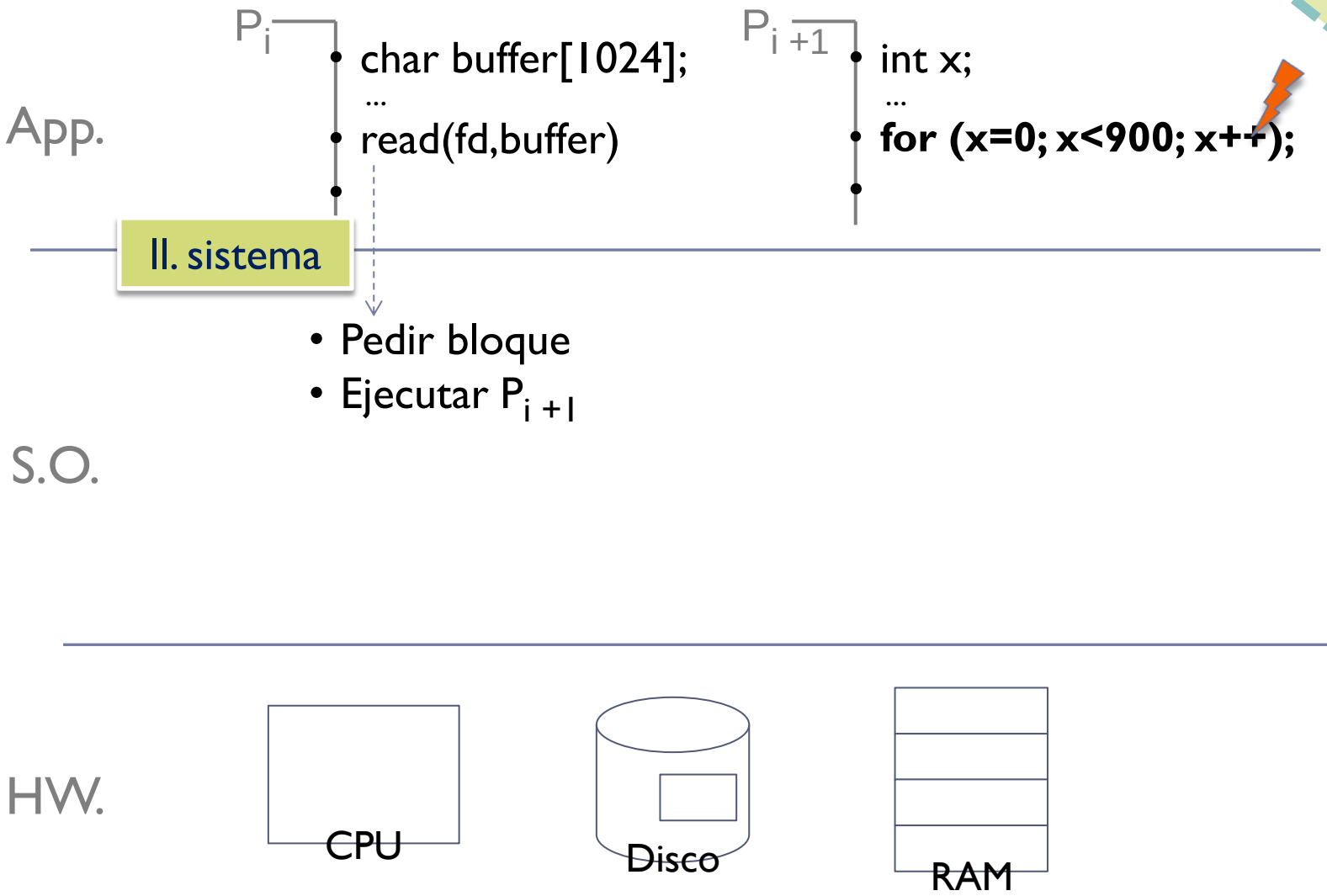
Ejecución asíncrona
ejemplo simplificado

Tema 2



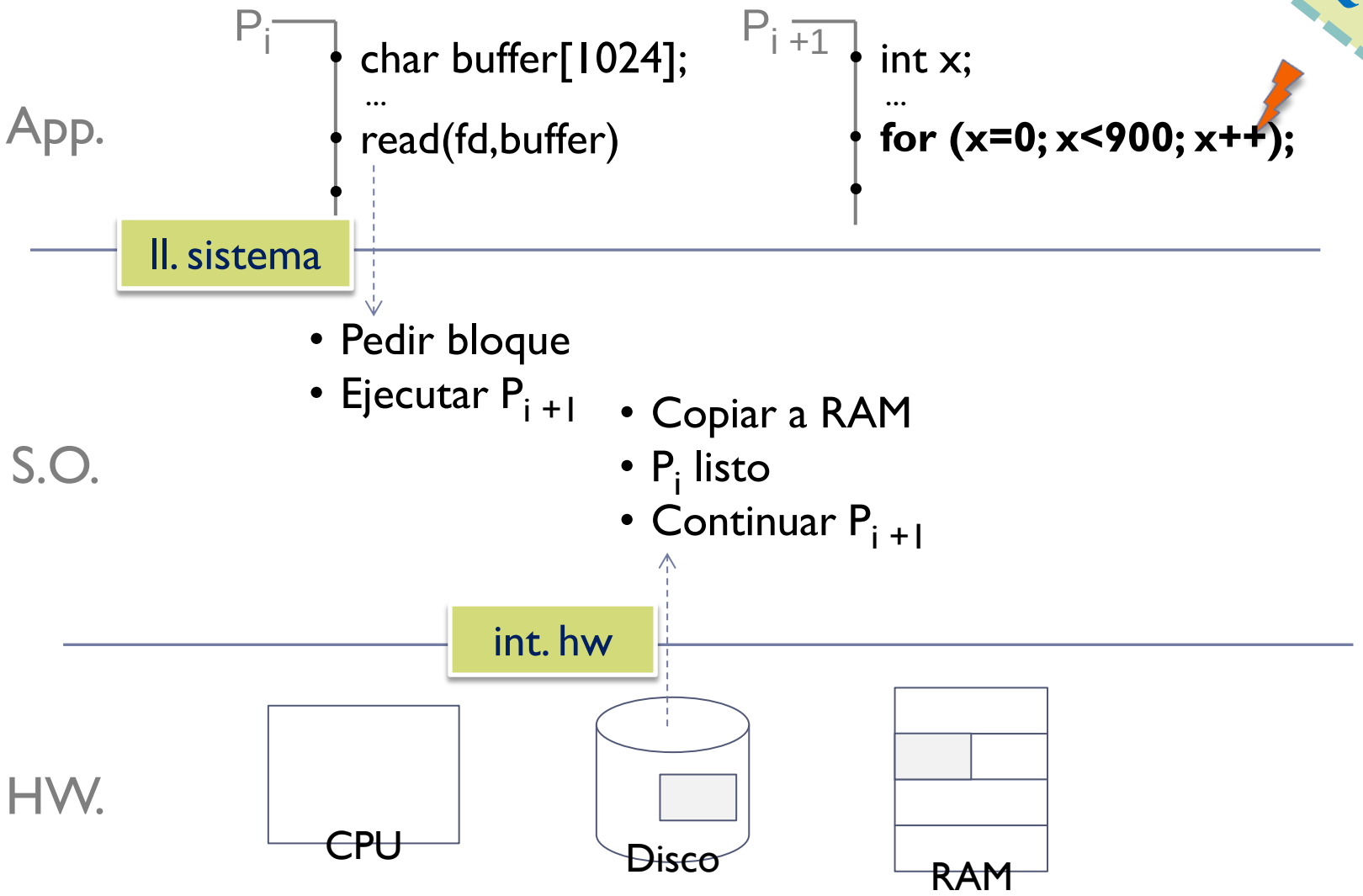
Ejecución asíncrona
ejemplo simplificado

