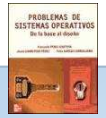
Grupo ARCOS
Universidad Carlos III de Madrid

Lección 2
Funcionamiento del sistema operativo

Diseño de Sistemas Operativos
Grado en Ingeniería Informática

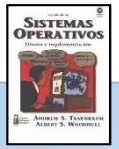
Lecturas recomendadas

Base

1. Carretero 2007:
   1. Cap.2

Recomendada



1. Tanenbaum 2006(en):
   1. Cap.1
2. Stallings 2005:
   1. Parte uno (transfondo)
3. Silberschatz 2006:
   1. Cap.2

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

- Introducción
- Funcionamiento del sistema operativo
  - Arranque del sistema
  - Características y tratamiento de los eventos
  - Procesos de núcleo
- Otros aspectos
  - Concurrencia en los eventos
  - Añadir nuevas funcionalidades al sistema

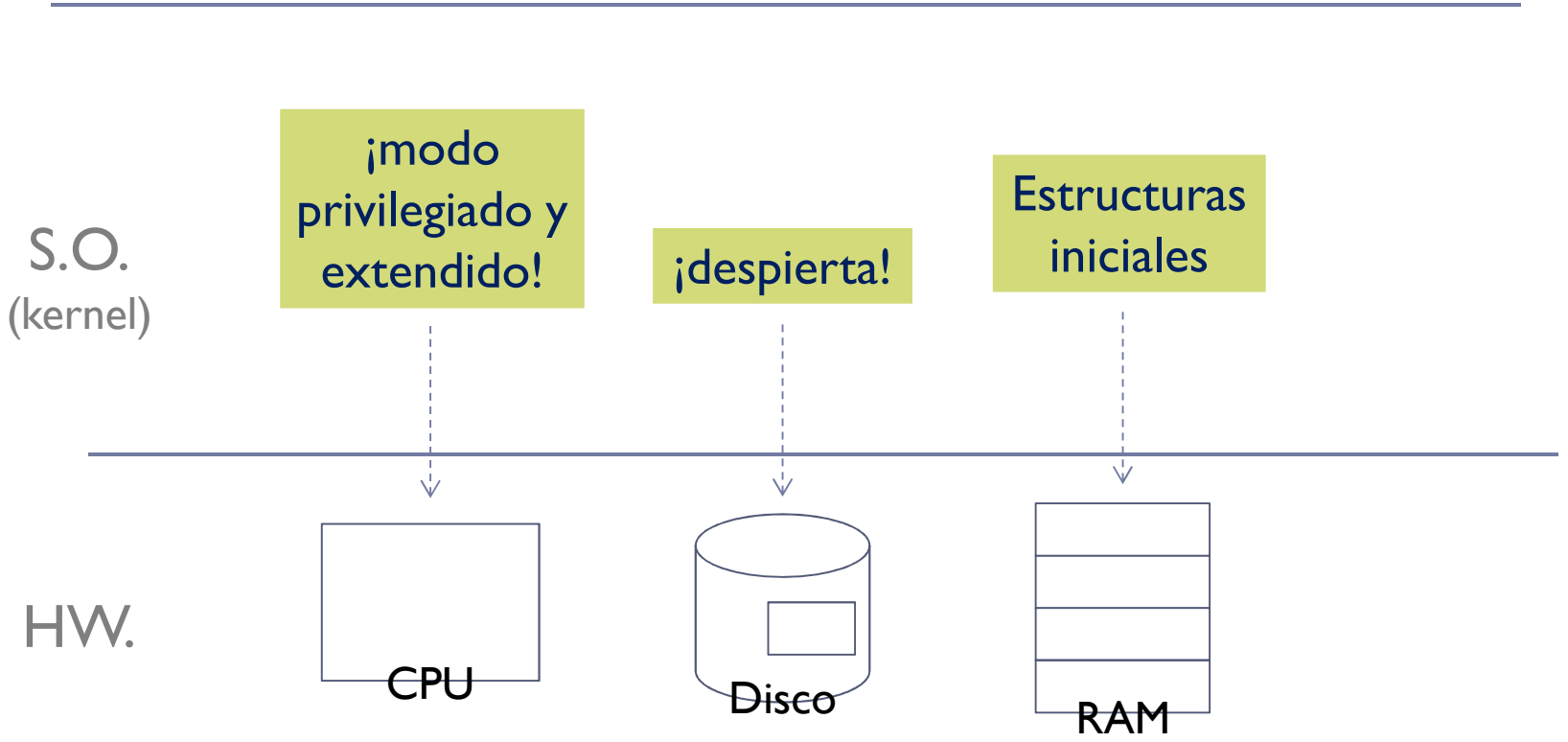
Contenidos

- **Introducción**
- Funcionamiento del sistema operativo
  - Arranque del sistema
  - Características y tratamiento de los eventos
  - Procesos de núcleo
- Otros aspectos
  - Concurrencia en los eventos
  - Añadir nuevas funcionalidades al sistema

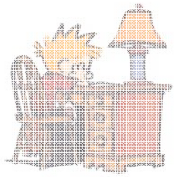
Contextos donde está presente el S.O. (1/3)

- Arranque del sistema
  - Realiza labores de iniciar el hardware y los procesos de núcleo, sistema y usuarios en el orden apropiados.
  - Ejecuta como programa ejecutable.

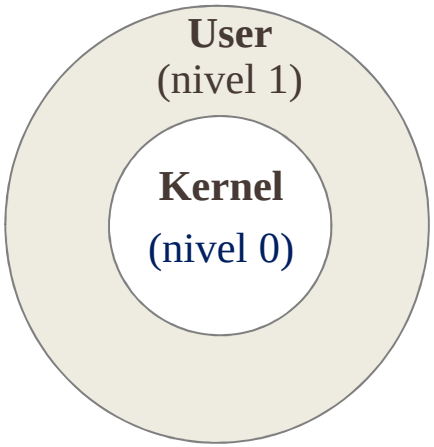
Ejemplo simplificado



Modo kernel y usuario
repaso



- El sistema operativo precisa de un **mínimo** de dos modos de ejecución:

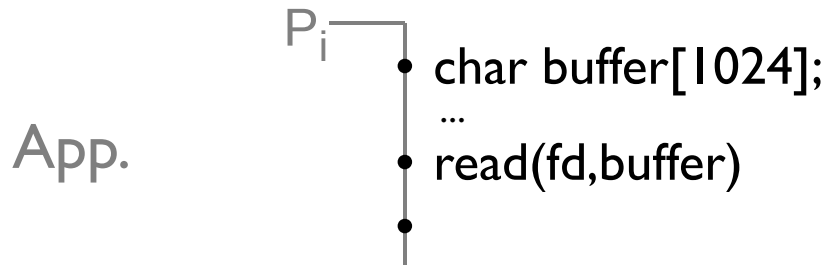


- Modo privilegiado (**kernel mode**)
  - Accede a todo el espacio de memoria
  - Uso de todo tipo de instrucciones de CPU
- Modo ordinario (**user mode**)
  - Accede solo al espacio de memoria del proceso asociado
  - No puede acceder a ciertos registros de la CPU o usar ciertas instrucciones

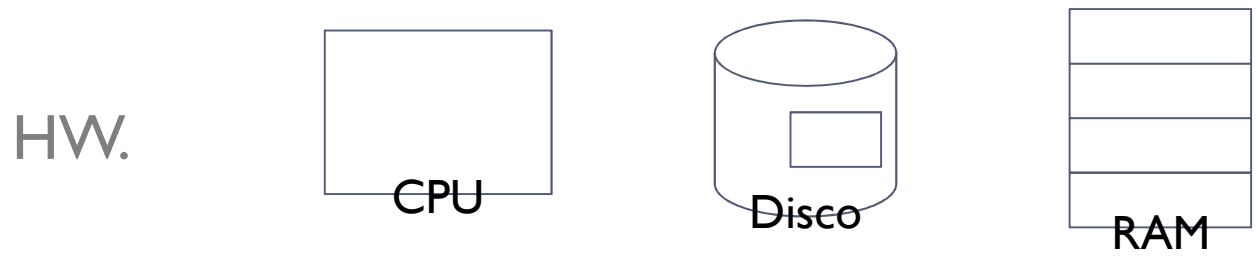
Contextos donde está presente el S.O. (2/3)

- Tratamiento de eventos
  - Finalizado el arranque, el sistema operativo es una entidad pasiva.
    - Los procesos y el hardware son entidades activas (usan el kernel)
    - Excepto al inicio, siempre hay un proceso ejecutando (*idle*)
  - Acceso a los servicios del S.O.
    - Interrupciones hardware
    - Interrupciones software
    - Excepciones
    - Llamadas al sistema
  - Como biblioteca.

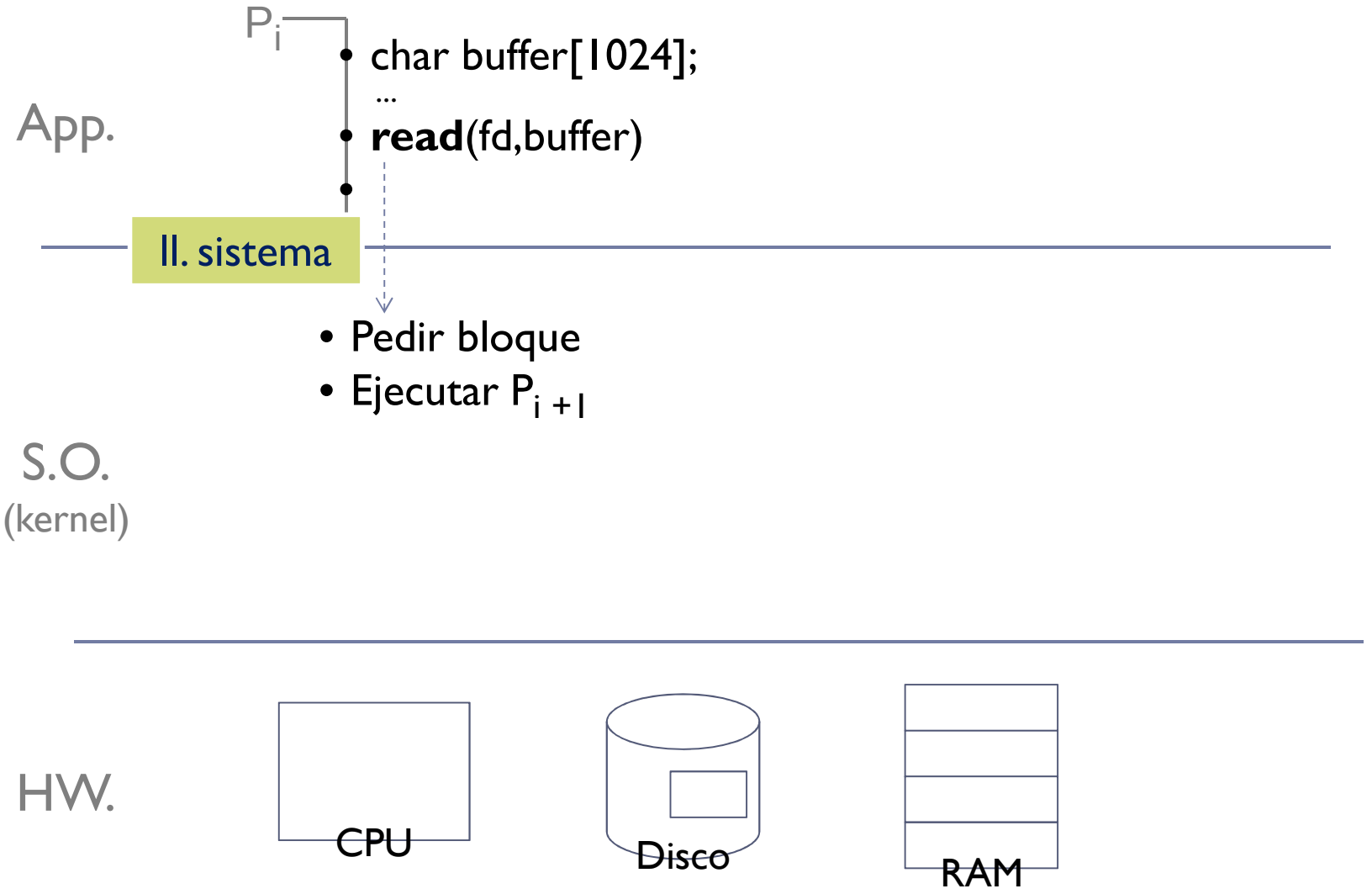
Ejemplo simplificado



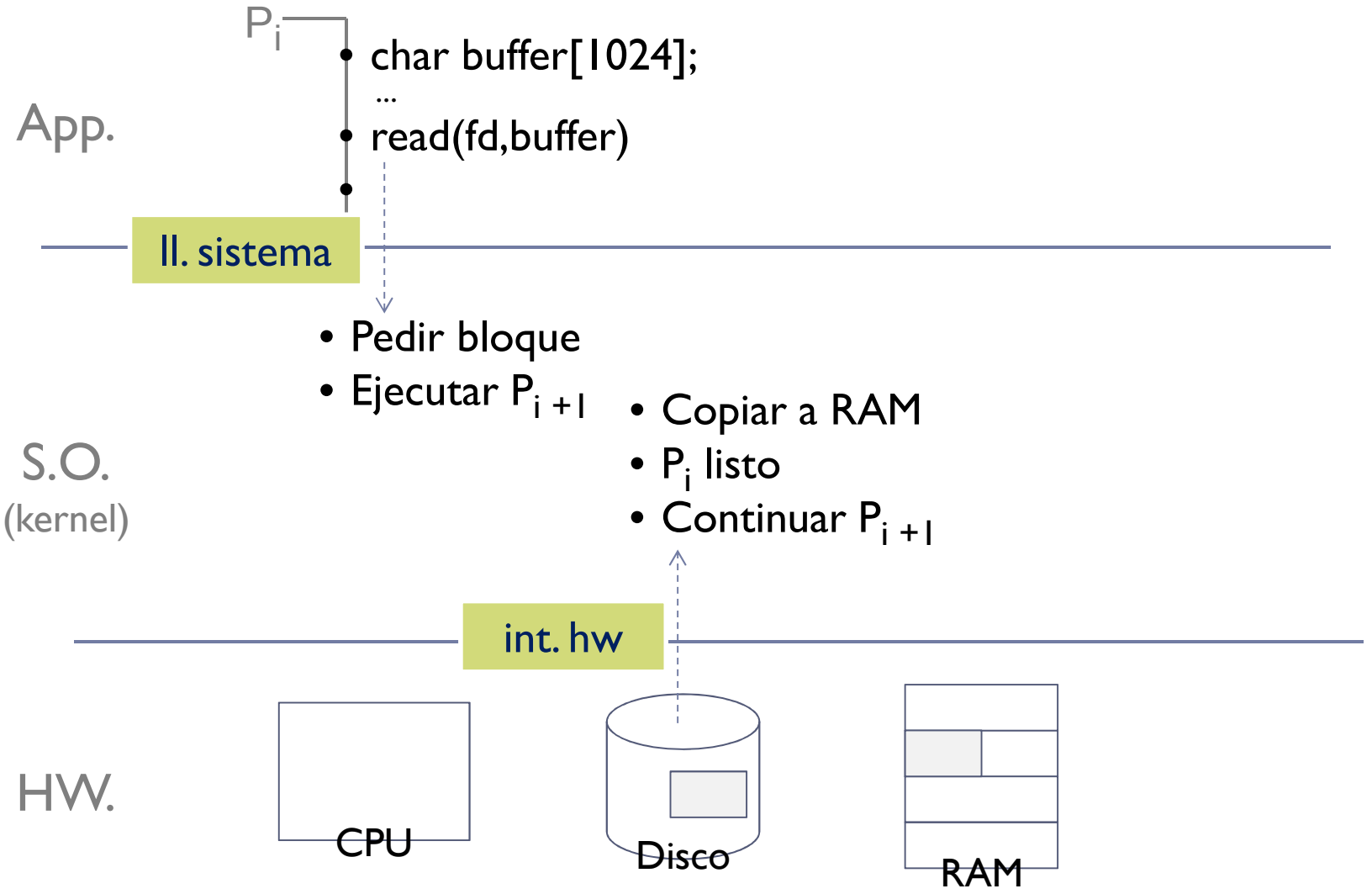
S.O.
(kernel)



Ejemplo simplificado



Ejemplo simplificado

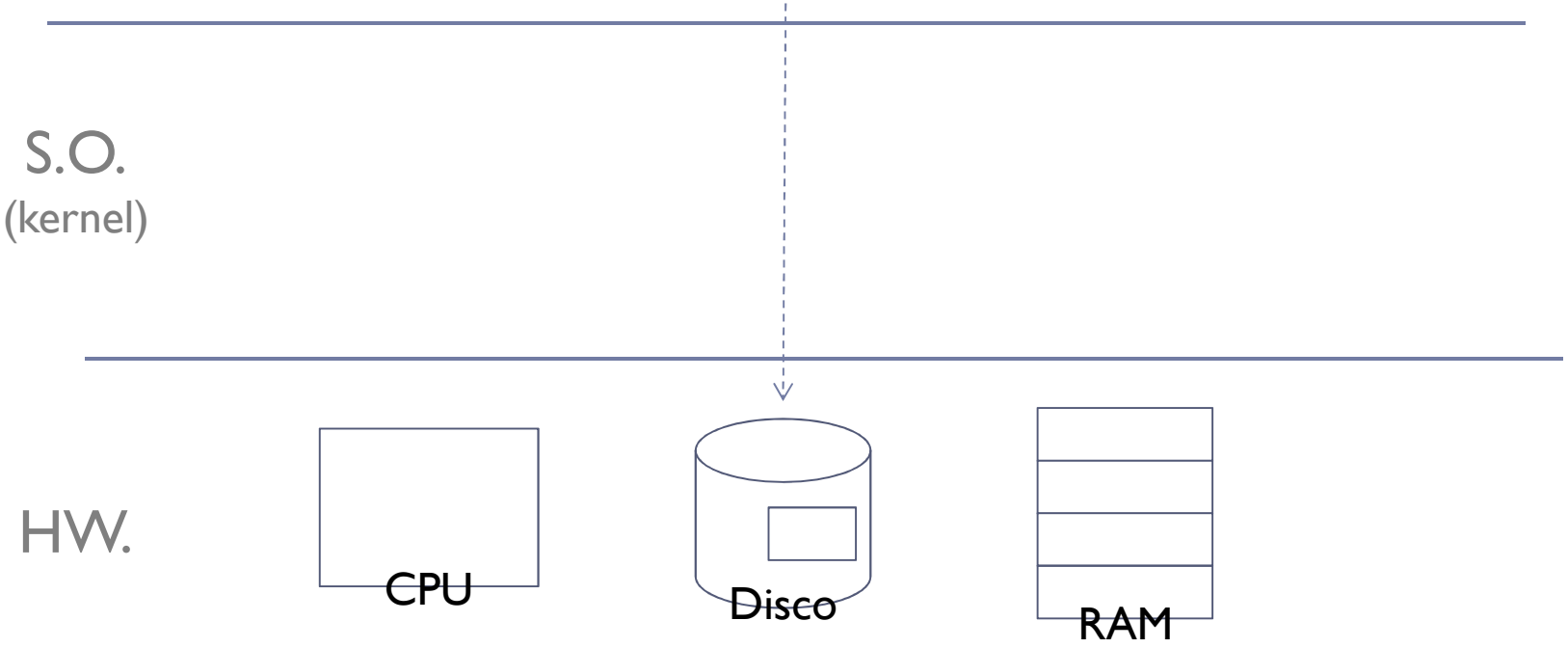


Contextos donde está presente el S.O. (3/3)

- Procesos de núcleo
  - Realiza labores del sistema operativo que se hacen mejor en el contexto de un proceso independiente
  - Como procesos prioritarios, para tareas especiales.