Tema 2



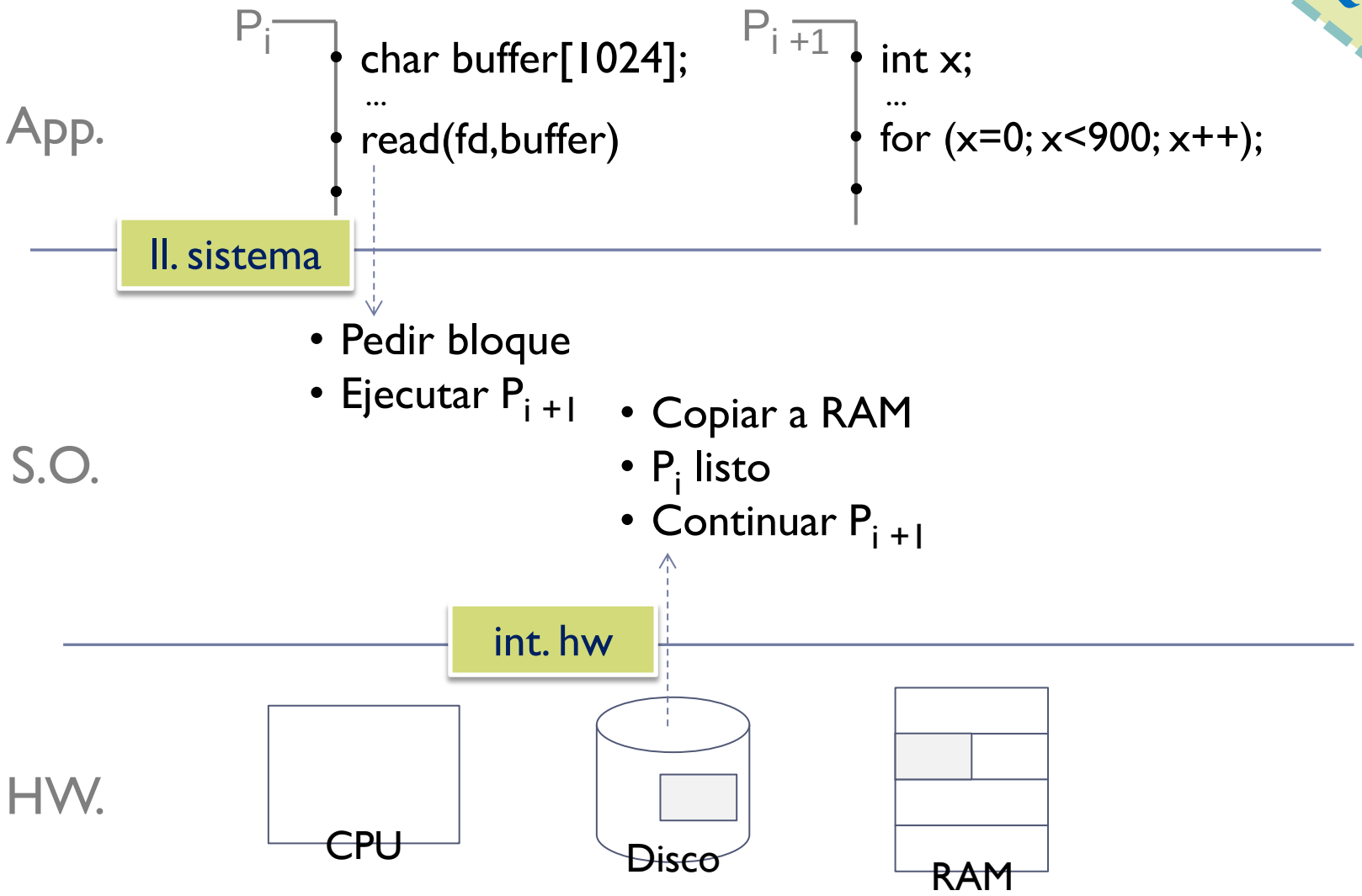
Ejecución asíncrona
ejemplo simplificado

Tema 2



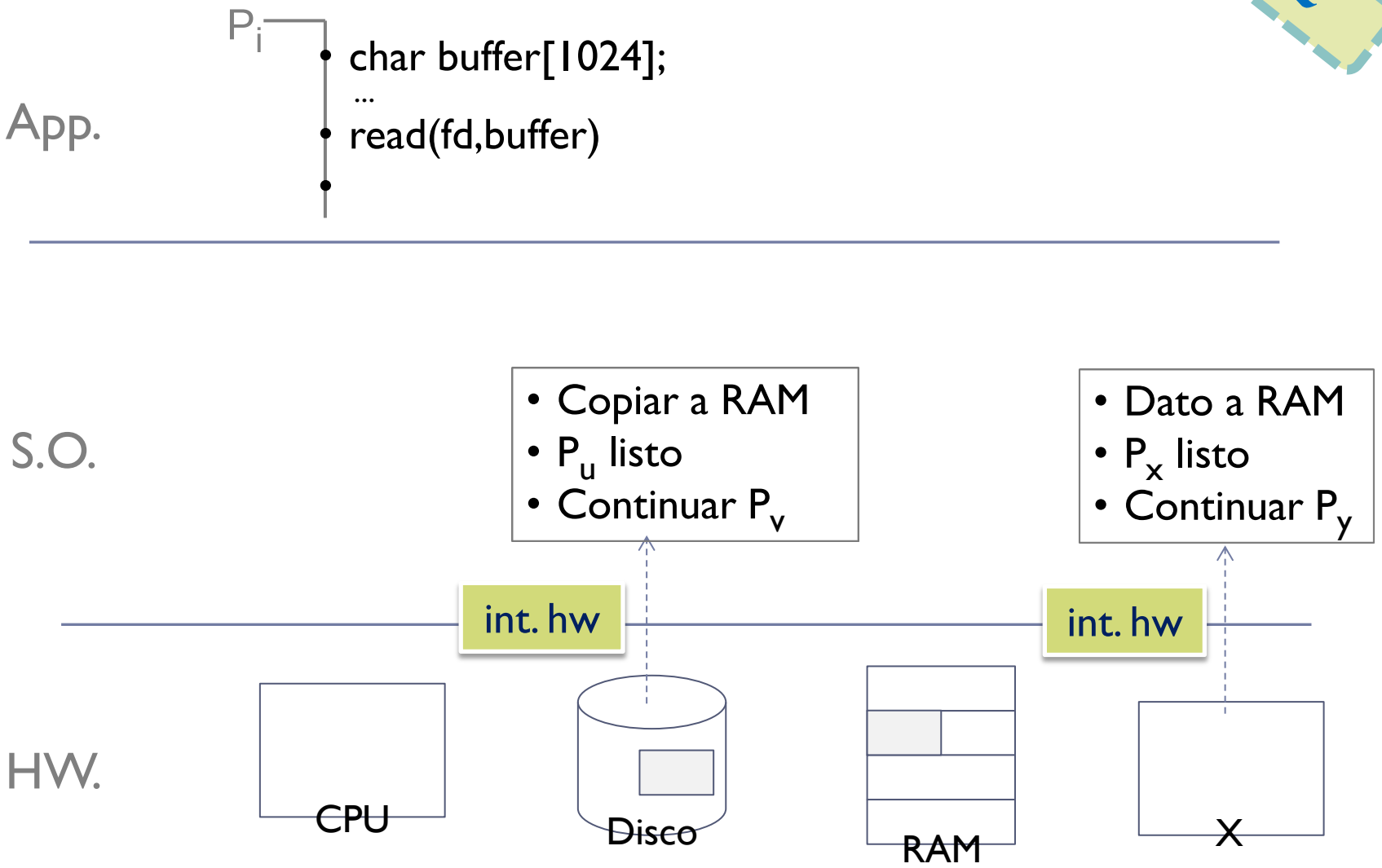
Ejecución asíncrona
ejemplo simplificado

Tema 2



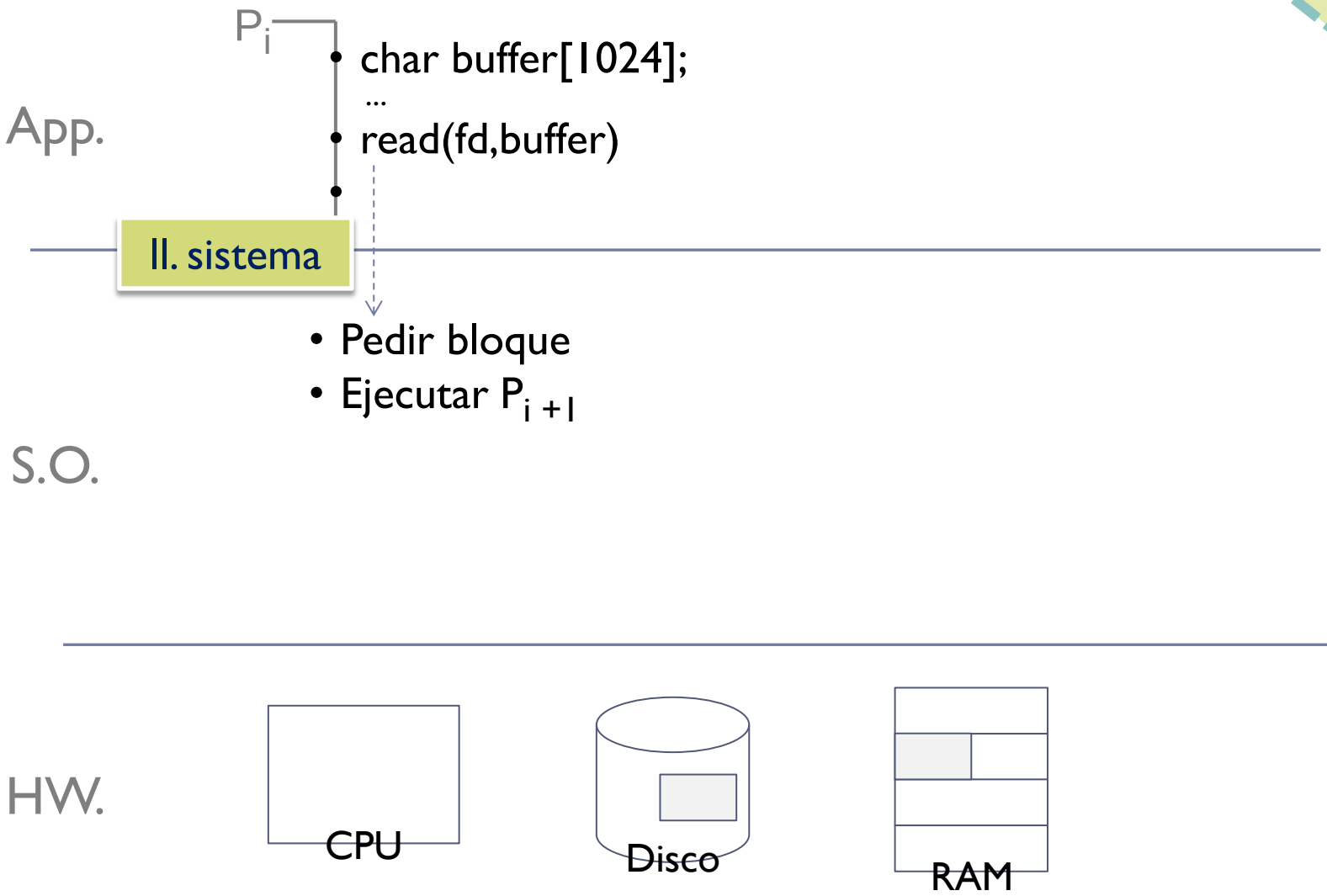
Ejecución asíncrona
ejemplo simplificado