Ejemplo simplificado

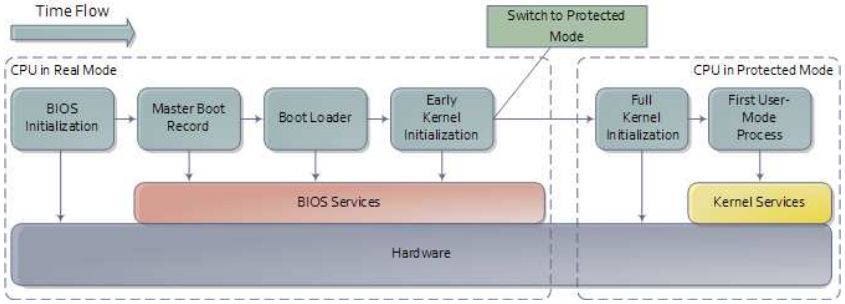
```
while (true) {
    • sleep(1);
    • Si (idle > 20m)
        dormir disco
}
```



Contextos donde está presente el S.O.
resumen

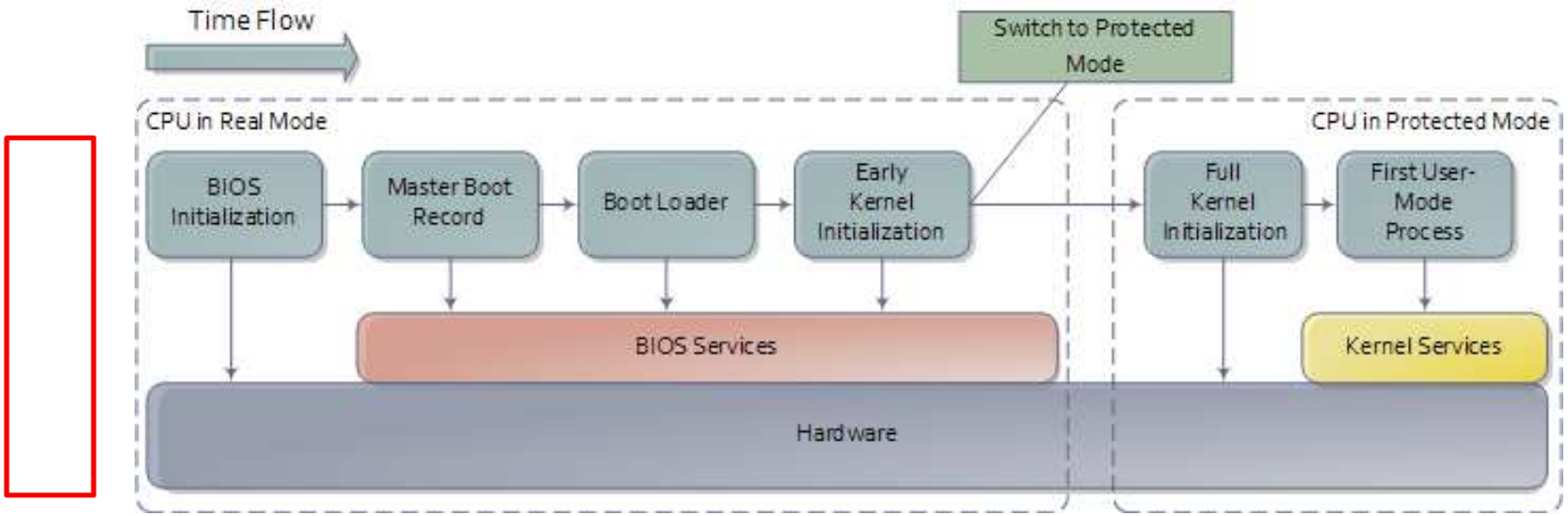
- Arranque del sistema
  - Realiza labores de iniciar el hardware y los procesos de núcleo, sistema y usuarios en el orden apropiados.
  - Ejecuta como programa ejecutable.
- Tratamiento de eventos
  - Finalizado el arranque, el sistema operativo es una entidad pasiva.
    - Los procesos y el hardware son entidades activas (usan el kernel)
    - Excepto al inicio, siempre hay un proceso ejecutando (idle)
  - Acceso a los servicios del S.O.
    - Int. hardware, Int. software, Excepciones, Llamadas al sistema
  - Como biblioteca.
- Procesos de núcleo
  - Realiza labores del sistema operativo que se hacen mejor en el contexto de un proceso independiente
  - Como procesos prioritarios, para tareas especiales.

Contenidos



- Introducción
- **Funcionamiento del sistema operativo**
  - **Arranque del sistema**
  - Características y tratamiento de los eventos
  - Procesos de núcleo
- Otros aspectos
  - Concurrencia en los eventos
  - Añadir nuevas funcionalidades al sistema

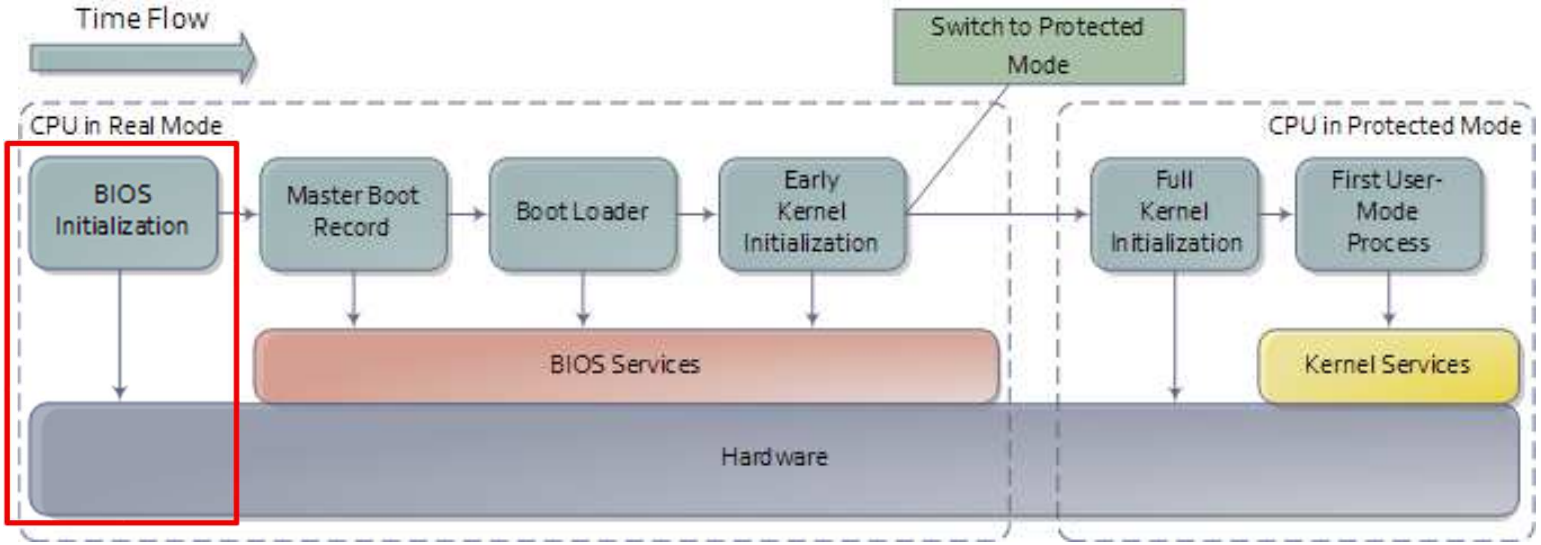
Proceso de arranque PC

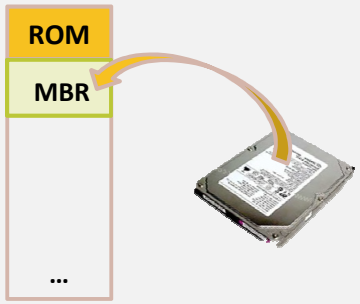


- El *Reset* carga en registros de CPU los valores iniciales
  - PC ← Dirección de arranque del cargador de la ROM (FFFF:0000)

http://duartes.org/gustavo/blog/post/how-computers-boot-up

Proceso de arranque PC

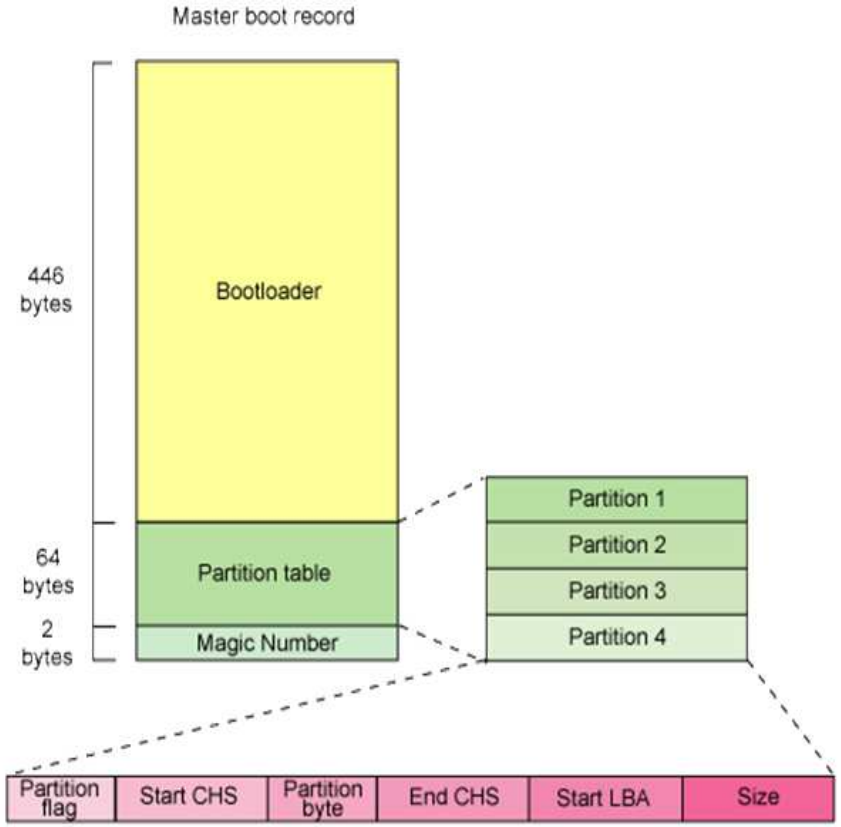


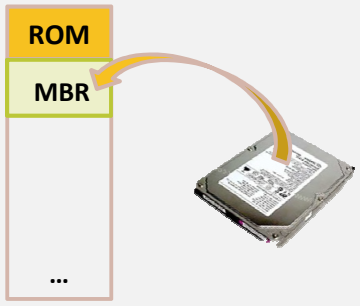


- Se ejecuta el cargador de la ROM
  - *Power-On Self Test* (POST)
  - Carga en memoria (0000:7C00) el *Master Boot Record*

http://duartes.org/gustavo/blog/post/how-computers-boot-up

Proceso de arranque PC

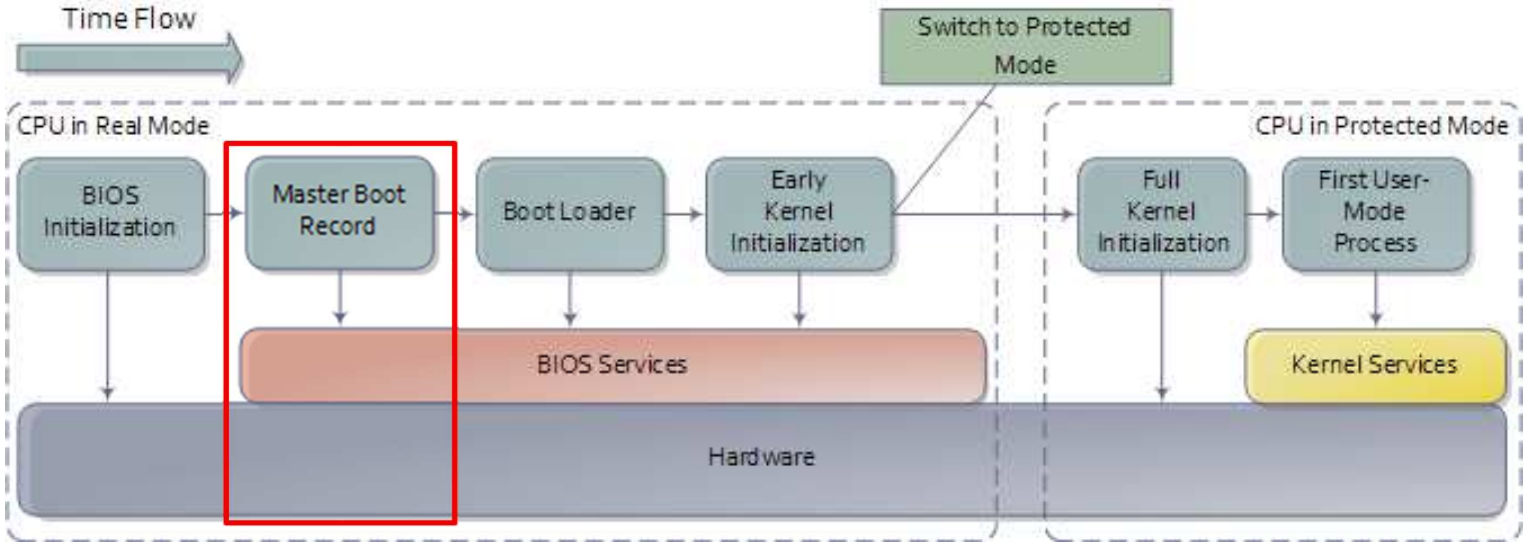


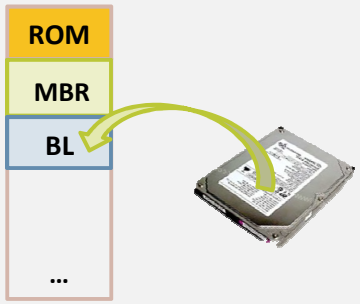


- Se ejecuta el cargador de la ROM
  - *Power-On Self Test* (POST)
  - Carga en memoria (0000:7C00) el ***Master Boot Record***

http://www.ibm.com/developerworks/linux/library/l-linuxboot/

Proceso de arranque PC

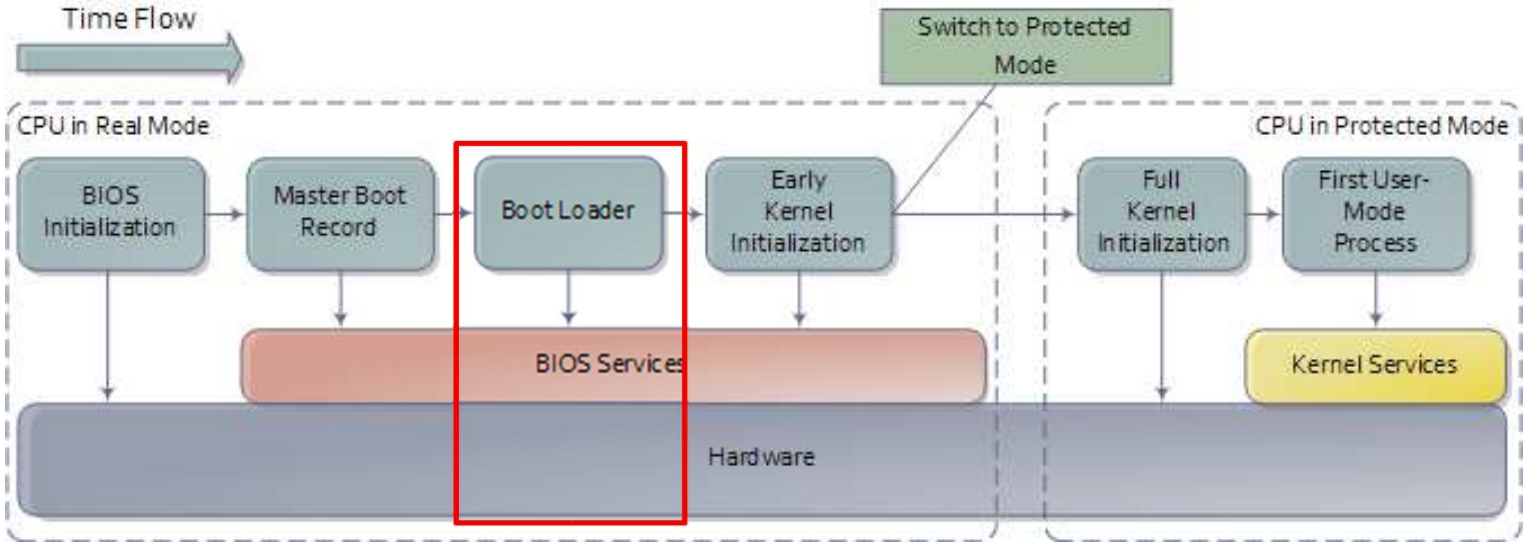


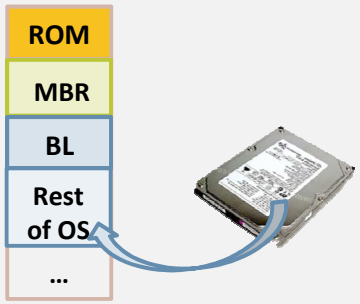


- Se ejecuta el *Master Boot Record*
  - (Es la primera parte del cargador del S.O.)
  - Busca una partición activa en la tabla de particiones
  - Carga el *Boot Record* en memoria desde esta partición

http://duartes.org/gustavo/blog/post/how-computers-boot-up

Proceso de arranque PC

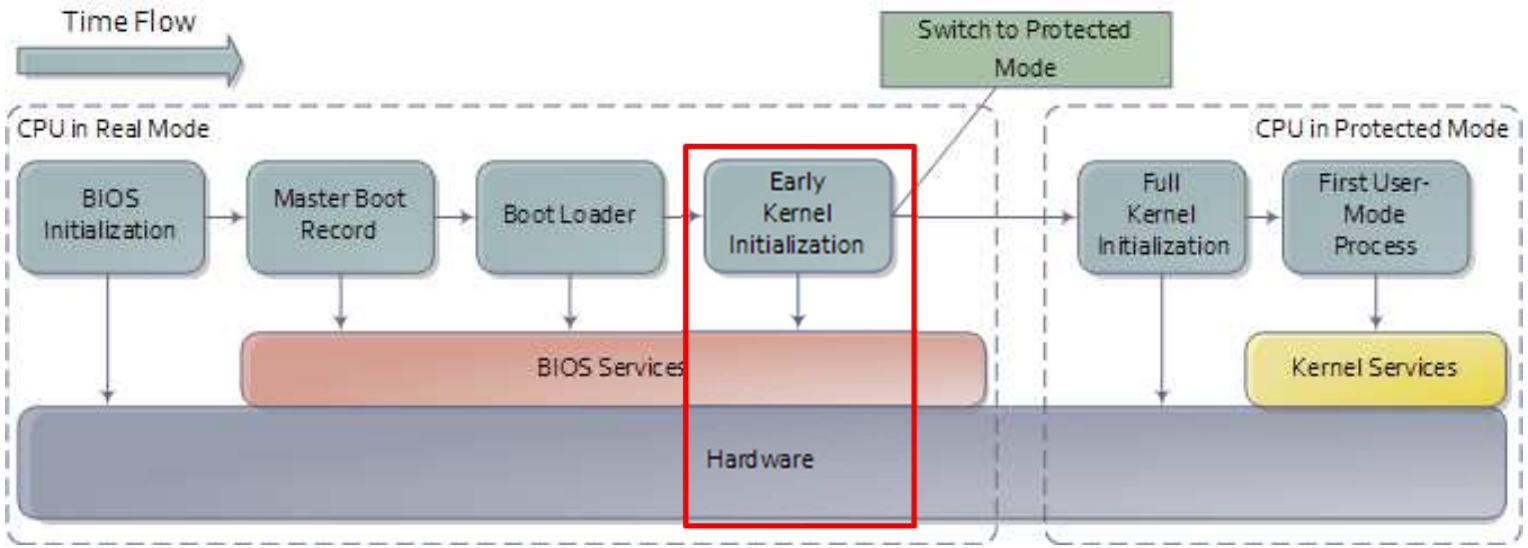




- Se ejecuta el *Boot Loader*
  - (Es la segunda parte del cargador del S.O.)
  - Podría presentar una lista de opciones de arranque…
  - El *boot loader* lleva a memoria la parte residente del sistema operativo (núcleo y módulos)