Tema 2



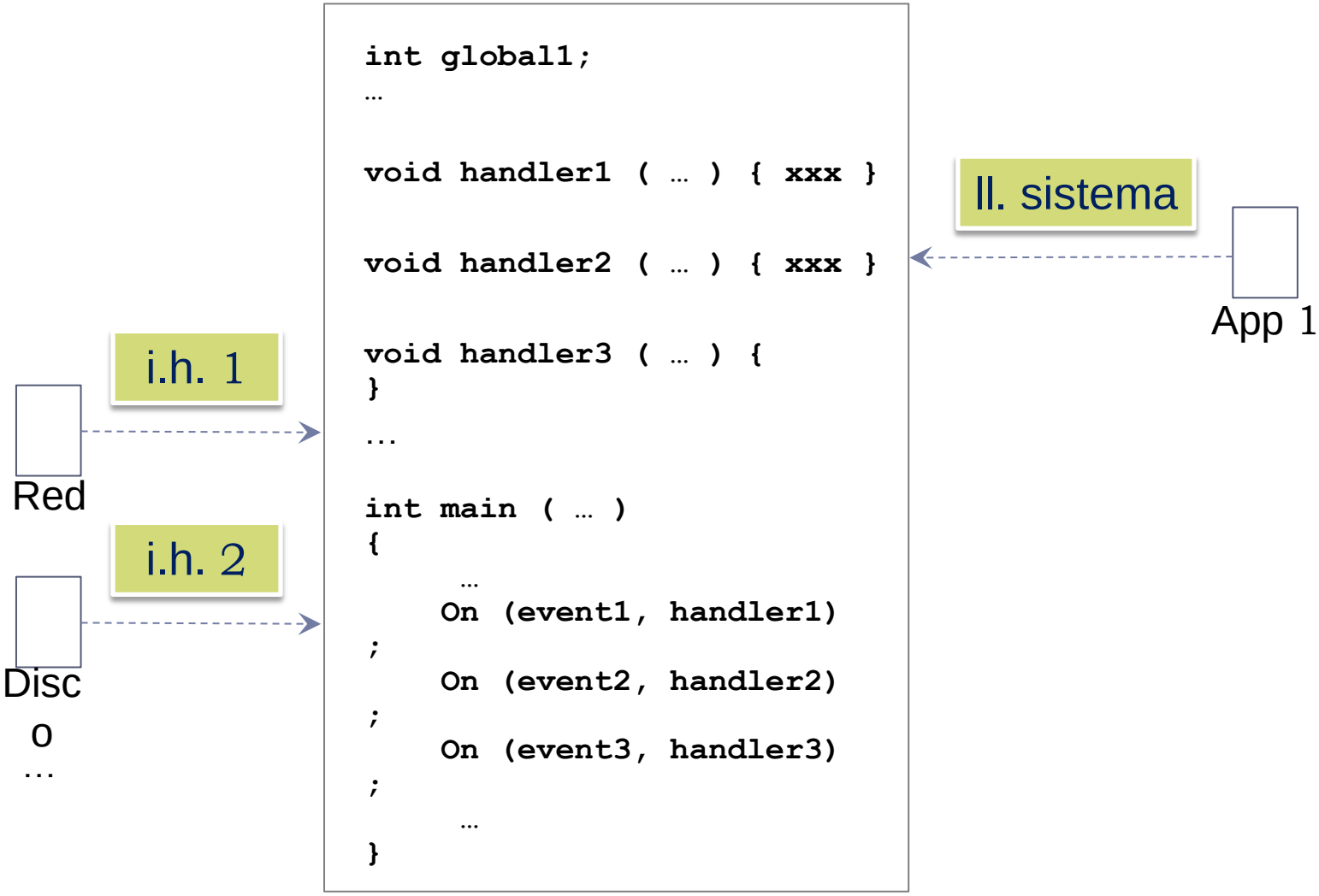
Ejecución asíncrona
ejemplo simplificado

Tema 2

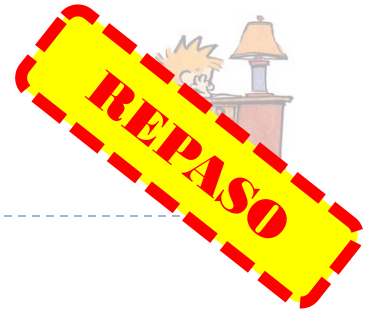


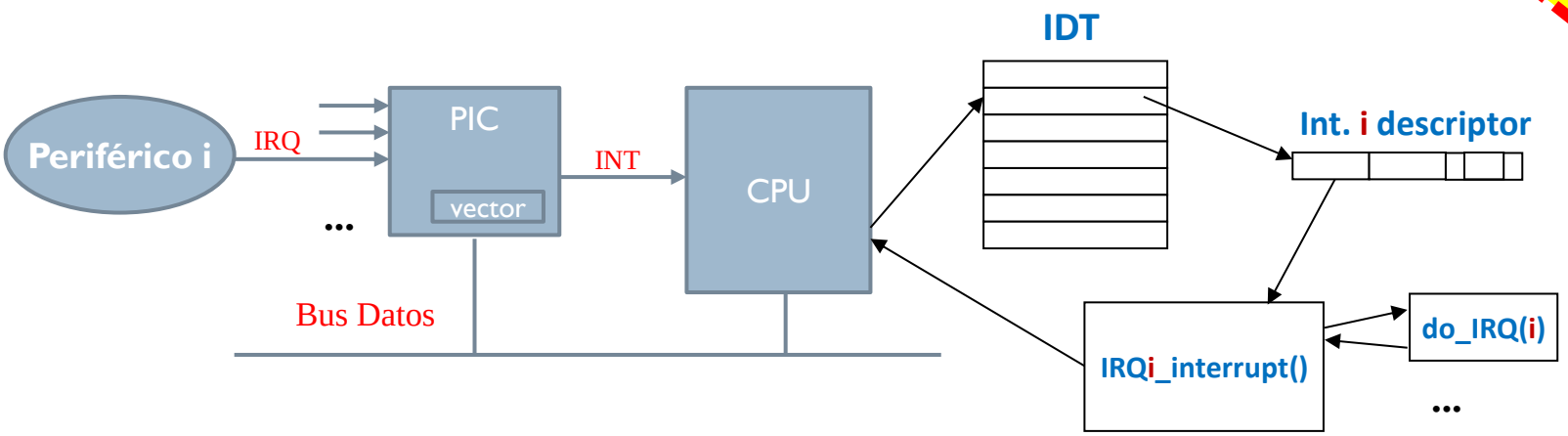
Estructura base del sistema operativo

código (general)