http://duartes.org/gustavo/blog/post/how-computers-boot-up

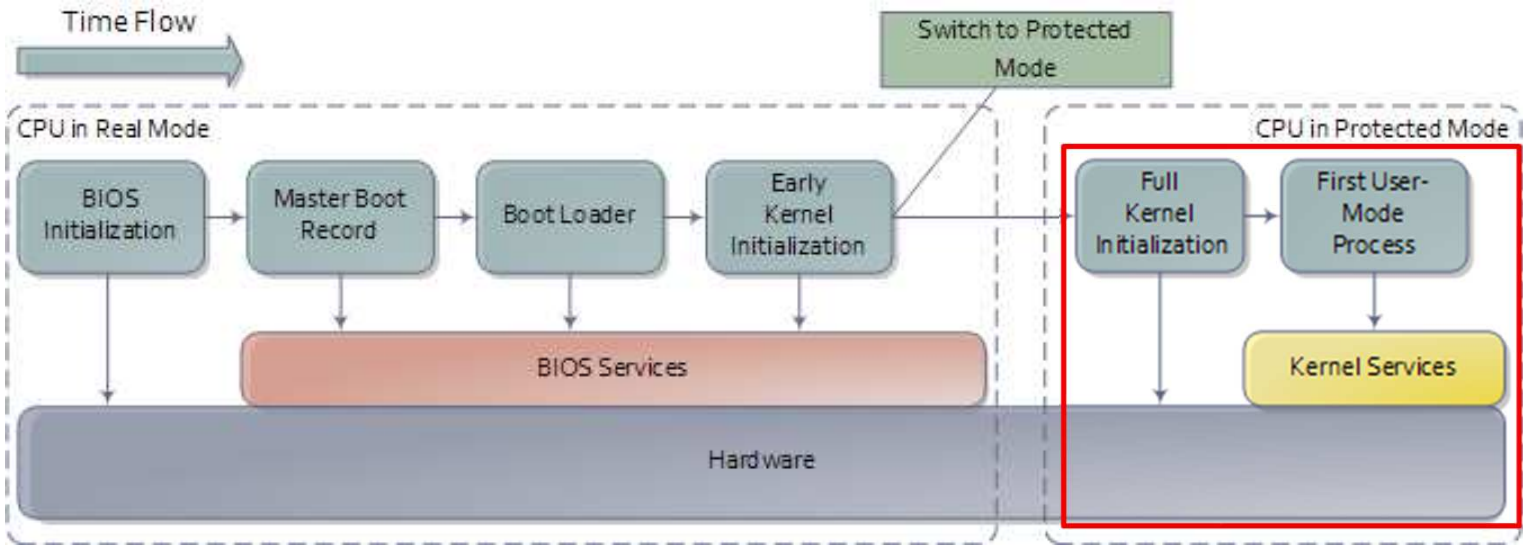
Proceso de arranque PC



- 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…

http://duartes.org/gustavo/blog/post/how-computers-boot-up

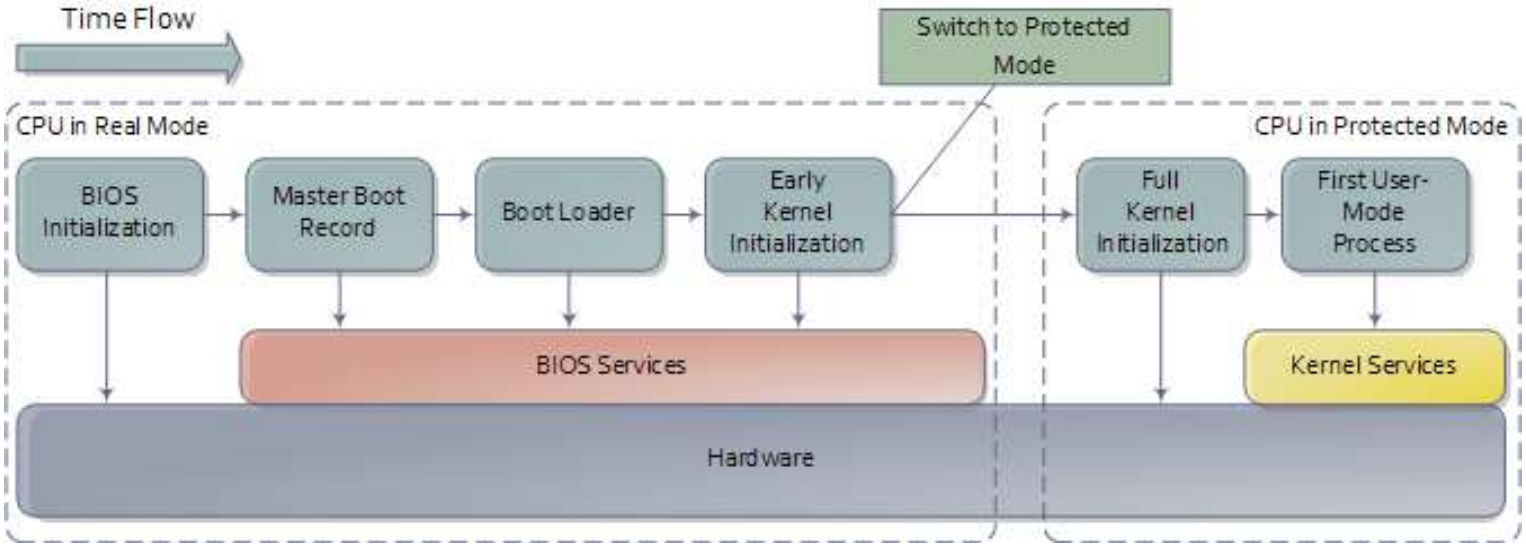
Proceso de arranque PC



- 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*)

http://duartes.org/gustavo/blog/post/how-computers-boot-up

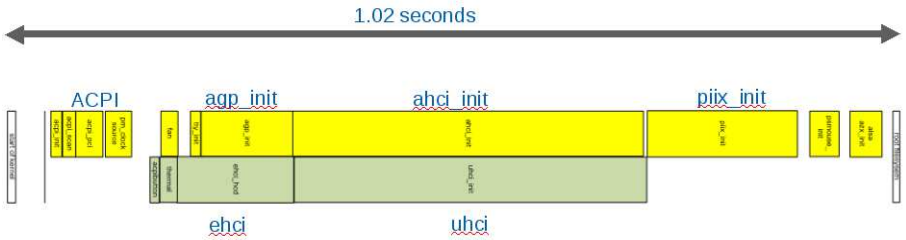
Proceso de arranque PC
resumen



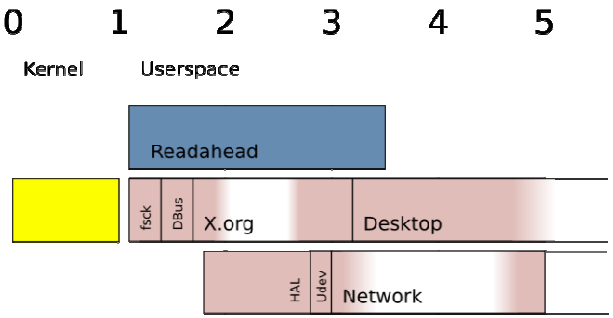
http://duartes.org/gustavo/blog/post/how-computers-boot-up

Acelerando el arranque en Linux

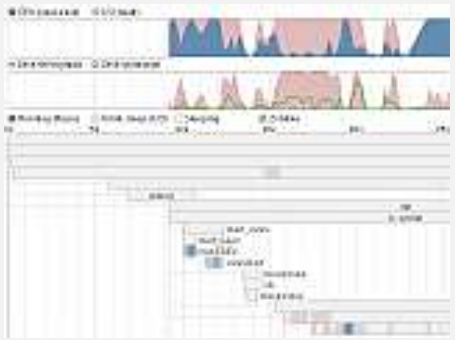
- Inicialización asíncrona del hardware



- Inicialización asíncrona de servicios

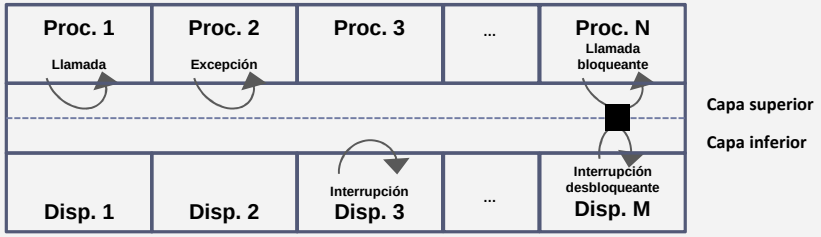


Bootchart



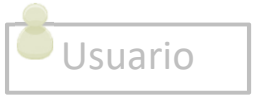
http://www.digitaltrendshttp://lwn.net/Articles/299483/

Contenidos



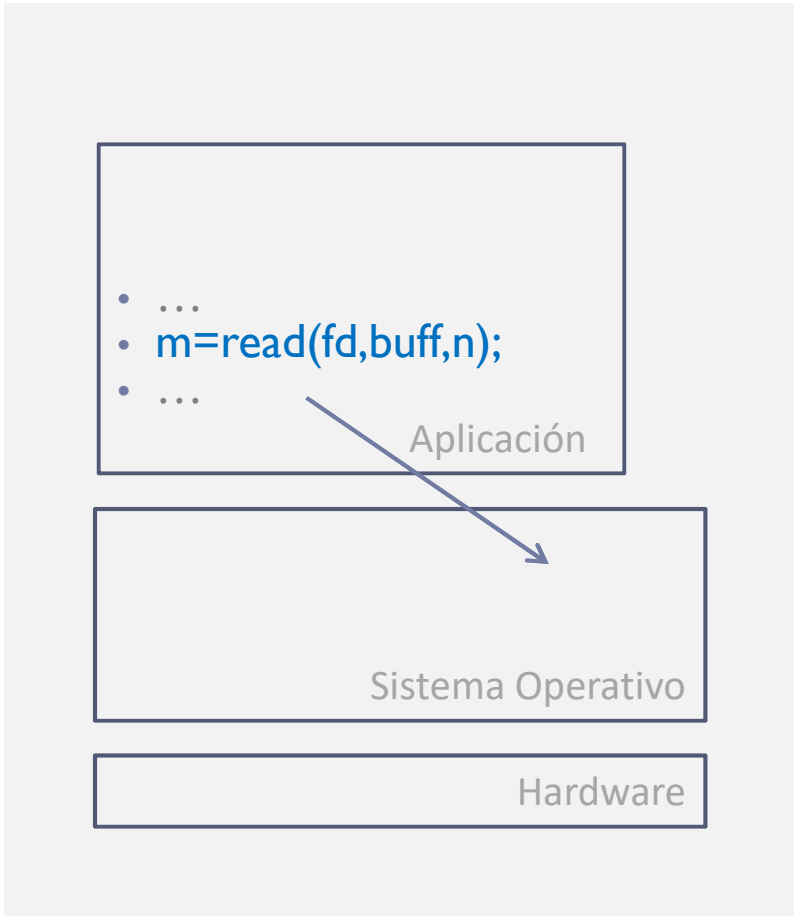
- Introducción
- **Funcionamiento del sistema operativo**
  - Arranque del sistema
  - Características y tratamiento de los **eventos**
  - Procesos de núcleo
- Otros aspectos
  - Concurrencia en los eventos
  - Añadir nuevas funcionalidades al sistema

Tipos de eventos



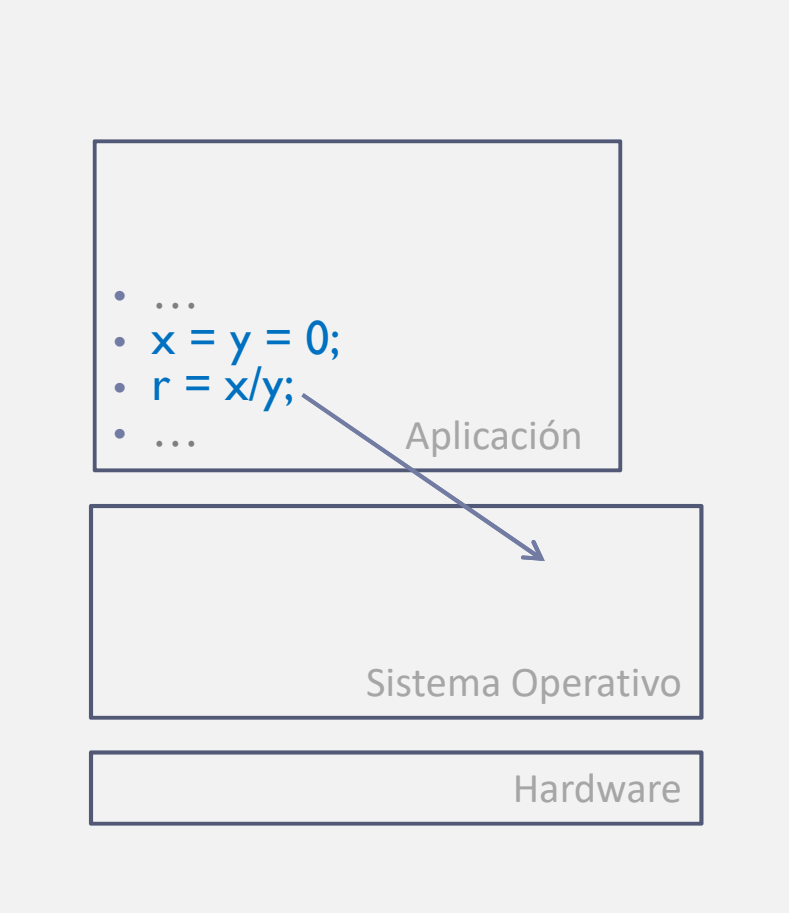
- Llamadas al sistema
  - Evento de solicitud de servicio del sistema operativo
- Excepciones
  - Eventos de carácter excepcional al ejecutar una instrucción
- Interrupciones software
  - Evento diferido de parte del tratamiento de evento pendiente
- Interrupciones hardware
  - Eventos que vienen del hardware

Tipos de eventos
Llamadas al sistema



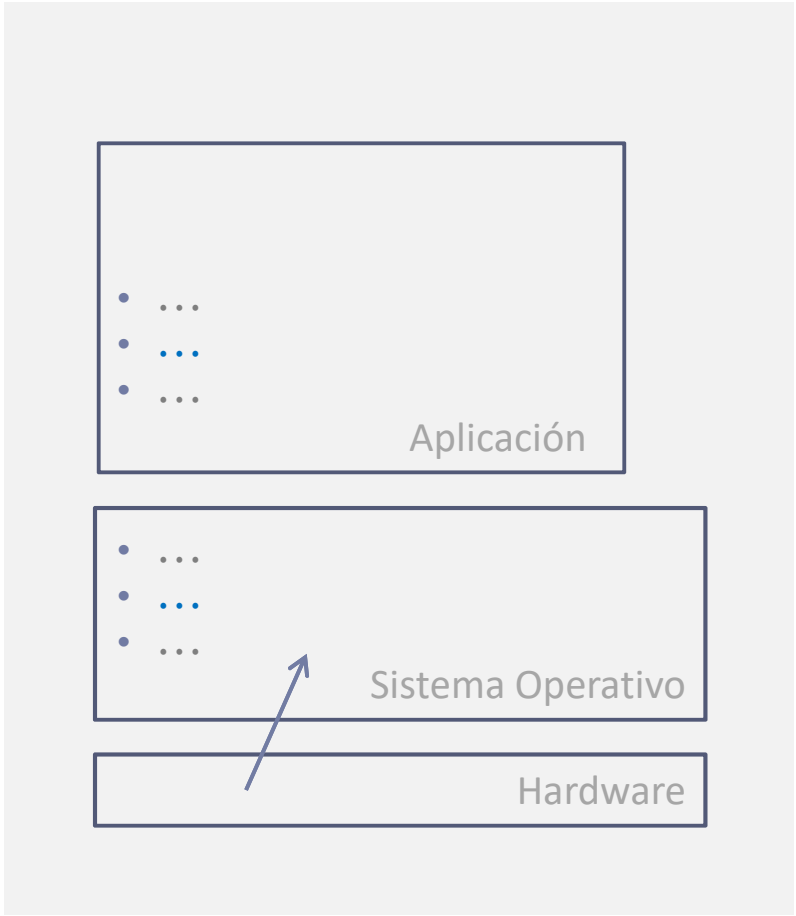
- Evento de solicitud de servicio del sistema operativo.
- Los programas de usuario acceden a los servicios del sistema operativo a través de llamadas al sistema.
- Son vistas por los usuarios programadores como llamadas a funciones.

Tipos de eventos
Excepciones



- Eventos de carácter excepcional al ejecutar una instrucción.
- Pueden ser problemas (división por cero, instrucción ilegal, violación de segmento, etc.) o avisos (fallo de página, etc.)
  - ~ Interrupción hardware generada por la propia CPU.
- Precisa de un conjunto de subrutinas asociadas a cada excepción que pueda darse.

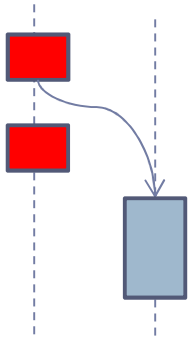
Tipos de eventos
Interrupciones hardware

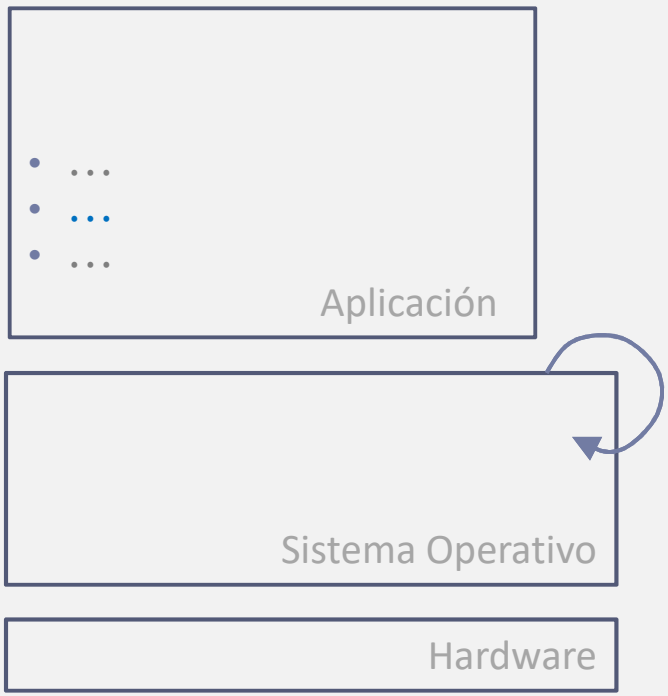


- Eventos que vienen del hardware.
- El sistema operativo tiene que atender a algo que necesita el hardware (llegada de datos, situación excepcional, etc.)
- Precisa de un conjunto de subrutinas asociadas a cada evento que el hardware pueda solicitar.

Tipos de eventos

Interrupciones software





- Evento para tratar en diferido la parte no crítica del tratamiento asociada a un evento.
- Se pospone parte del tratamiento de un evento:
  - Por esperar a circunstancias oportunas.
  - Se hayan tratado el resto de eventos más urgentes.

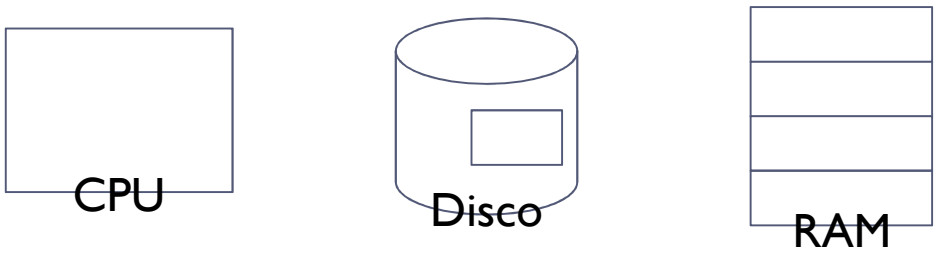
Ejemplo simplificado

App.

- char buffer[1024];
  ...
- read(fd,buffer)
- buffer[2048]='\0';

S.O.
(kernel)

HW.



Ejemplo simplificado

App.

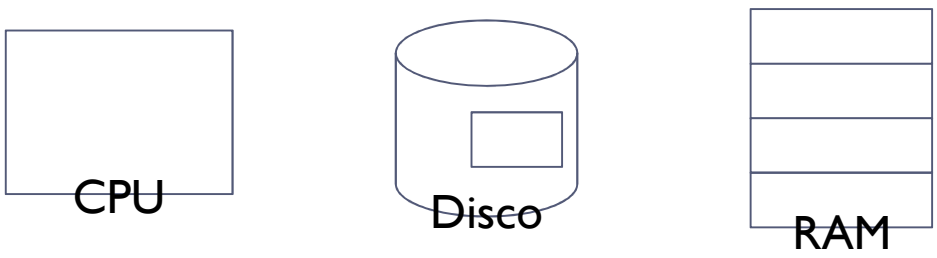
- char buffer[1024];
  ...
- **read**(fd,buffer)
- buffer[2048]='\0';

ll. sistema

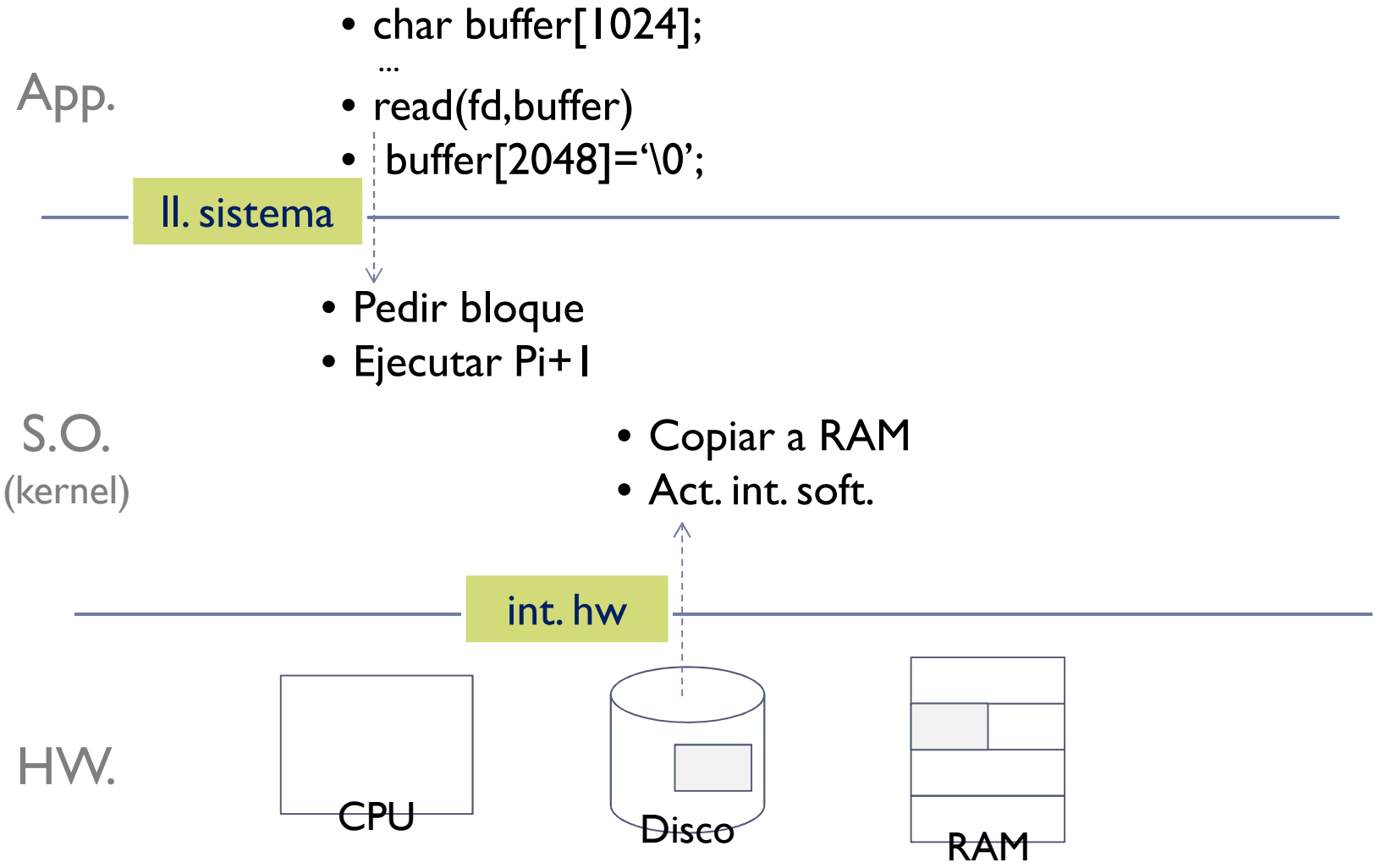
- Pedir bloque
- Ejecutar Pi+1

S.O.
(kernel)

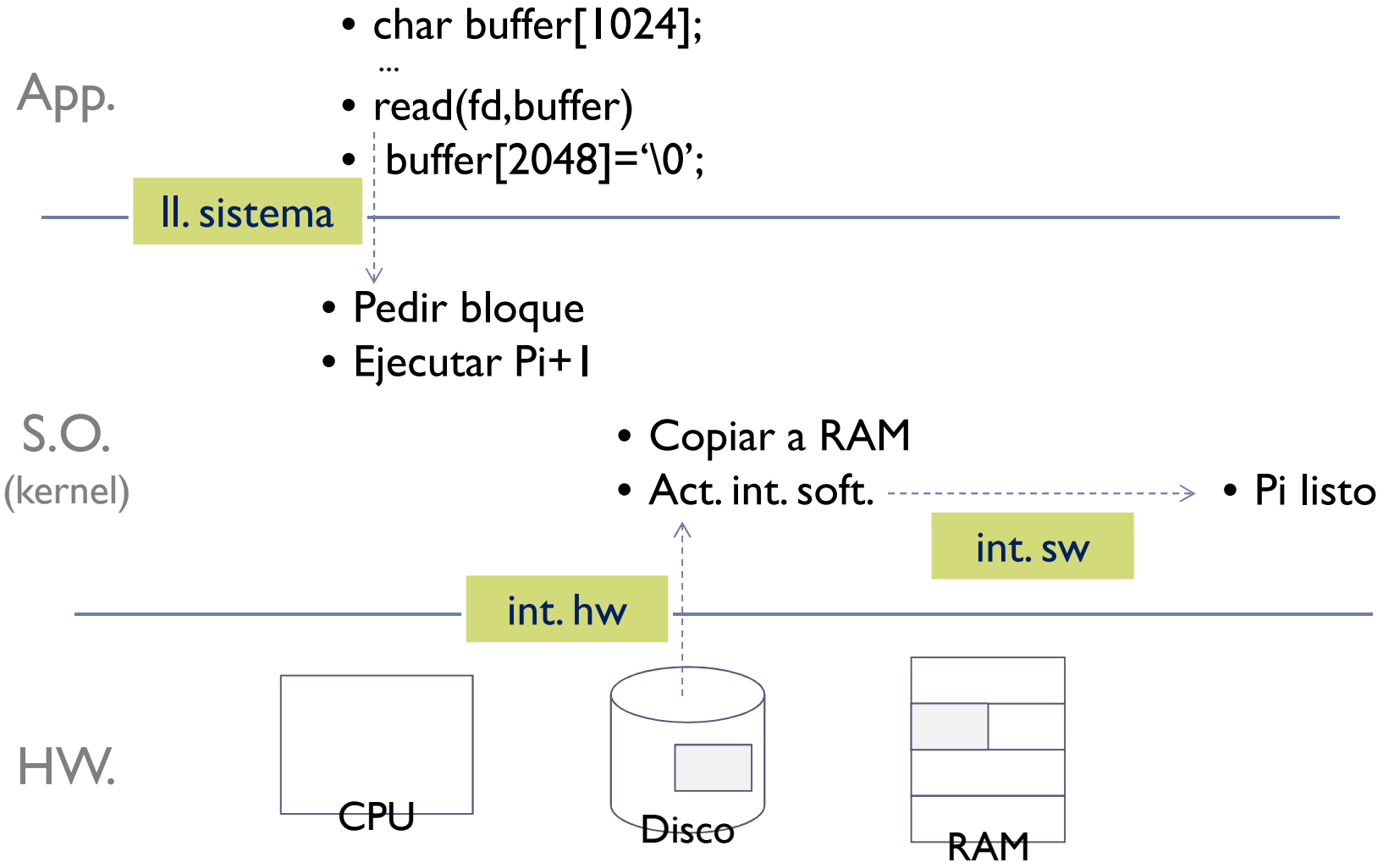
HW.



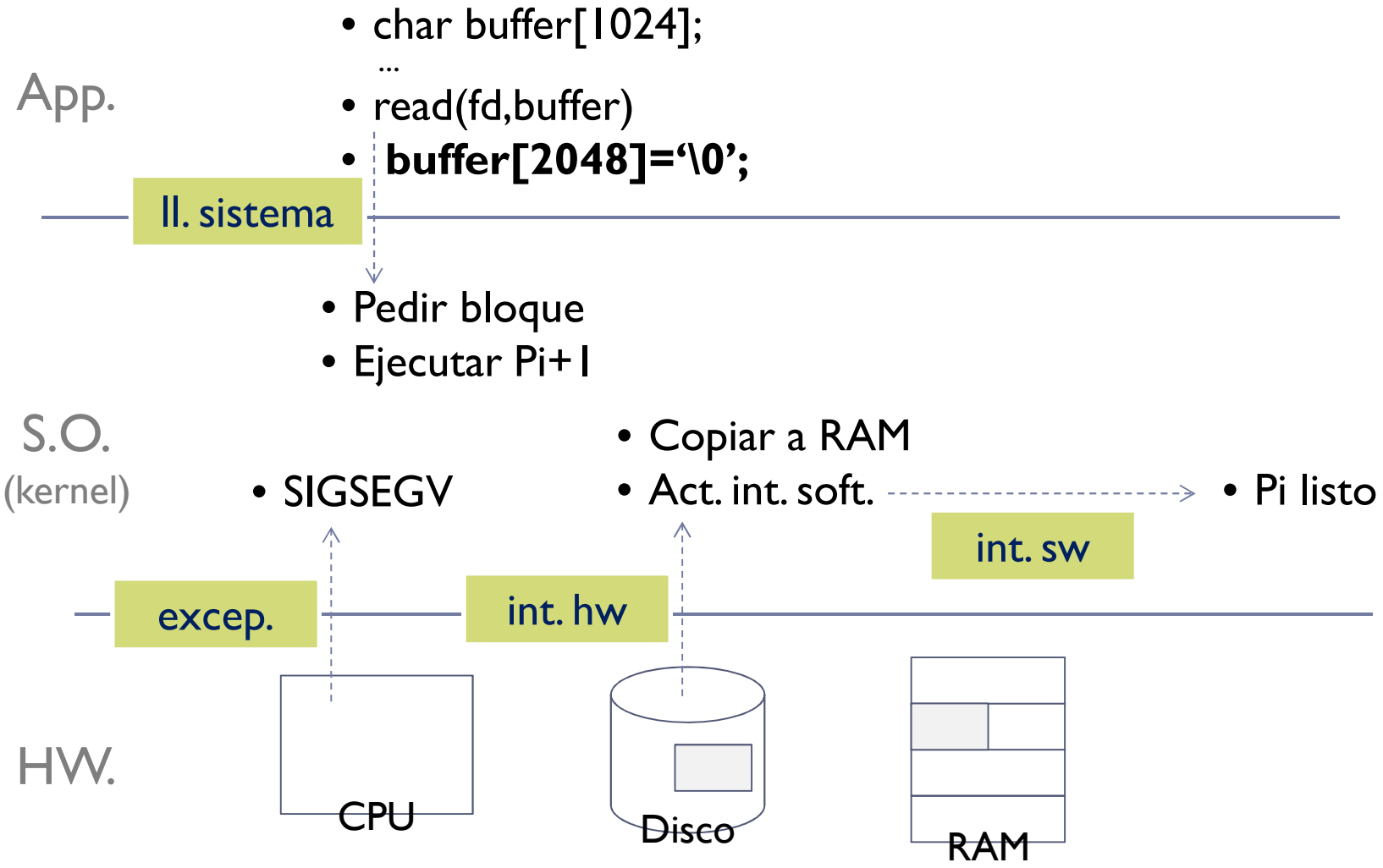
Ejemplo simplificado



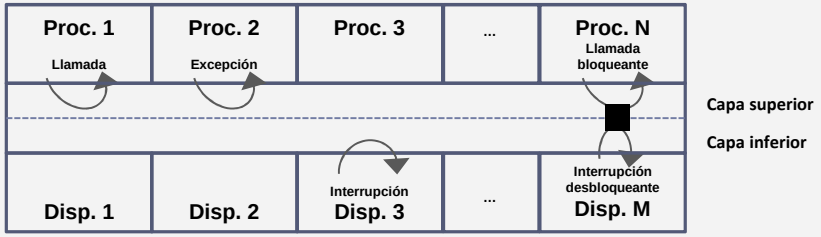
Ejemplo simplificado



Ejemplo simplificado



Contenidos



- Introducción
- **Funcionamiento del sistema operativo**
  - Arranque del sistema
  - **Características** y tratamiento de los eventos
  - Procesos de núcleo
- Otros aspectos
  - Concurrencia en los eventos
  - Añadir nuevas funcionalidades al sistema

Clasificación de los eventos

Interrupciones hardware

Llamada al sistema

Interrupciones software

Excepciones

	Síncronos	Asíncronos
Hardware		
Software		

Clasificación de los eventos

	Síncronos	Asíncronos
Hardware	Excepciones	Interrupciones hardware
Software	Llamada al sistema	Interrupciones software

- Generadas por software o hardware:
  - Generadas por hardware
    - La solicitud y el vector se obtiene del hardware implicado
  - Generadas por software
    - La solicitud y el vector componen una instrucción ensamblador

Clasificación de los eventos

	Síncronos	Asíncronos
Hardware	Excepciones	Interrupciones hardware
Software	Llamada al sistema	Interrupciones software

- Eventos síncronos y asíncronos:
  - Eventos síncronos
    - Su activación es previsible, refiriéndose al código del proceso actual
    - Ejecución en el contexto del proceso "solicitante"
  - Eventos asíncronos
    - Su activación es imprevisible y referida a cualquier (o ningún) proceso
    - Ejecución en el contexto de un proceso no relacionado con la interrupción

Características básicas…

Usuario Sistema Dispositivos S.O. C.P.U. Aplicaciones

	Modo de ejecución previo	Generadas por
Interrupción hardware		
Excepciones		
Llamadas al sistema		
Software		

Características básicas…

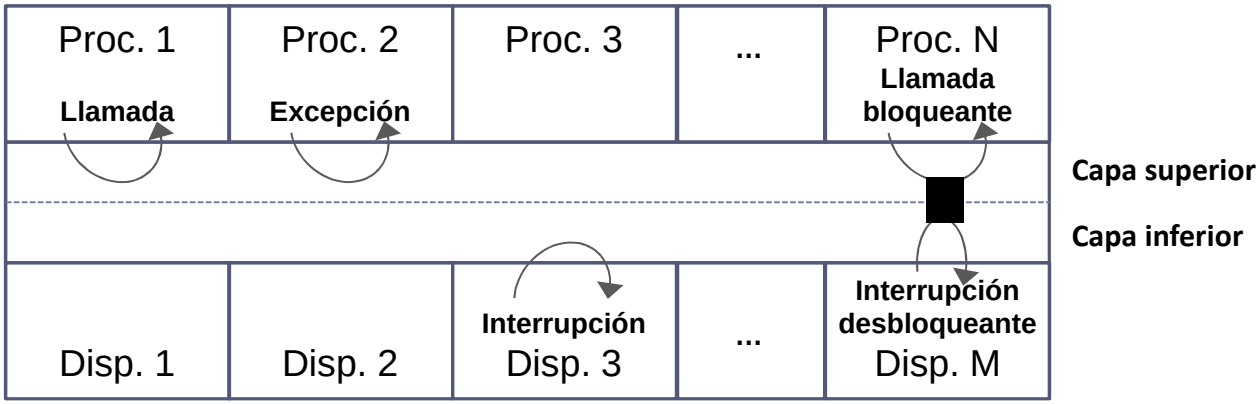
	Modo de ejecución previo	Generadas por
Interrupción hardware	• Puede ser **usuario o sistema** • **NO** influye en el tratamiento	• Dispositivos de E/S • Interrupción entre las CPU (IPI)

Excepciones	• Puede ser **usuario o sistema**
	  • **SI** influye en el tratamiento
		• La CPU (int. hw. de CPU)
		  • Normalmente errores de programación,
		NO siempre (fallo de página, depuración, etc.)

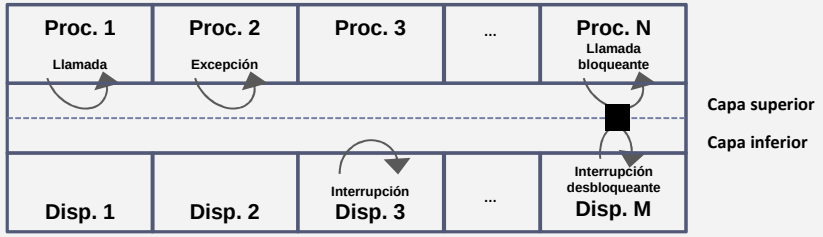
Llamadas al sistema	• Siempre **usuario**	• Las aplicaciones
Software	• Siempre **sistema**	• El tratamiento de cualquiera de los eventos anteriores: usado para la parte no crítica

Relación entre eventos

- Componentes que tratan eventos síncronos
  - Más relacionados con los procesos
- Componentes que tratan eventos asíncronos
  - Más relacionados con los dispositivos
- Existen tareas que involucran ambos tipos de eventos.
  - Ej.: acceso al disco (llamada lectura + interrupción del disco)



Contenidos



- Introducción
- **Funcionamiento del sistema operativo**
  - Arranque del sistema
  - Características y **tratamiento** de los eventos
  - Procesos de núcleo
- Otros aspectos
  - Concurrencia en los eventos
  - Añadir nuevas funcionalidades al sistema

Gestión de eventos

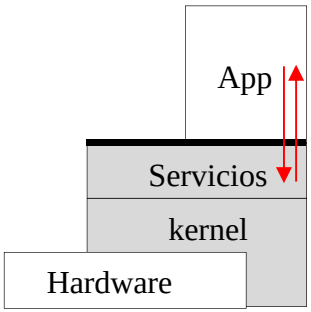
- El esquema del S.O. busca ser genérico e independiente de la arq. de hw.
  - Linux sin prioridad (SPARC si soporta) y Windows con prioridad (Intel no lo soporta)

Gestión de eventos

- El esquema del S.O. busca ser genérico e independiente de la arq. de hw.
  - Linux sin prioridad (SPARC si soporta) y Windows con prioridad (Intel no lo soporta)
- Todos los eventos se tratan de forma similar (~int. hw.)
  - Se ha ido introduciendo ya la gestión de eventos

Gestión de eventos

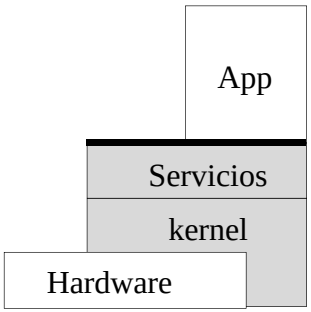
- El esquema del S.O. busca ser genérico e independiente de la arq. de hw.
  - Linux sin prioridad (SPARC si soporta) y Windows con prioridad (Intel no lo soporta)
- Todos los eventos se tratan de forma similar (~int. hw.)
  - Se ha ido introduciendo ya la gestión de eventos



- Se salva el estado en la pila de sistema
  - Típicamente los registros PC y estado
- La CPU pasa en modo privilegiado y salta a la rutina de tratamiento asociada
  - Salva registros extra si es necesario
  - La rutina de manejo del evento trata el evento
  - Restaura registros extra si es necesario
- La rutina de manejo del evento termina: RETI
  - Se restaura el estado salvado en pila y se vuelve al modo previo

Gestión de eventos

- Detalle 1 > No se tratan eventos durante el arranque
  - Modo sistema, deshabilitada las interrupciones y MMU inactiva
- Detalle 2 > Cuando ocurre un evento, entra el S.O para tratarlo:
  - Se cambia de modo (a modo privilegiado)
  - pero NO necesariamente hay cambio de contexto



  - El evento se trata en el contexto del proceso activo
  - El mapa de memoria activo del proceso en ejecución, incluso aún no teniendo relación con el evento.
  - El sistema debe usar dos pilas independientes:
    - Pila de usuario: para modo usuario
    - Pila de sistema: para modo sistema
- Detalle 3 > Puede 'saltar' un evento durante el tratamiento de otro
  - Si prioritario: se apila el actual y se trata el nuevo; si no, espera la finalización.