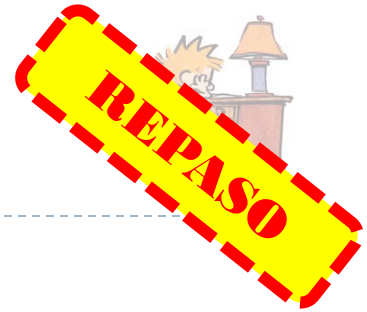
Interrupciones hardware
repaso (1/4)

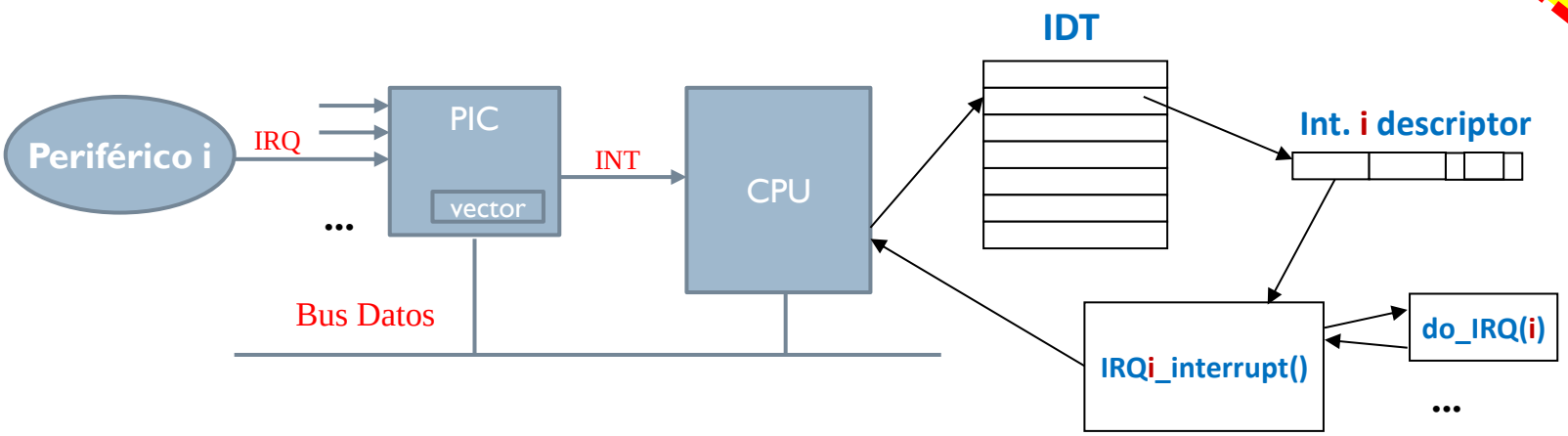




- Cada periférico (capaz de generar una petición de interrupción) dispone de una línea denominada IRQ (*Interrupt ReQuest*)
  - Puede haber múltiples dispositivos en una línea, se precisa muestreo para conocer el peticionario
- Todas las líneas se conectan a un PIC (*Programmable Interrupt Controller*)
  - Actualmente se usa un APIC (*Advanced Programmable Interrupt Controller*)
- El PIC se conecta a la CPU por una línea de aviso de interrupción pendiente (INT)
- El PIC y la CPU también están conectados por el bus de datos

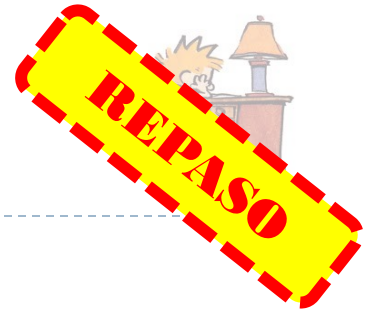
Interrupciones hardware
repaso (2/4)

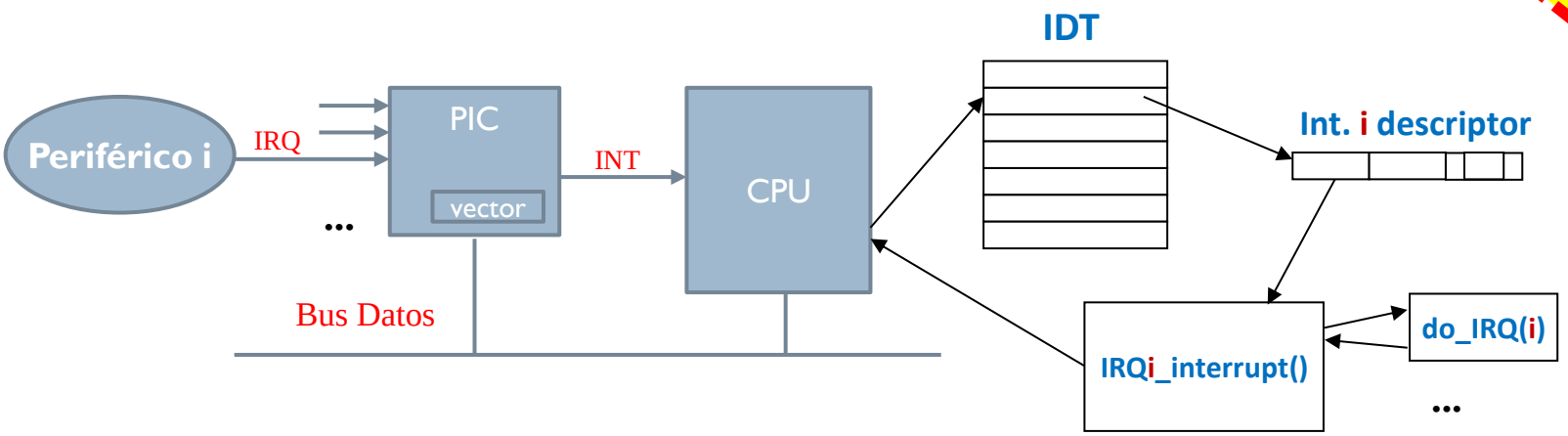




- El PIC monitoriza las líneas de IRQ esperando la llegada de una señal
- Si llega una señal entonces:
  - Asocia al IRQ del periférico un valor que guarda en un registro del PIC (llamado **vector**)
  - Avisa a la CPU a través de la línea de interrupción pendiente (INT)
  - La CPU lee del registro (como puerto de E/S o como dirección de memoria) el vector
  - La CPU escribe en el registro de control del PIC que ya leyó el vector
  - El PIC desactiva la línea de interrupción pendiente. borra el vector y vuelve a monitorizar…