Gestión de eventos

- Interrupción hardware:
  - Tratamiento general
  - Ejemplo: W y L
- Excepción:
  - Tratamiento general
- Llamada al sistema:
  - Tratamiento general
  - Ejemplo: W y L
- Interrupción software:
  - Tratamiento general
  - Ejemplo: W y L

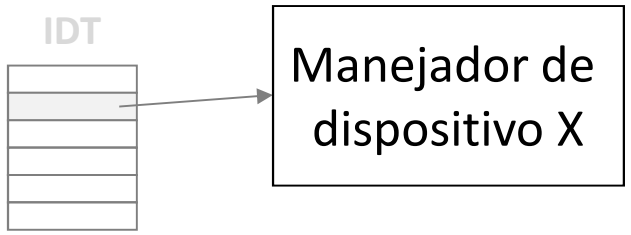
Interrupciones hardware
características

- Notifican eventos asíncronos que viene del hardware
- Modo de ejecución previo:
  - Puede ser usuario o sistema (NO influye en el tratamiento)
- Generadas por:
  - Dispositivos de E/S
  - Condiciones críticas en el sistema (ej.: corte energía)
  - Interrupción entre procesadores (IPI)

Interrupciones hardware
tratamiento (1/5)

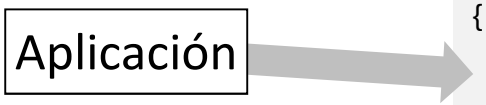
Modo Usuario

Modo Kernel




```
int main (int argc, char **argv)
{
        ...
        /* instalar los manejadores para los vectores de interrupción */
        instal_man_int(EXC_ARITMETICA,   excepcionAritmetica);
        instal_man_int(EXC_MEMORIA,      excepcionMemoria);
        instal_man_int(INT_RELOJ,        interrupcionReloj);
        instal_man_int(INT_DISPOSITIVOS, interrupcionDispositivos);
        instal_man_int(LLAM_SISTEMA,     tratarLlamadaSistema);
        instal_man_int(INT_SW,           interrupcionSoftware);
        ...
```

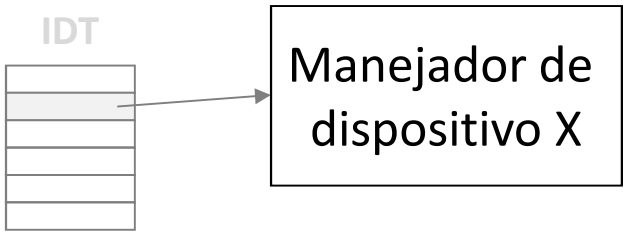
Interrupciones hardware
tratamiento (2/5)




```
#include "servicios.h"

int main ()
{
    for (int i=0; i<1000000; i++)
        printf("resultado = %d\n",calculo_complejo(i));
    return 0;
}
```

Modo Usuario

Modo Kernel



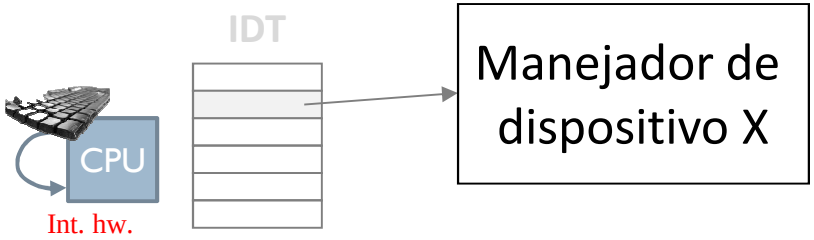
Interrupciones hardware
tratamiento (3/5)

Aplicación

```
#include "servicios.h"

int main ()
{
    for (int i=0; i<1000000; i++)
        printf("resultado = %d\n",calculo_complejo(i));
    return 0;
}
```

Modo Usuario

Modo Kernel

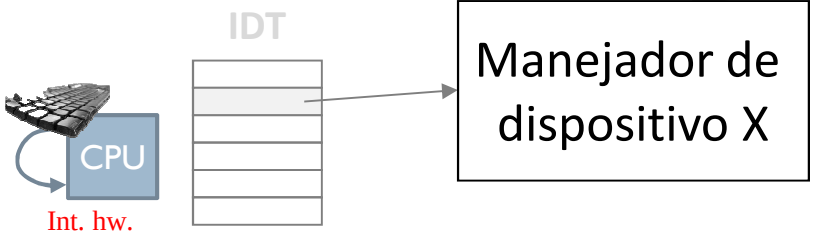


- Primero salva estado básico (PC, RE, SP) en la pila de sistema
- La CPU pasa en modo privilegiado y salta a la rutina de tratamiento asociada

Interrupciones hardware
tratamiento (4/5)

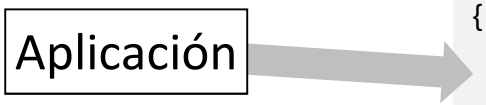
Modo Usuario

Modo Kernel




```
void interrupcionDispositivo ()
{
    ▸ Salvar estado (si es necesario)
    ▸ La subrutina trata el evento:
        ▸ Realiza lo urgente
        ▸ Programa una tarea pendiente
          (si necesario)
    ▸ Restaura el estado (si necesario)
    ▸ Ejecuta instrucción de retorno de interrupción (RETI)
        ▸ Restaura estado básico y modo.
}
```

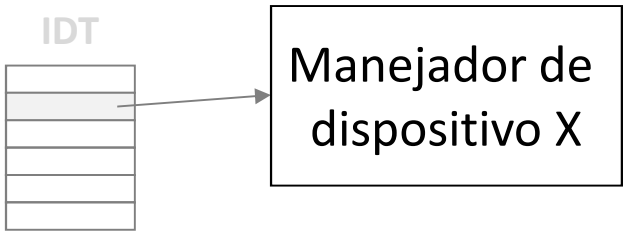
Interrupciones hardware
tratamiento (5/5)



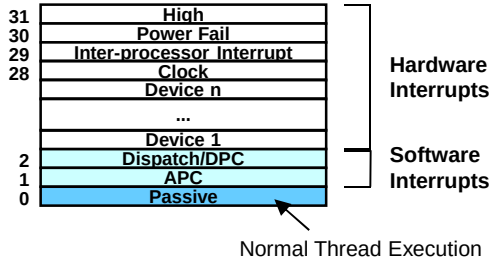

```
#include "servicios.h"

int main ()
{
    for (int i=0; i<1000000; i++)
        printf("resultado = %d\n",calculo_complejo(i));
    return 0;
}
```

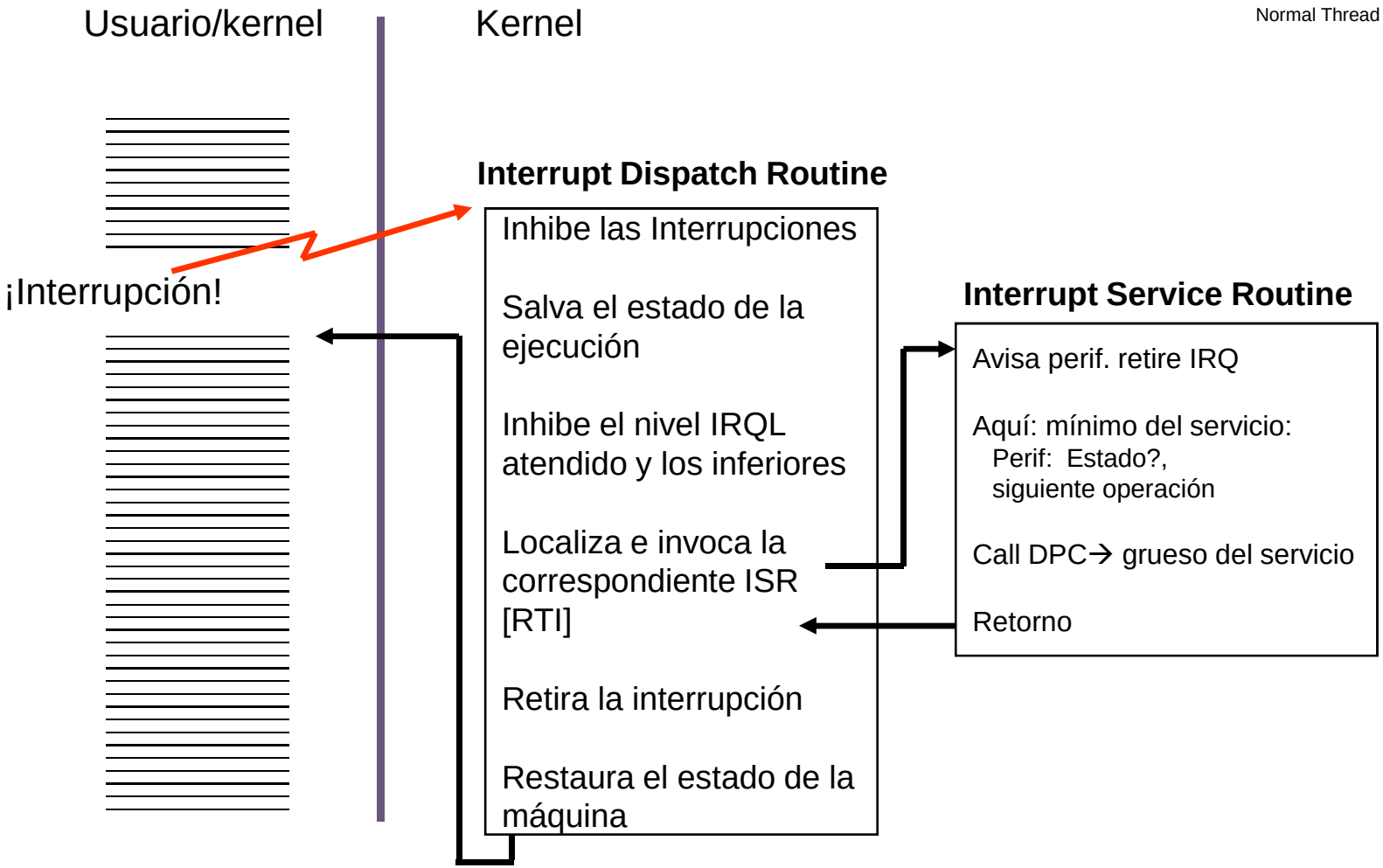
Modo Usuario

Modo Kernel



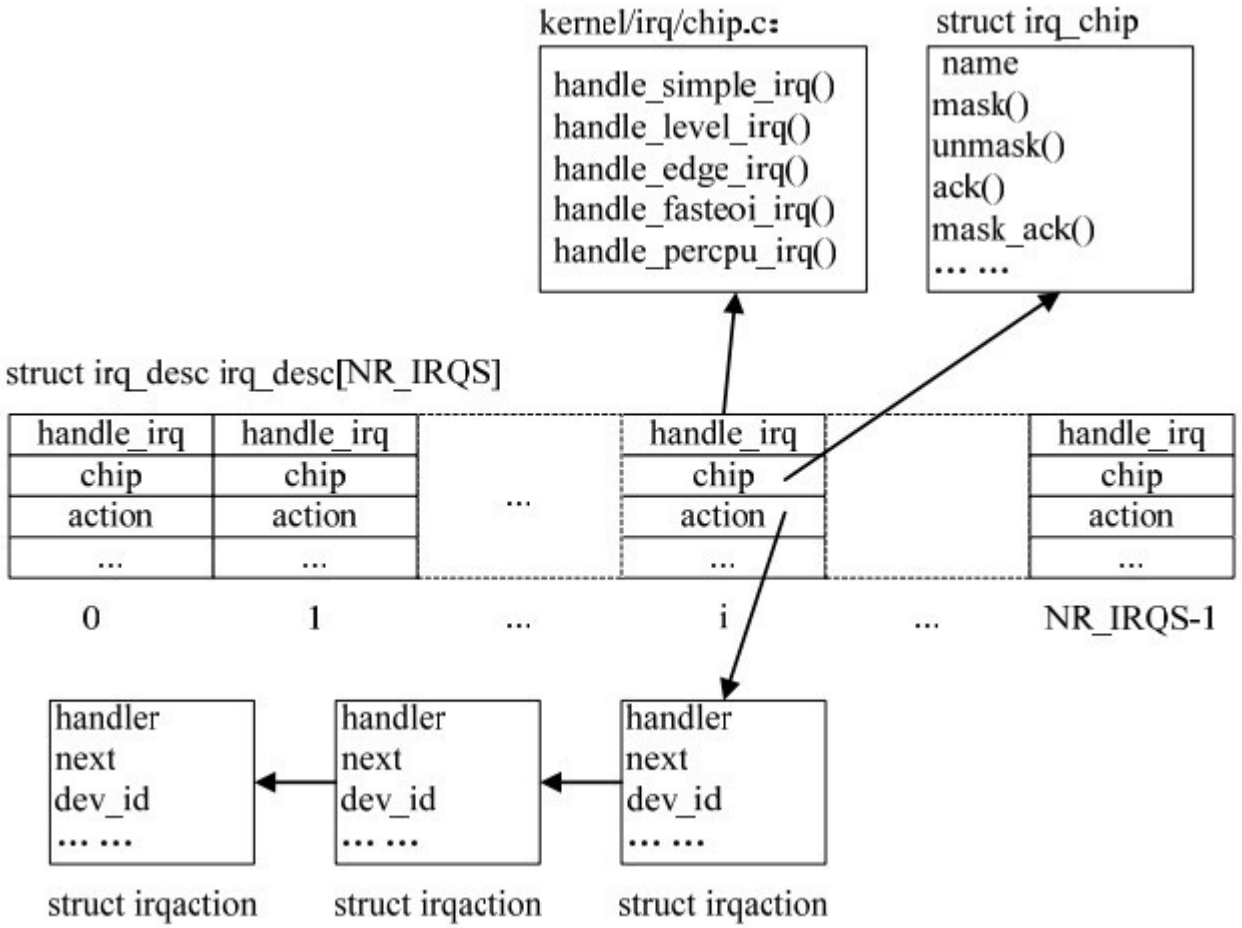
Interrupciones hardware
tratamiento en Windows





Inside Windows 2000 (página 104)

Interrupciones hardware
tratamiento en Linux



http://chxxxyg.blog.163.com/blog/static/150281193201062701509 8/

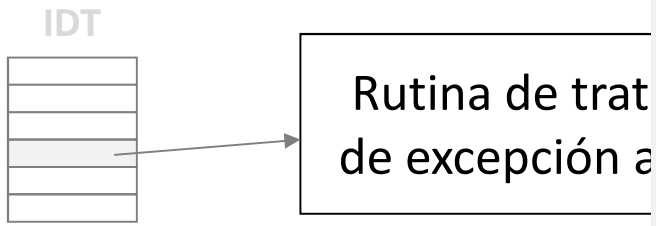
Excepciones
características

- Eventos síncronos de carácter excepcional al ejecutar una instrucción
- Modo de ejecución previo:
  - Puede ser usuario o sistema (SI influye en el tratamiento)
- Generadas por:
  - El hardware, normalmente errores de programación
    - NO siempre son errores (fallo de página, depuración, etc.)

Excepciones
tratamiento (1/4)

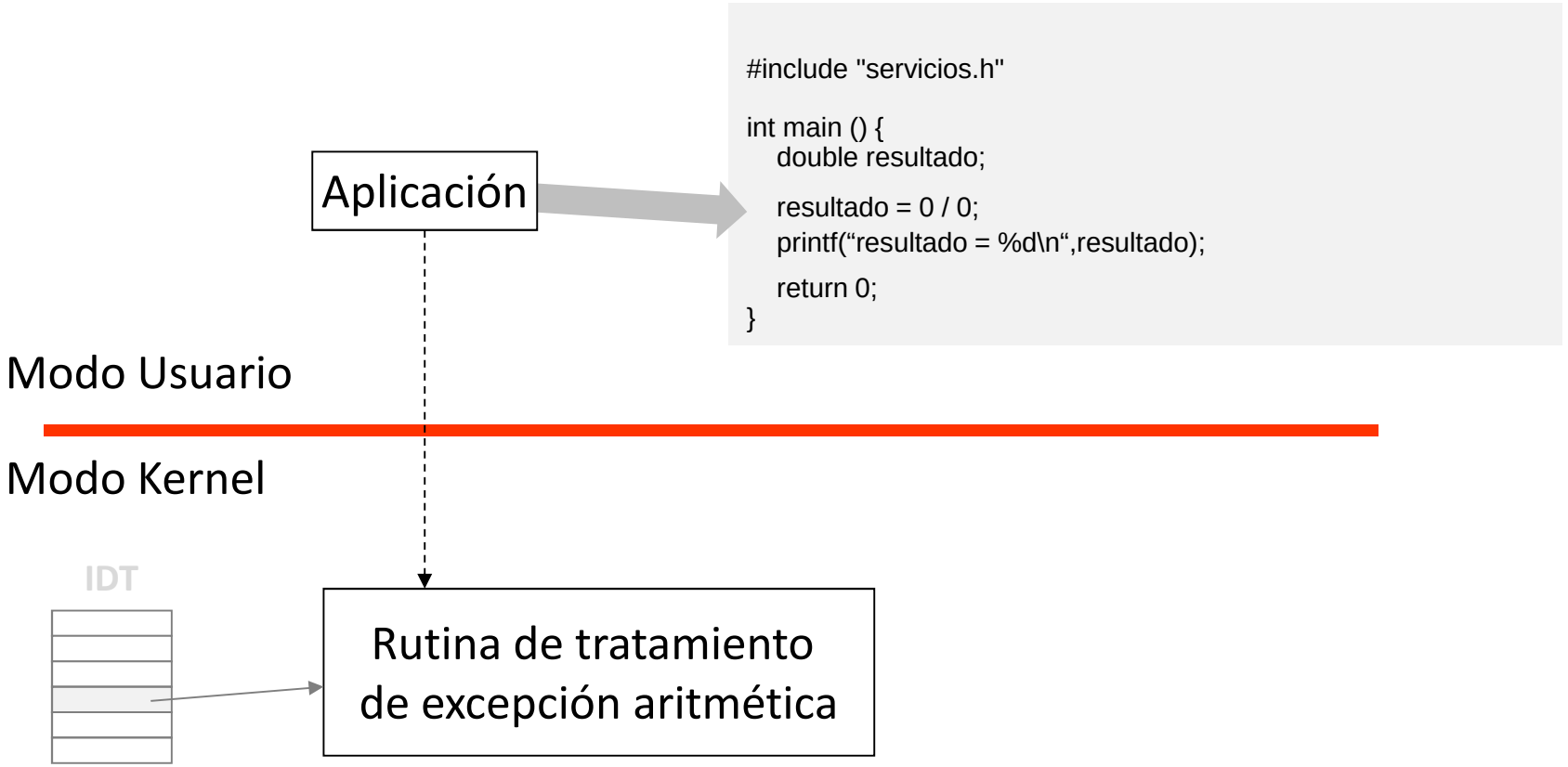
Modo Usuario

Modo Kernel



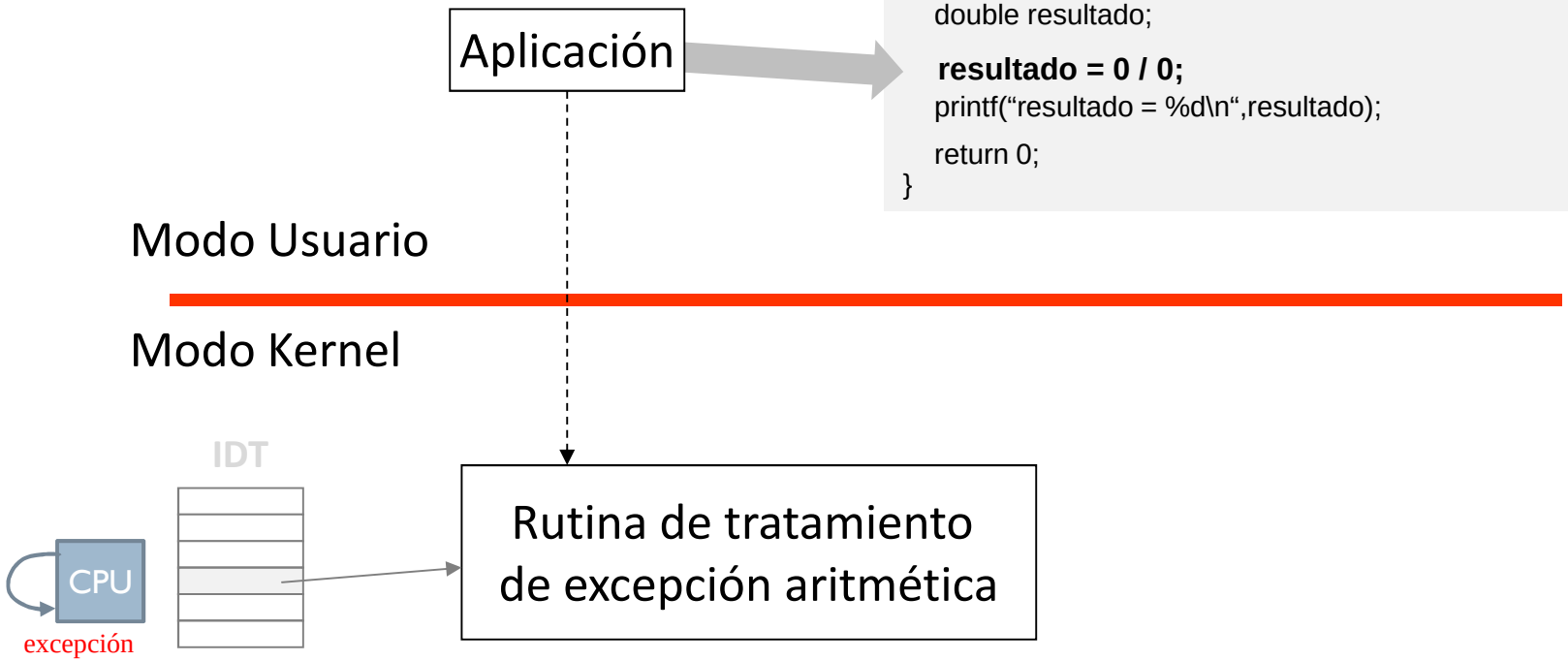

```
int main (int argc, char **argv)
{
      ...
      /* instalar los manejadores para los vectores de interrupción */
      instal_man_int(EXC_ARITMETICA,   excepcionAritmetica);
      instal_man_int(EXC_MEMORIA,      excepcionMemoria);
      instal_man_int(INT_RELOJ,        interrupcionReloj);
      instal_man_int(INT_DISPOSITIVOS, interrupcionDispositivos);
      instal_man_int(LLAM_SISTEMA,     tratarLlamadaSistema);
      instal_man_int(INT_SW,           interrupcionSoftware);
      ...
```

Excepciones
tratamiento (2/4)



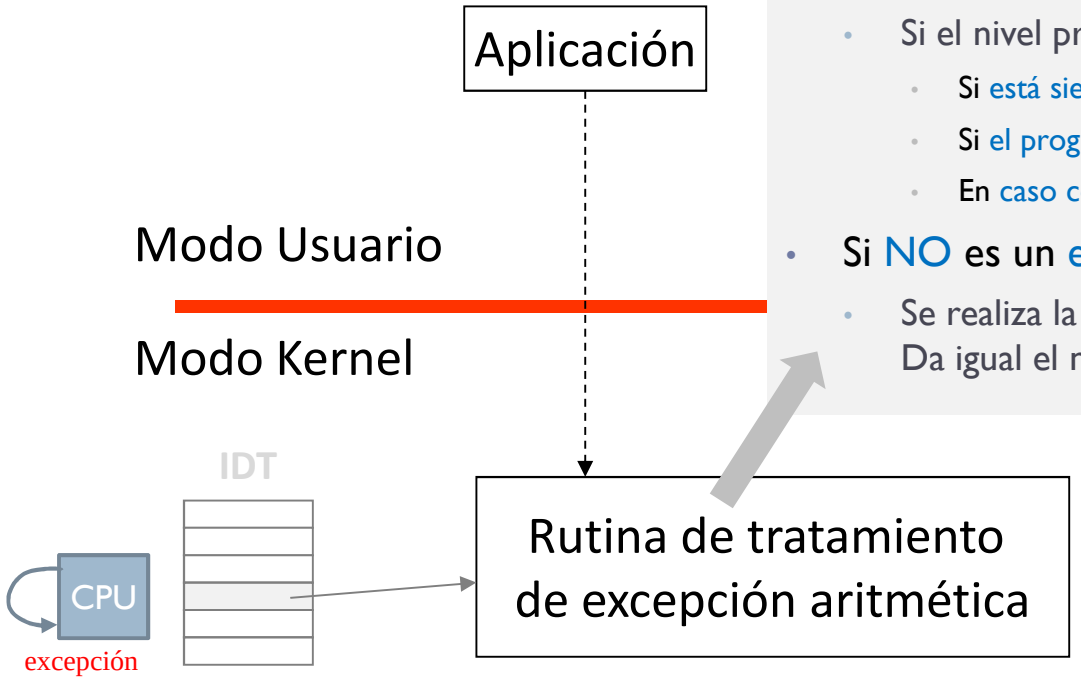
Excepciones
tratamiento (3/4)

```
#include "servicios.h"

int main () {
    double resultado;
    resultado = 0 / 0;
    printf("resultado = %d\n",resultado);
    return 0;
}
```



- Primero salva estado básico (PC, RE, SP) en la pila de sistema
- La CPU pasa en modo privilegiado y salta a la rutina de tratamiento asociada

Excepciones
tratamiento (4/4)



- Si es un error:
  - Si el nivel previo de la CPU era de sistema:
    - Pánico: error en el código del S.O. => mensaje + detener el S.O.
  - Si el nivel previo de la CPU era de usuario:
    - Si está siendo depurado, se notifica a depurador
    - Si el programa establece un manejador de la excepción, ejecutarlo
    - En caso contrario, se aborta el proceso
- Si NO es un error: (Ej.: fallo de página previsto)
  - Se realiza la tarea prevista (Ej.: asignar una nueva pagina)
    Da igual el nivel previo de ejecución

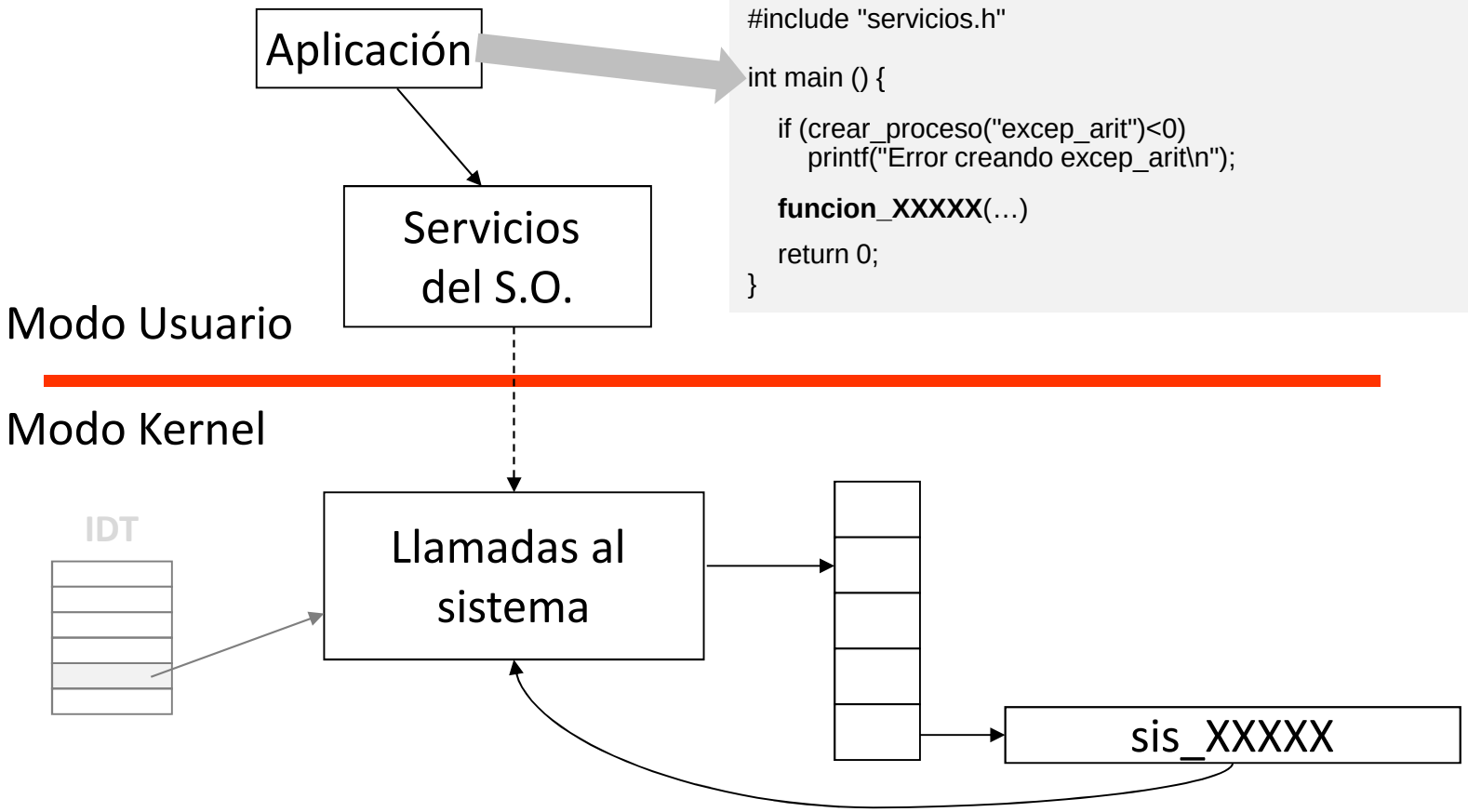
Llamadas al sistema
características

- Eventos síncronos de solicitud de servicio del sistema operativo con una instrucción no privilegiada
- Modo de ejecución previo:
  - Siempre usuario
- Generadas por:
  - Los programas

Llamadas al sistema
tratamiento

```
int main (int argc, char **argv)
{
        ...

        /* instalar los manejadores para los vectores de interrupción */
        instal_man_int(EXC_ARITMETICA,    excepcionAritmetica);
        instal_man_int(EXC_MEMORIA,       excepcionMemoria);
        instal_man_int(INT_RELOJ,         interrupcionReloj);
        instal_man_int(INT_DISPOSITIVOS, interrupcionDispositivos);
        instal_man_int(LLAM_SISTEMA,      tratarLlamadaSistema);
        instal_man_int(INT_SW,            interrupcionSoftware);

        ...
```

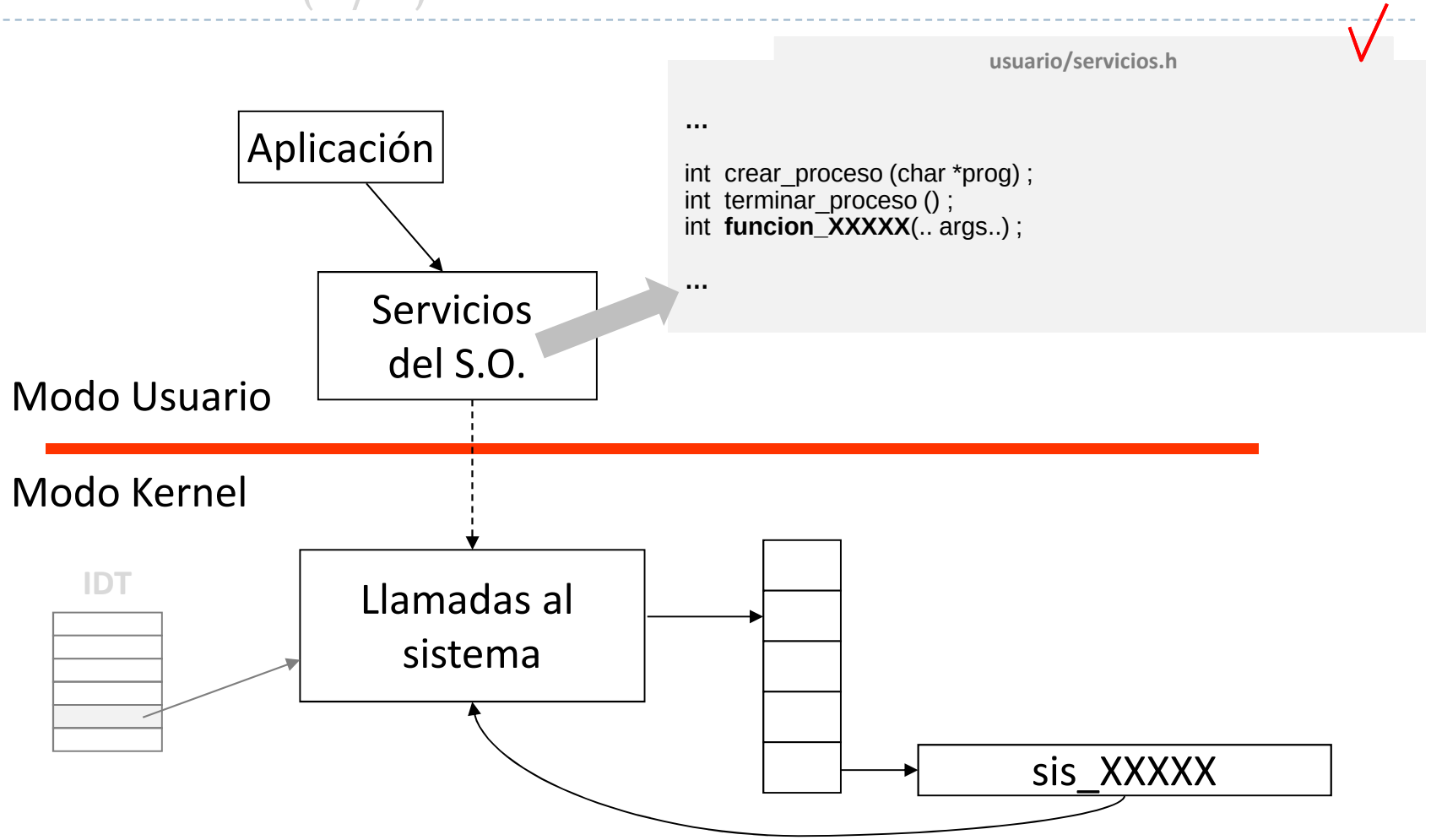
Llamadas al sistema
tratamiento (1/9)



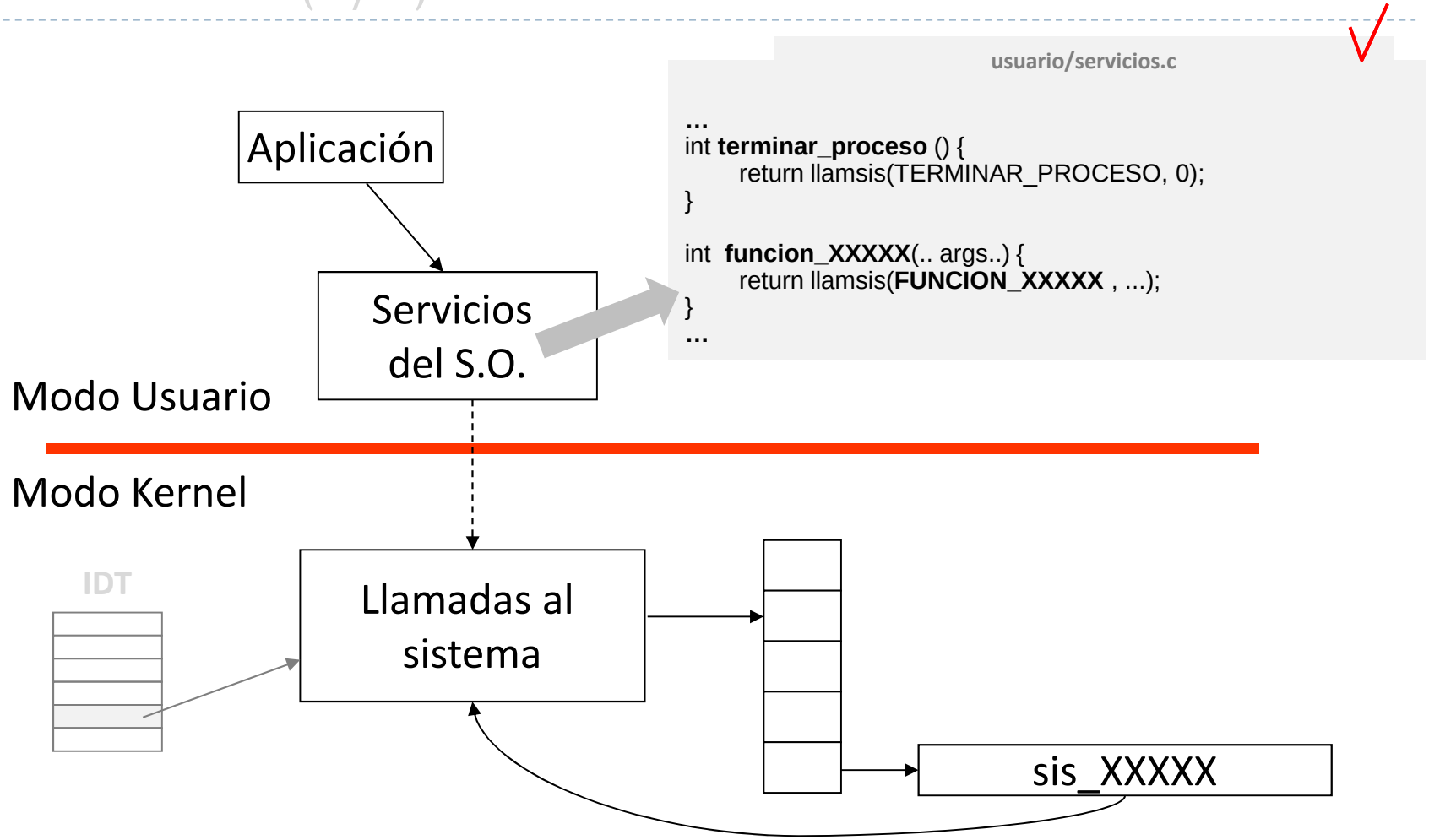
usuario/aplicación-01.c

```
#include "servicios.h"

int main () {

    if (crear_proceso("excep_arit")<0)
        printf("Error creando excep_arit\n");

    funcion_XXXXX(…)

    return 0;
}
```

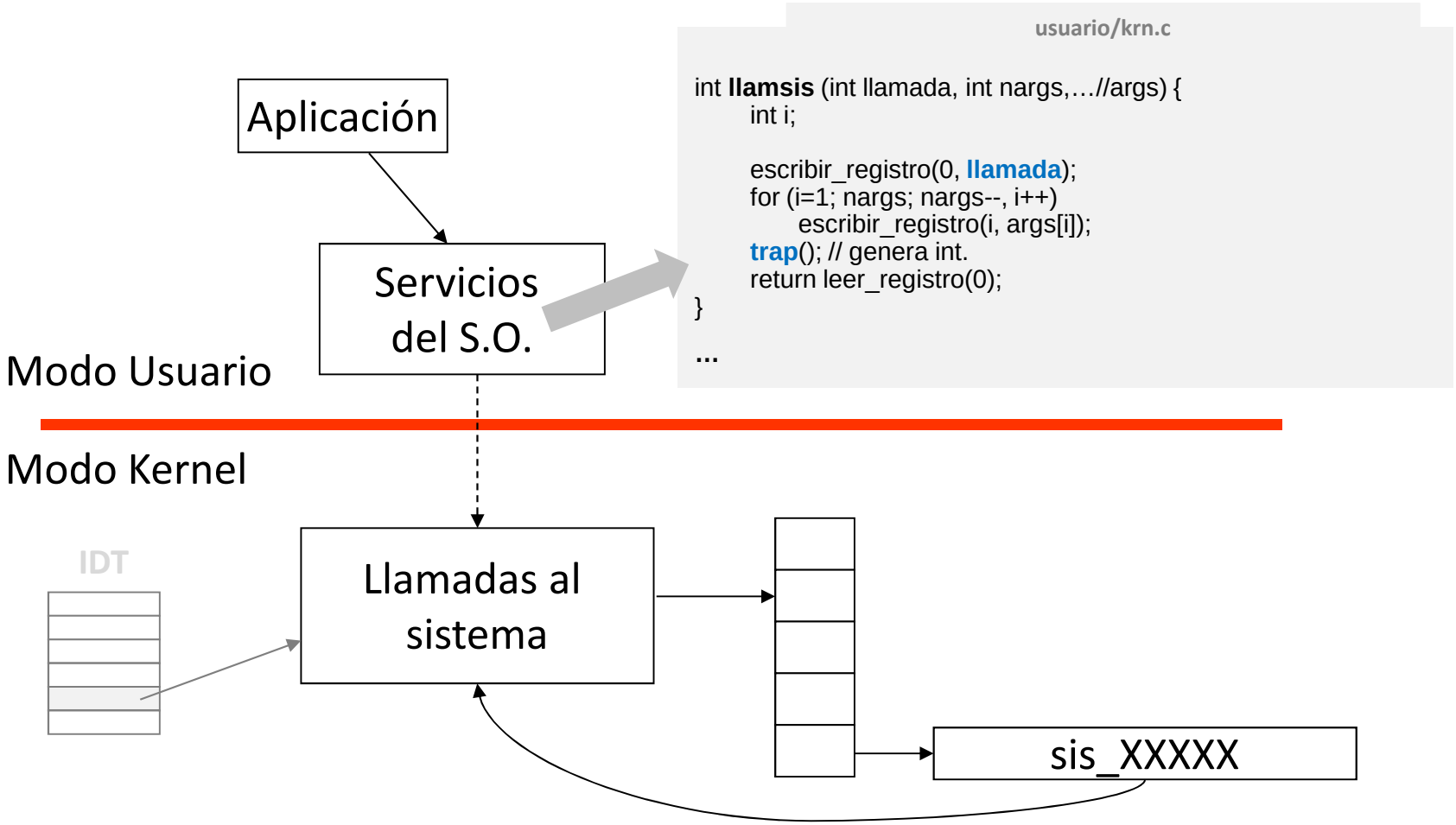
Llamadas al sistema
tratamiento (2/9)



Llamadas al sistema
tratamiento (3/9)