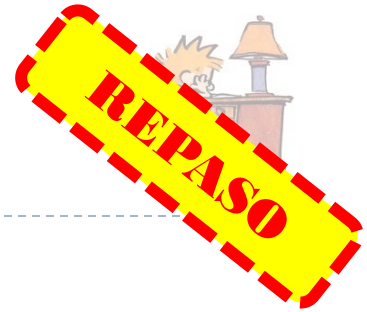
Interrupciones hardware
repaso (3/4)

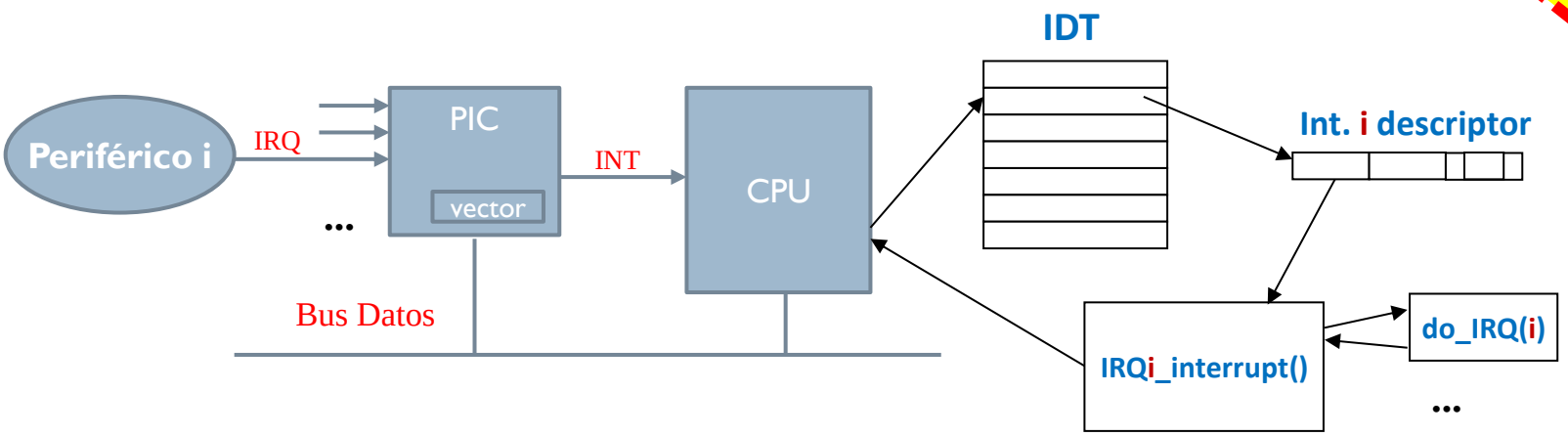




- El PIC puede permitir deshabilitar selectivamente las IRQ
  - El PIC deja de avisar a la CPU de la petición de una IRQ hasta que se habiliten: no se pierden.
  - Deshabilitar a nivel de CPU (mask/unmask) es diferente: ignora la INT
- El PIC puede permitir tener niveles de prioridad de interrupción
  - Se asocia a cada IRQ con una prioridad
  - Si hay varias IRQ, el PIC 'atiende' primero las de mayor prioridad (resto: deshabilitado temporalmente)
  - Si el PIC no tiene niveles de prioridad, se pueden simular por software en el sistema operativo

Interrupciones hardware
repaso (4/4)