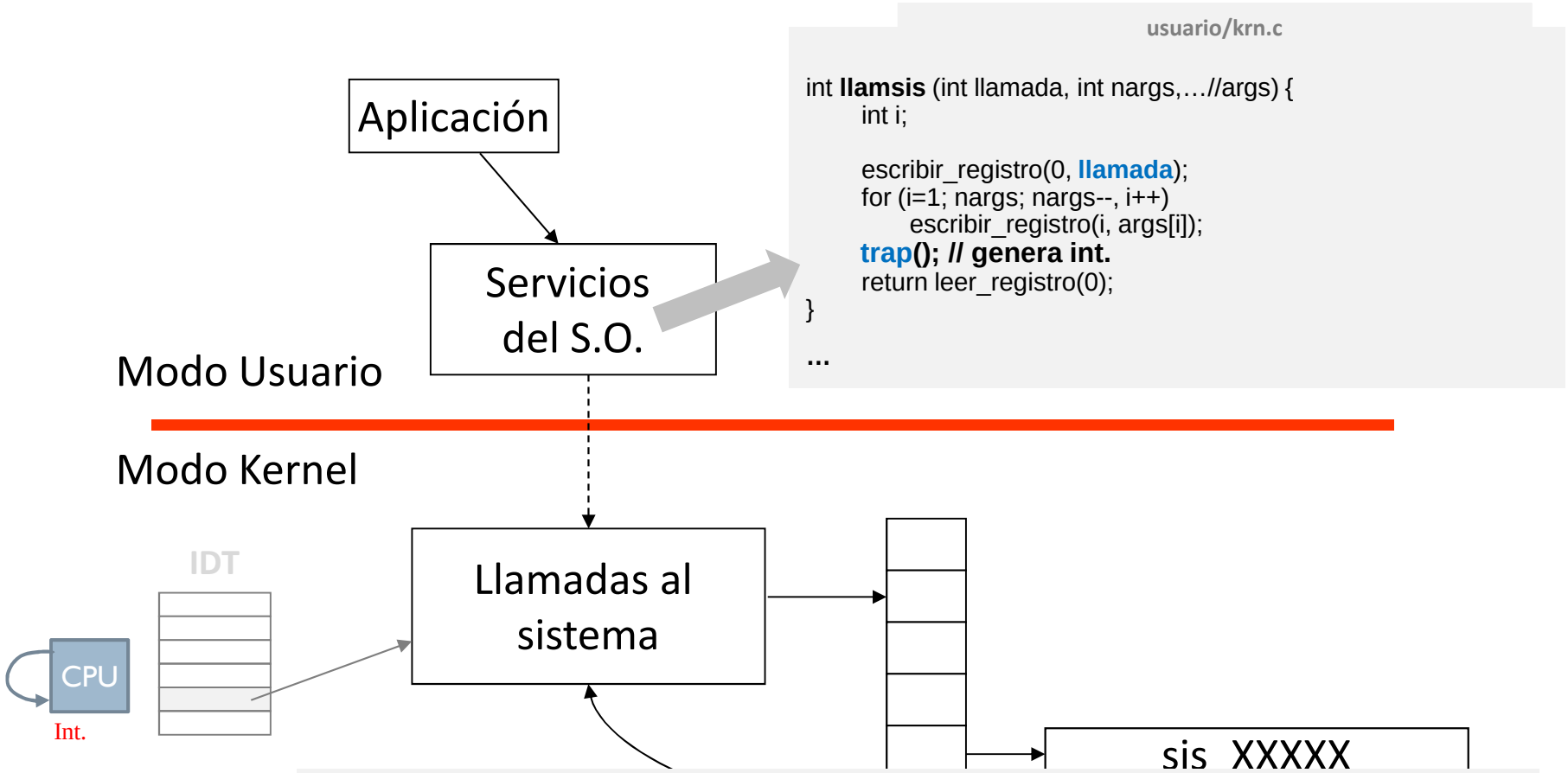
Llamadas al sistema
tratamiento (4/9)



usuario/krn.c

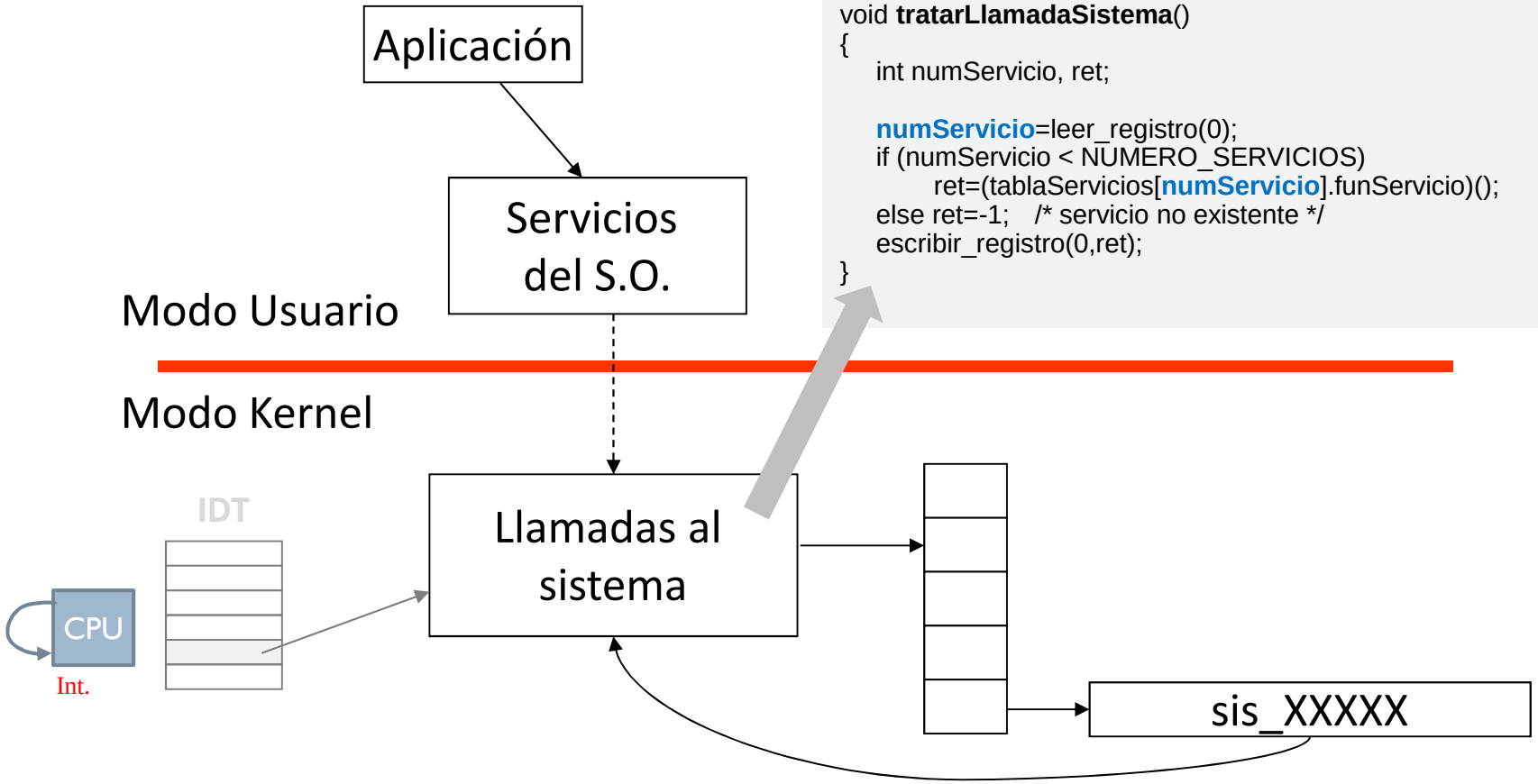
```
int llamsis (int llamada, int nargs,…//args) {
        int i;

        escribir_registro(0, llamada);
        for (i=1; nargs; nargs--, i++)
              escribir_registro(i, args[i]);
        trap(); // genera int.
        return leer_registro(0);
}
...
```

Llamadas al sistema
tratamiento (5/9)



- Primero salva estado básico (PC, RE, SP) en la pila de sistema
- La CPU pasa en modo privilegiado y salta a la rutina de tratamiento asociada

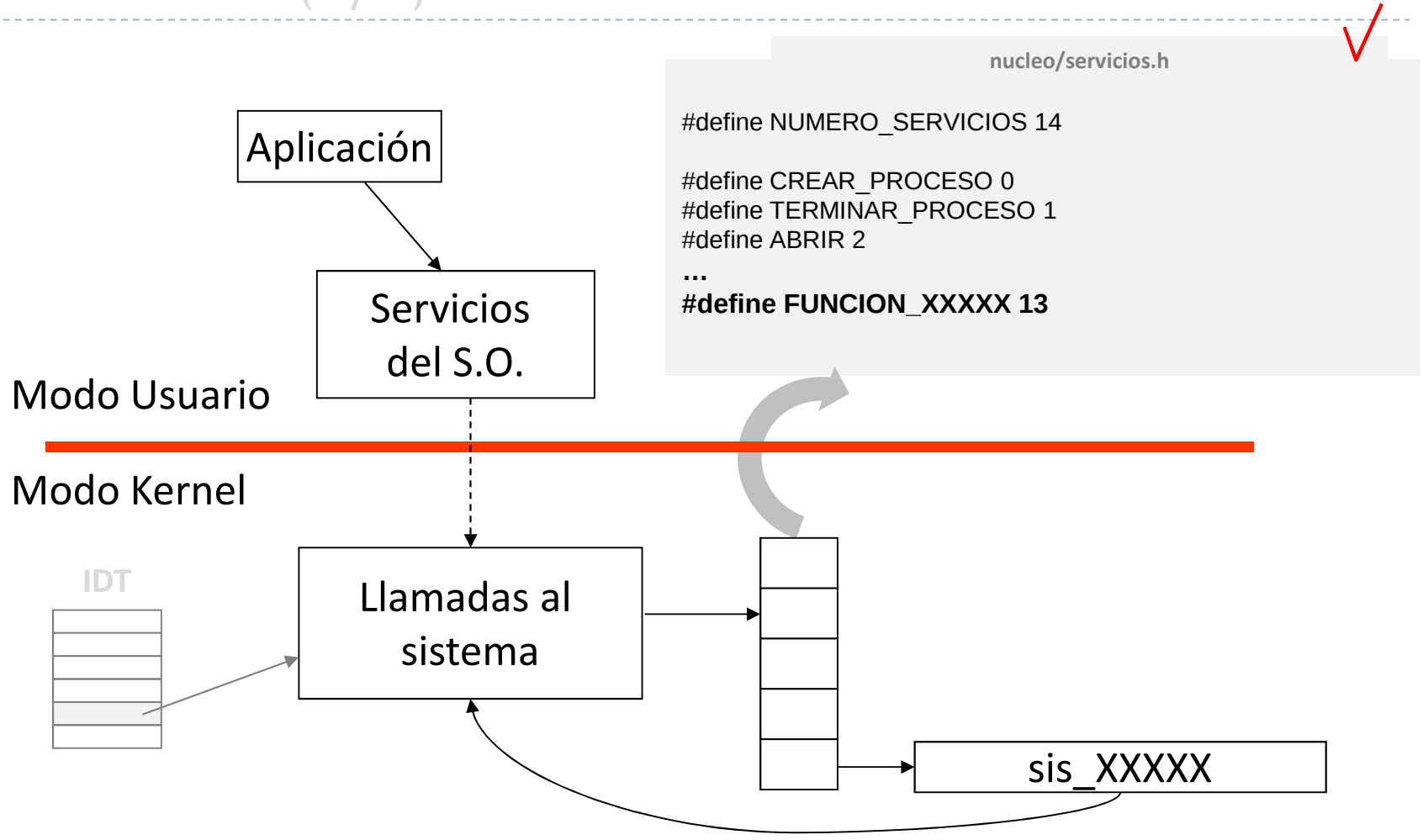
Llamadas al sistema
tratamiento (6/9)



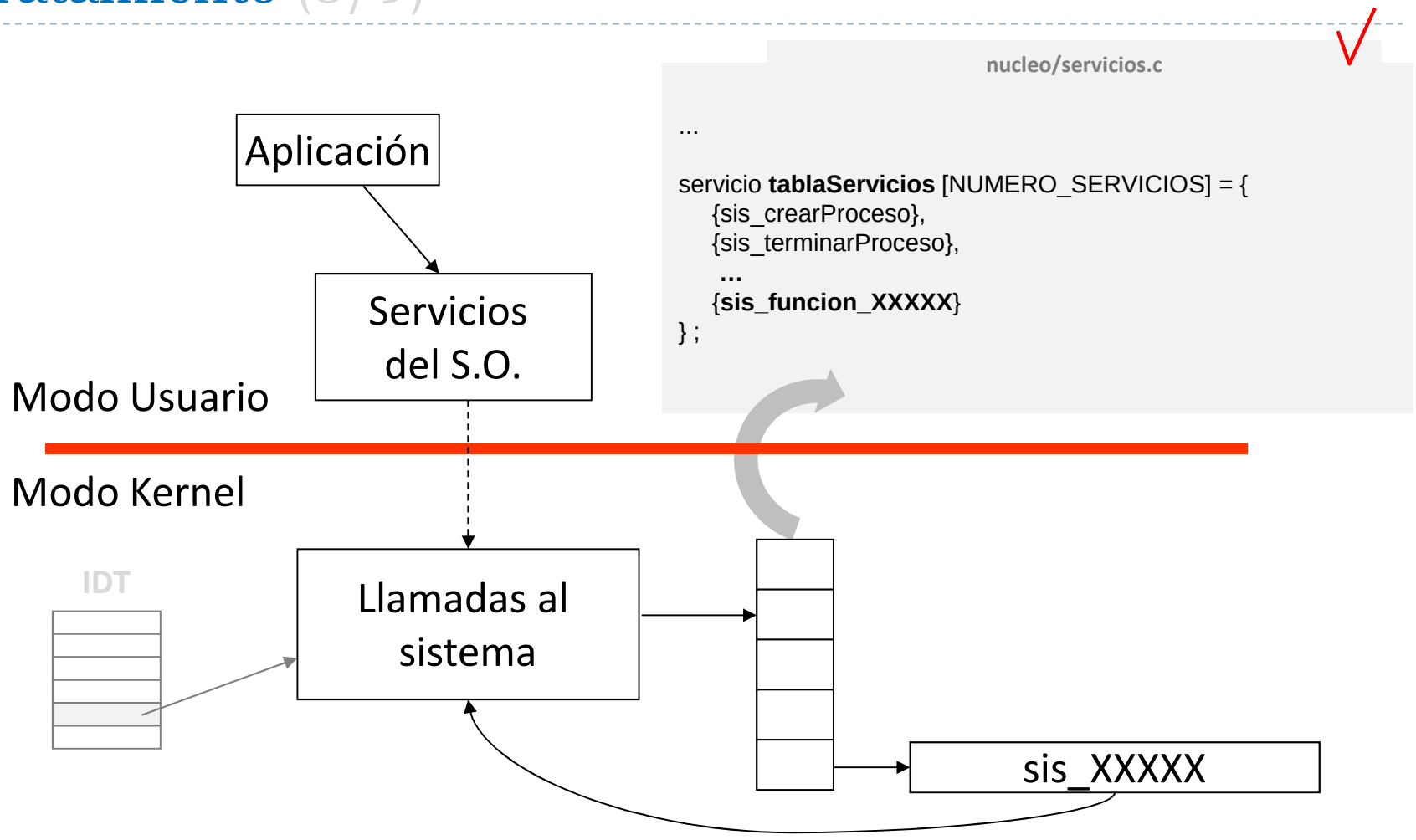
nucleo/servicios.c

```
void tratarLlamadaSistema()
{
    int numServicio, ret;

    numServicio=leer_registro(0);
    if (numServicio < NUMERO_SERVICIOS)
        ret=(tablaServicios[numServicio].funServicio)();
    else ret=-1;    /* servicio no existente */
    escribir_registro(0,ret);
}
```

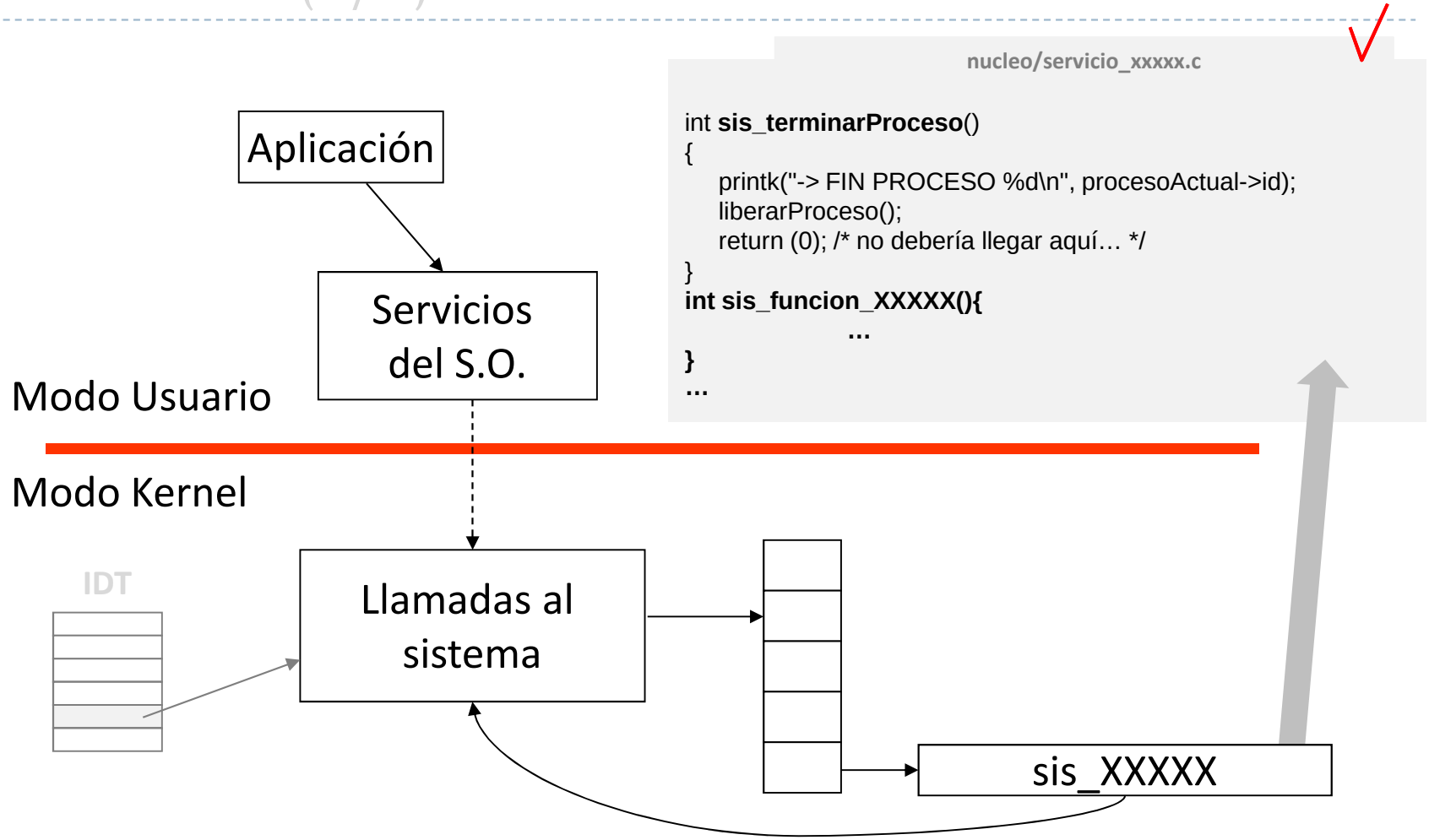
Llamadas al sistema
tratamiento (7/9)



Llamadas al sistema
tratamiento (8/9)



Llamadas al sistema
tratamiento (9/9)

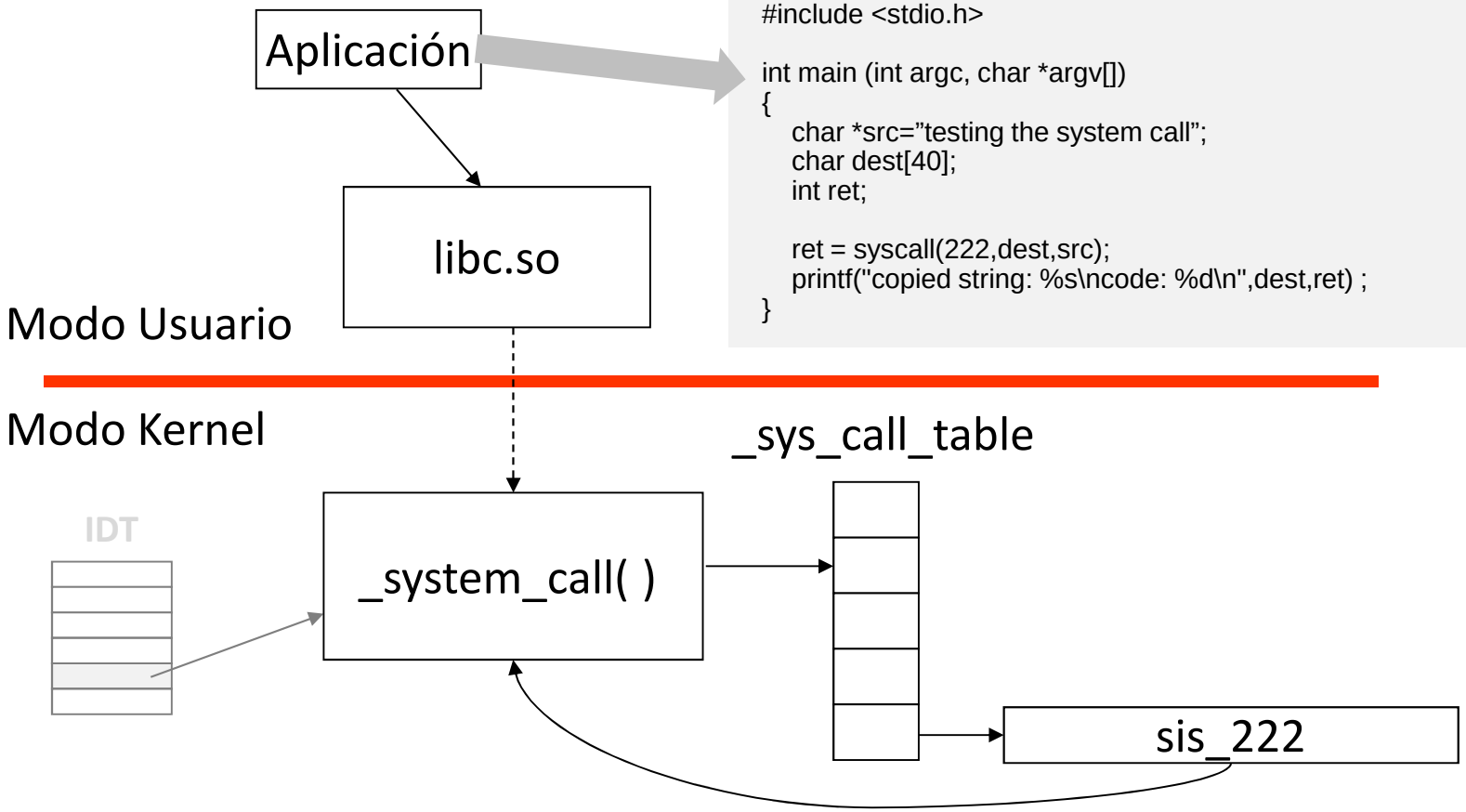


Llamadas al sistema
tratamiento en Linux (1/7)