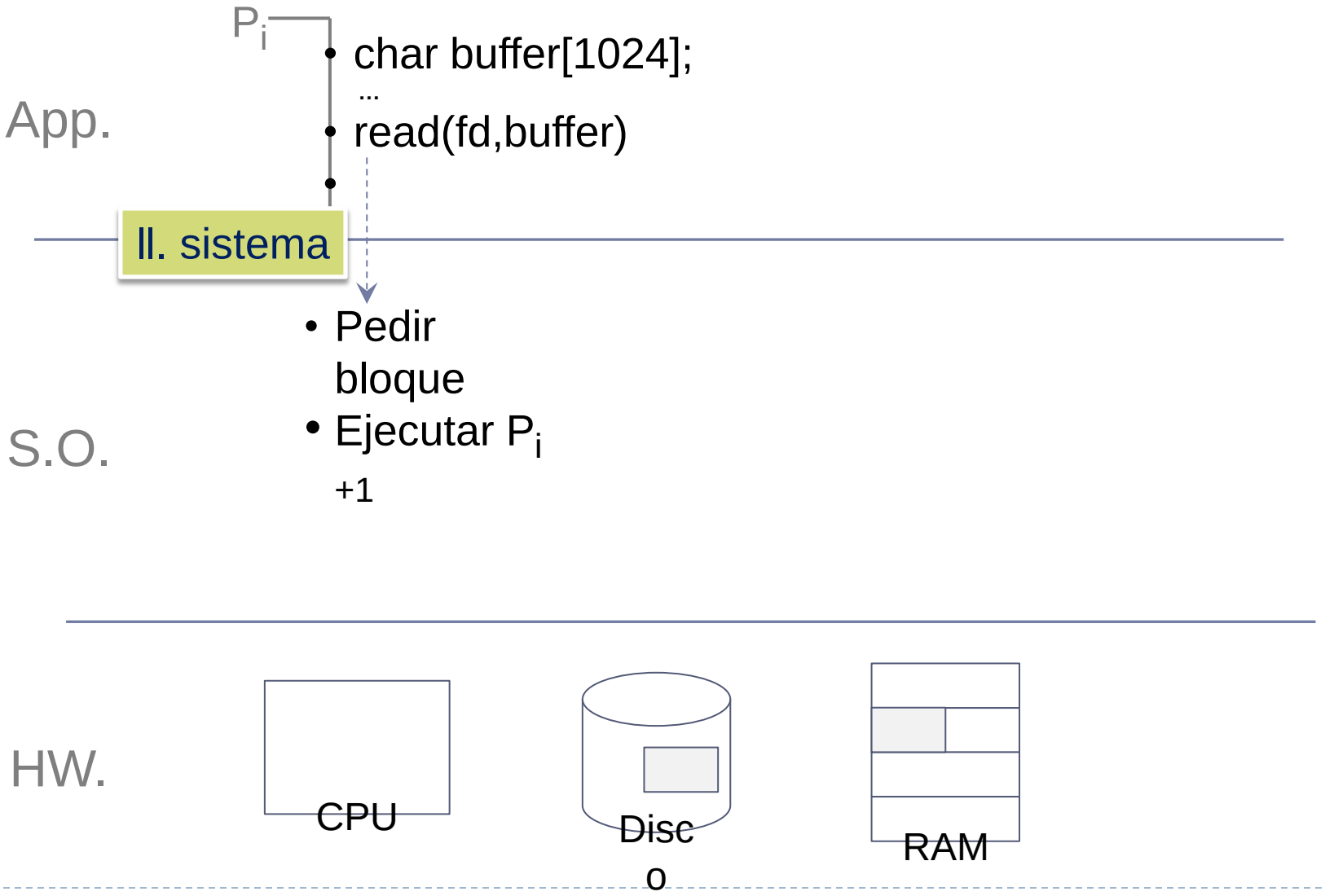




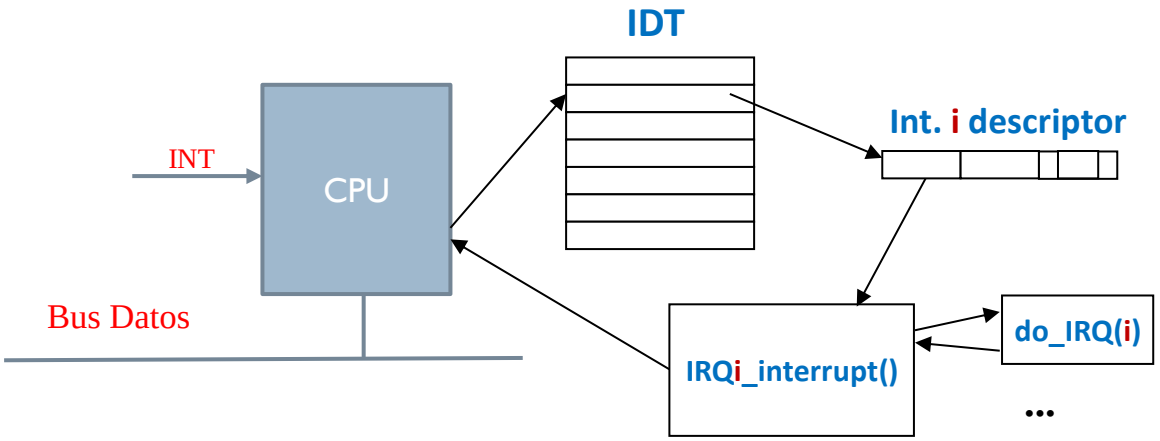
- La CPU detecta la petición de INT
- Acepta la interrupción: copia el vector por el bus de datos y ACK al PIC
- Busca en la *Interrupt Descriptor Table* (IDT) la rutina de tratamiento asociada
- Guarda el estado del procesador en pila, pasa a modo privilegiado y ejecuta la RTI
  - Puede que varias RTI (do_IRQ) compartan una misma interrupción (uso de enumeración)
  - Puede que varias interrupciones compartan una rutina genérica común a ellas
- Recupera el estado de pila y ejecuta RETI (paso a modo previo y vuelta a lo interrumpido)

Llamada al sistema

soporte hardware

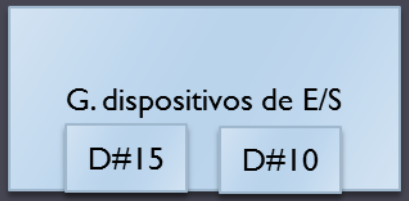


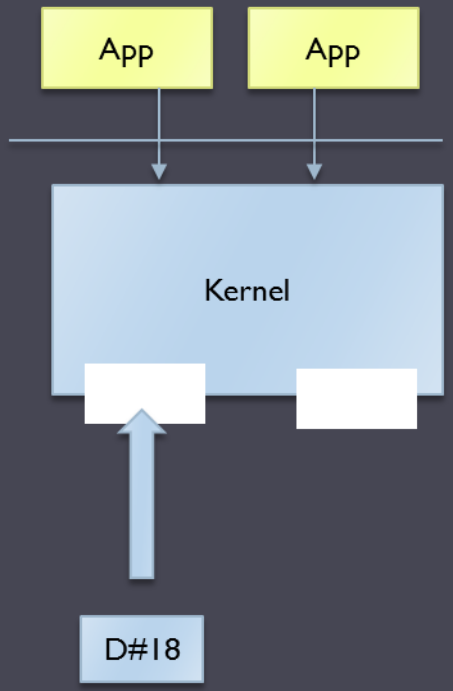
Llamada al sistema

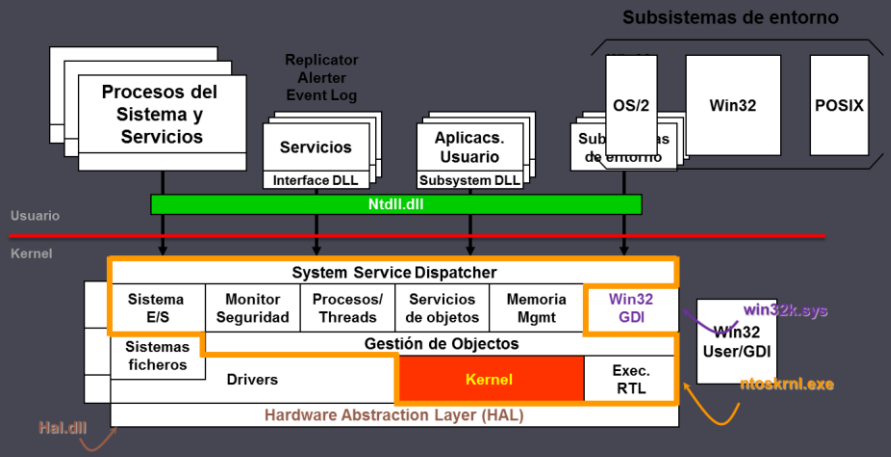


- Existe una instrucción en ensamblador que genera una interrupción por software
- La CPU detecta la petición de INT
- Busca en la *Interrupt Descriptor Table* (IDT) la rutina de tratamiento asociada
- Guarda el estado del procesador en pila, pasa a modo privilegiado y ejecuta la RTI
  - Puede que varias RTI (do_IRQ) compartan una misma interrupción (uso de enumeración)
  - Puede que varias interrupciones compartan una rutina genérica común a ellas
- Recupera el estado de pila y ejecuta RETI (paso a modo previo y vuelta a lo interrumpido)

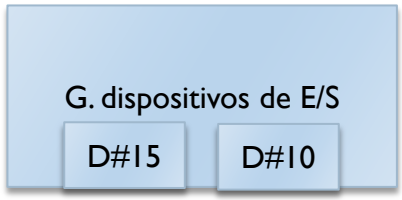
¿Cómo es por fuera?







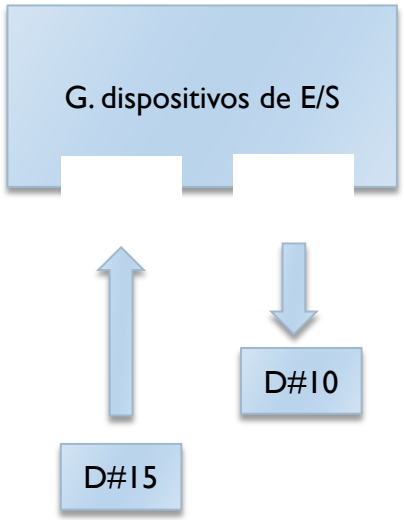
Ejecutables (1/1)



- Los primeros *kernels* tenían que:
  - Incluir código para todos los posibles dispositivos.
  - Cada cierto tiempo se recompilaba para añadir los nuevos dispositivos.
  - Se distribuía como un conjunto de **ejecutables**.

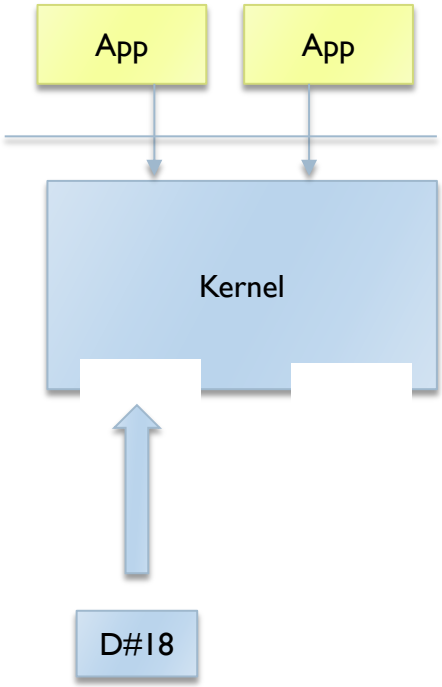
http://www.cc.gatech.edu/~pwh/

Módulos (1/2)



- 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/.so/.dmp).
  - El módulo puede descargarse cuando el dispositivo deje de usarse.

Módulos (2/2)

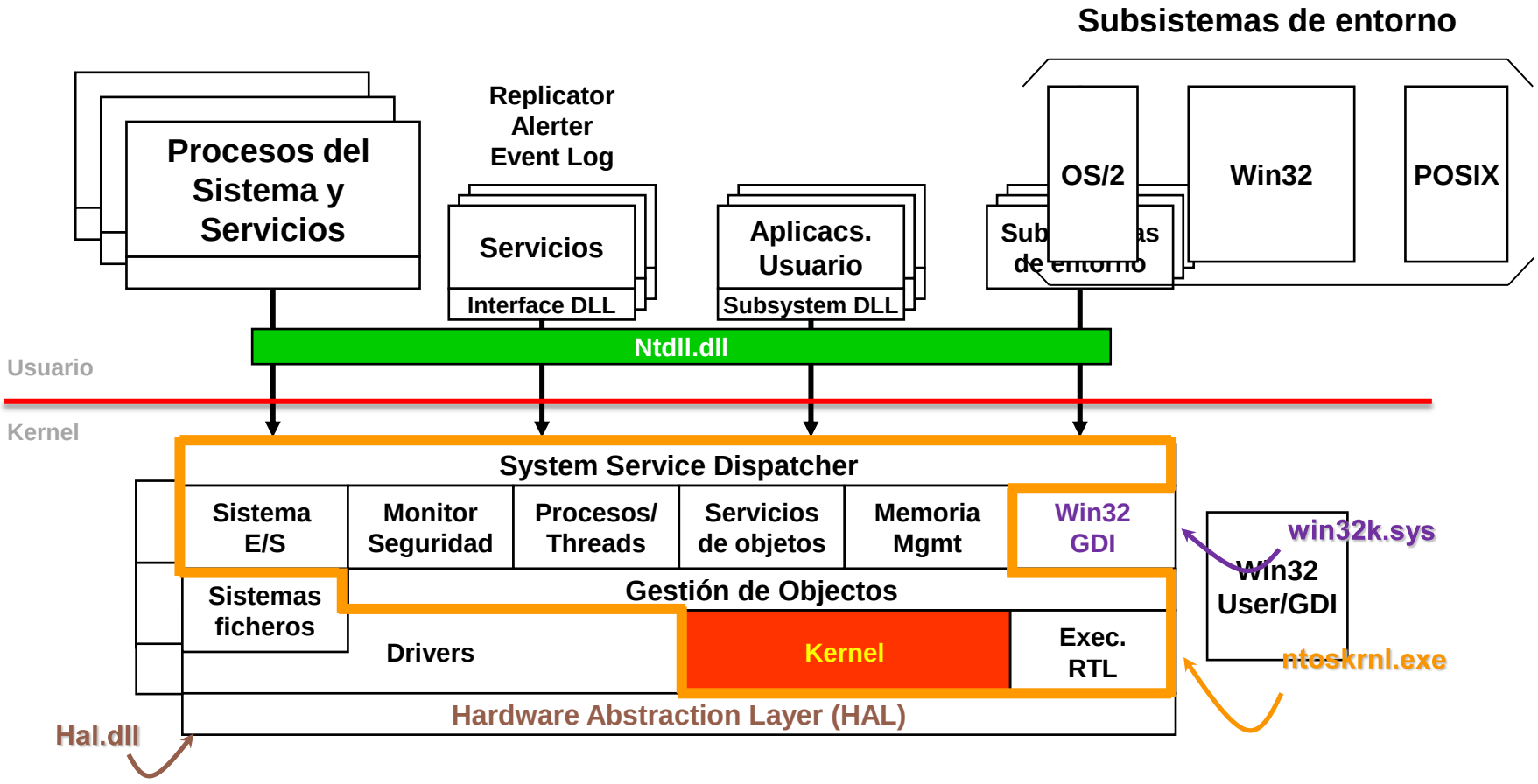


- La gran mayoría de sistemas operativos modernos tienen un kernel que permite el uso de módulos:
  - Linux, Solaris, BSD, Windows, etc.
- Los módulos se utilizan no solo para los drivers de los dispositivos, actualmente también se utilizan para añadir otro tipos de funcionalidad:
  - El kernel de Linux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

http://www.cc.gatech.edu/~pwh/

Ejemplo de Módulos

Arquitectura de la Familia Windows 2000



Lección 1
Introducción

Universidad Carlos III de Madrid
Grupo ARCOS

Diseño de Sistemas Operativos

Ingeniería Informática