/usr/src/linux/arch/x86/kernel/traps.c

```
void __init trap_init(void)
{
        ...
        set_intr_gate(X86_TRAP_DE, divide_error);
        set_intr_gate(X86_TRAP_NP, segment_not_present);
        set_intr_gate(X86_TRAP_GP, general_protection);
        set_intr_gate(X86_TRAP_SPURIOUS, spurious_interrupt_bug);
        set_intr_gate(X86_TRAP_MF, coprocessor_error);
        set_intr_gate(X86_TRAP_AC, alignment_check);

#ifdef CONFIG_IA32_EMULATION
        set_system_intr_gate(IA32_SYSCALL_VECTOR, ia32_syscall);
        set_bit(IA32_SYSCALL_VECTOR, used_vectors);
#endif

#ifdef CONFIG_X86_32
        set_system_trap_gate(SYSCALL_VECTOR, &system_call);
        set_bit(SYSCALL_VECTOR, used_vectors);
#endif
        ...
```

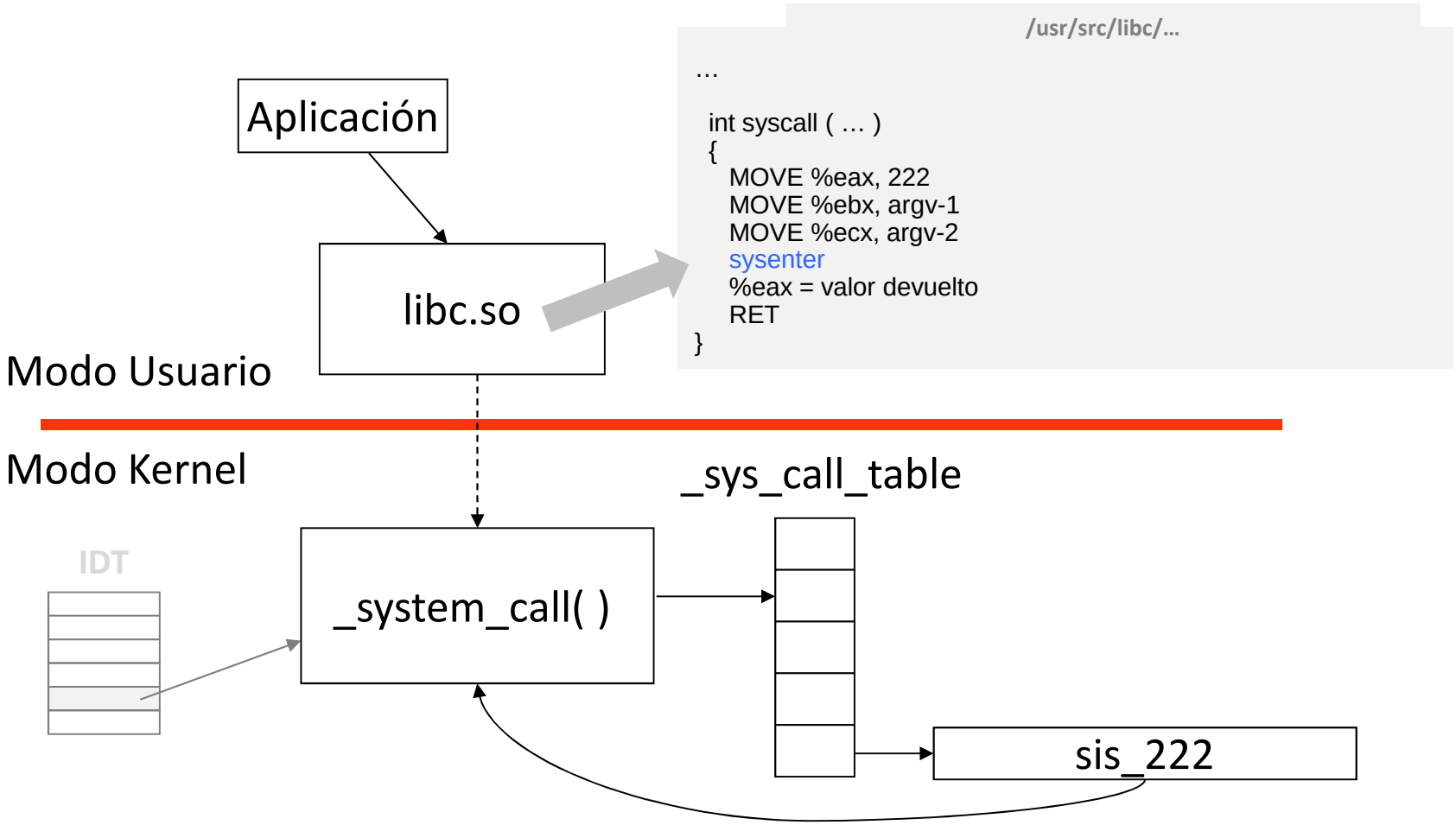
Llamadas al sistema
tratamiento en Linux (2/7)



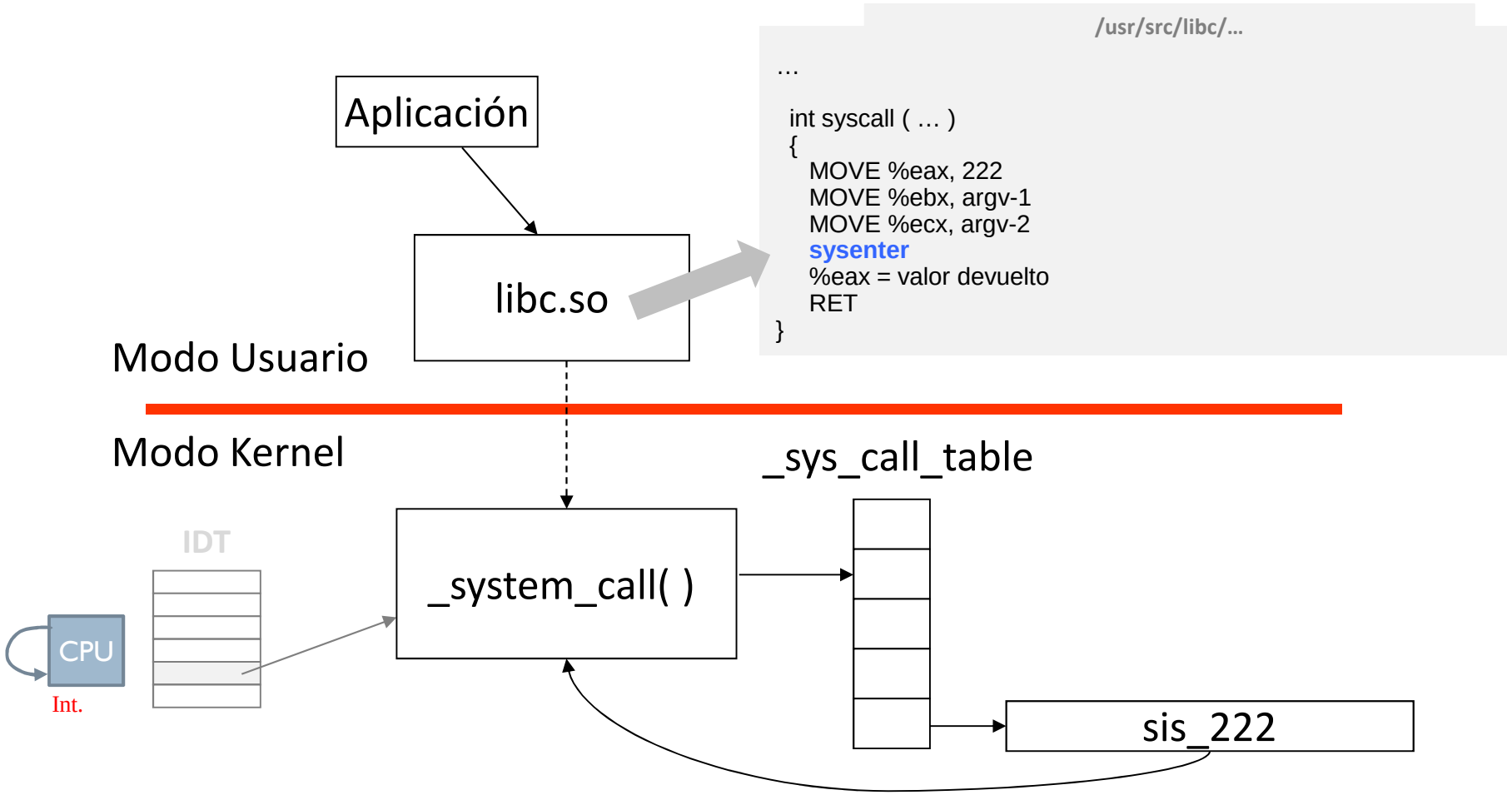
/usr/src/linux/test/test1.c

```
#include <stdio.h>

int main (int argc, char *argv[])
{
    char *src="testing the system call";
    char dest[40];
    int ret;

    ret = syscall(222,dest,src);
    printf("copied string: %s\ncode: %d\n",dest,ret) ;
}
```

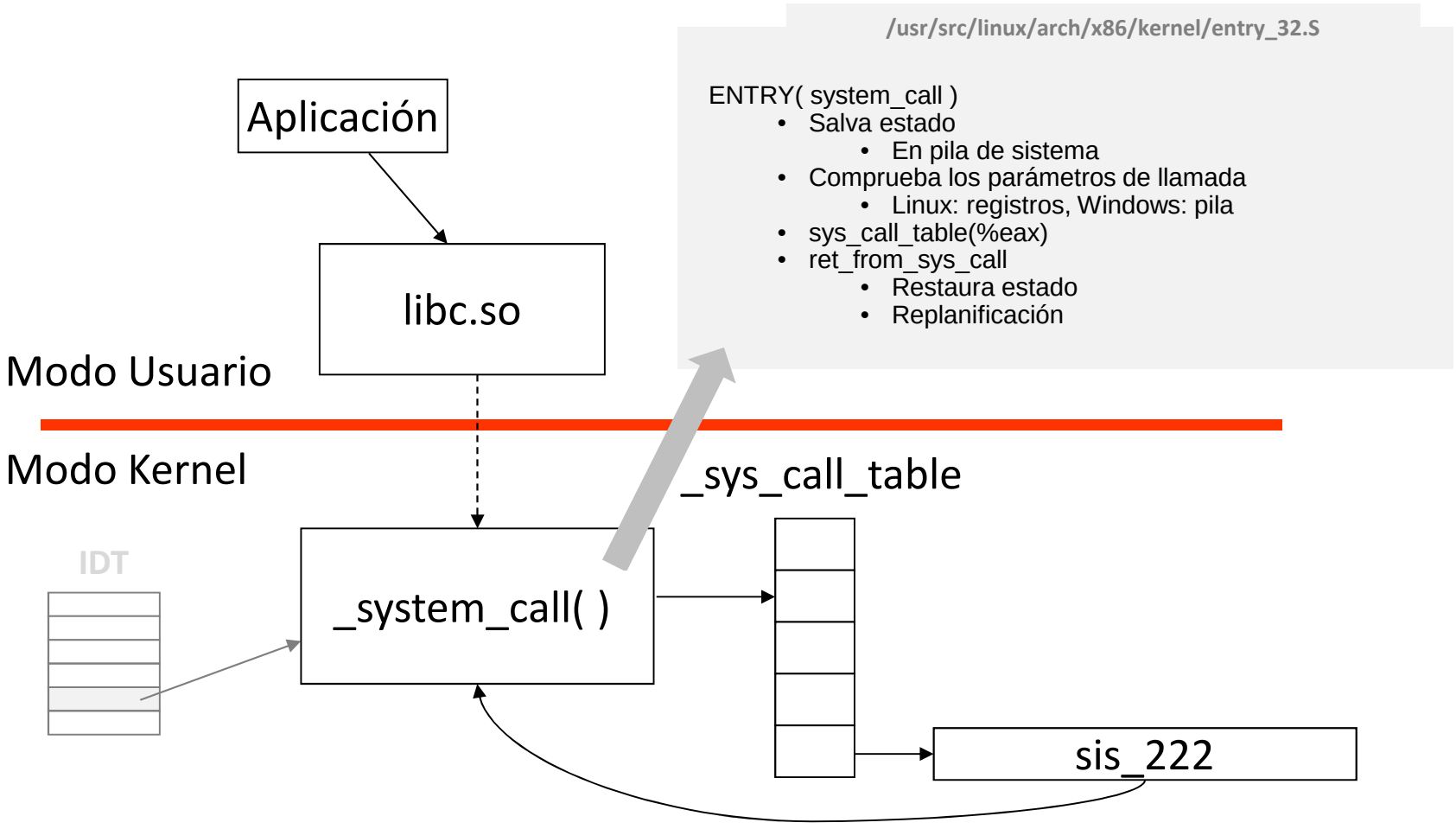
Llamadas al sistema
tratamiento en Linux (3/7)



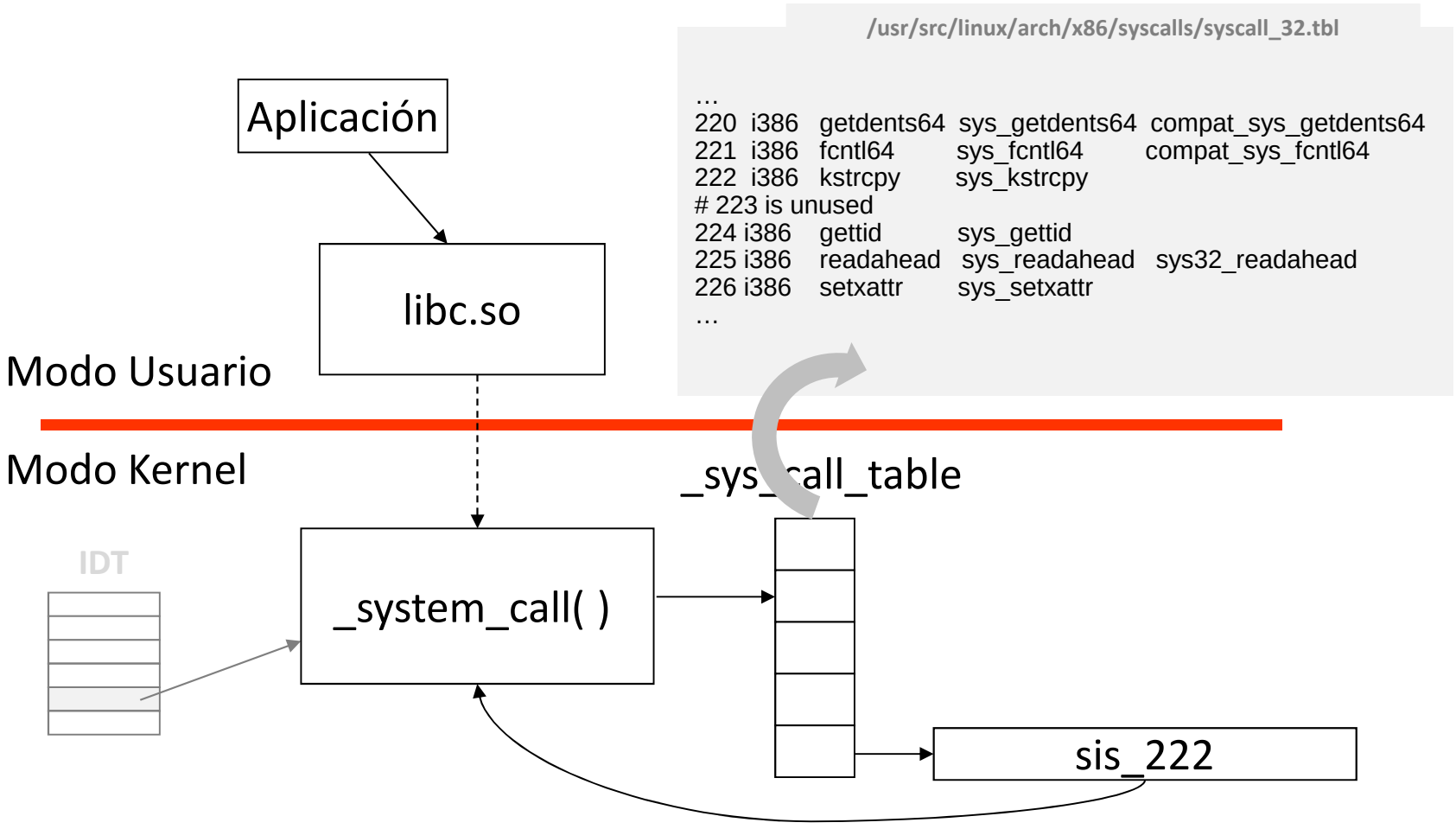
Llamadas al sistema
tratamiento en Linux (3/7)



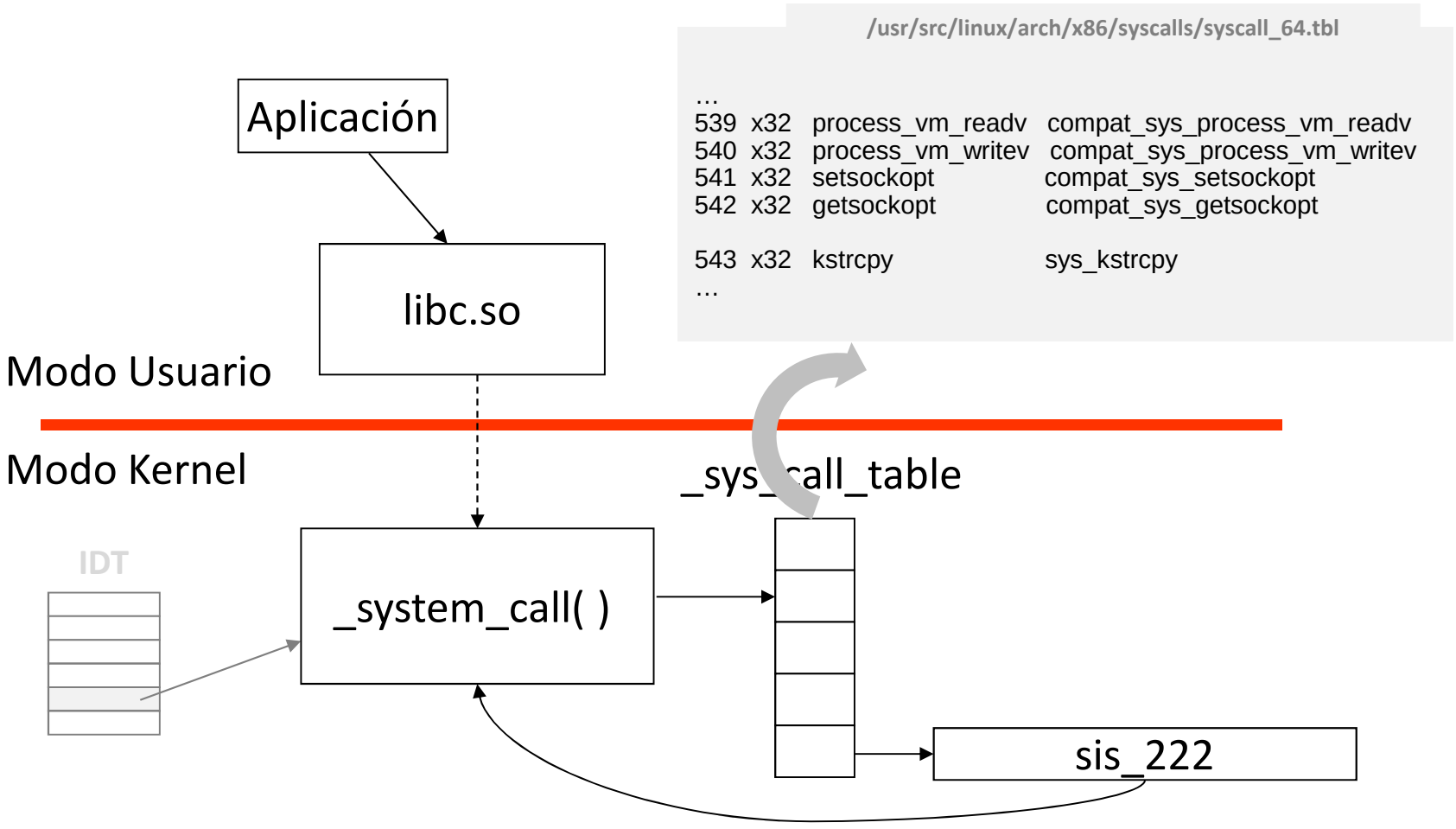
Llamadas al sistema
tratamiento en Linux (4/7)



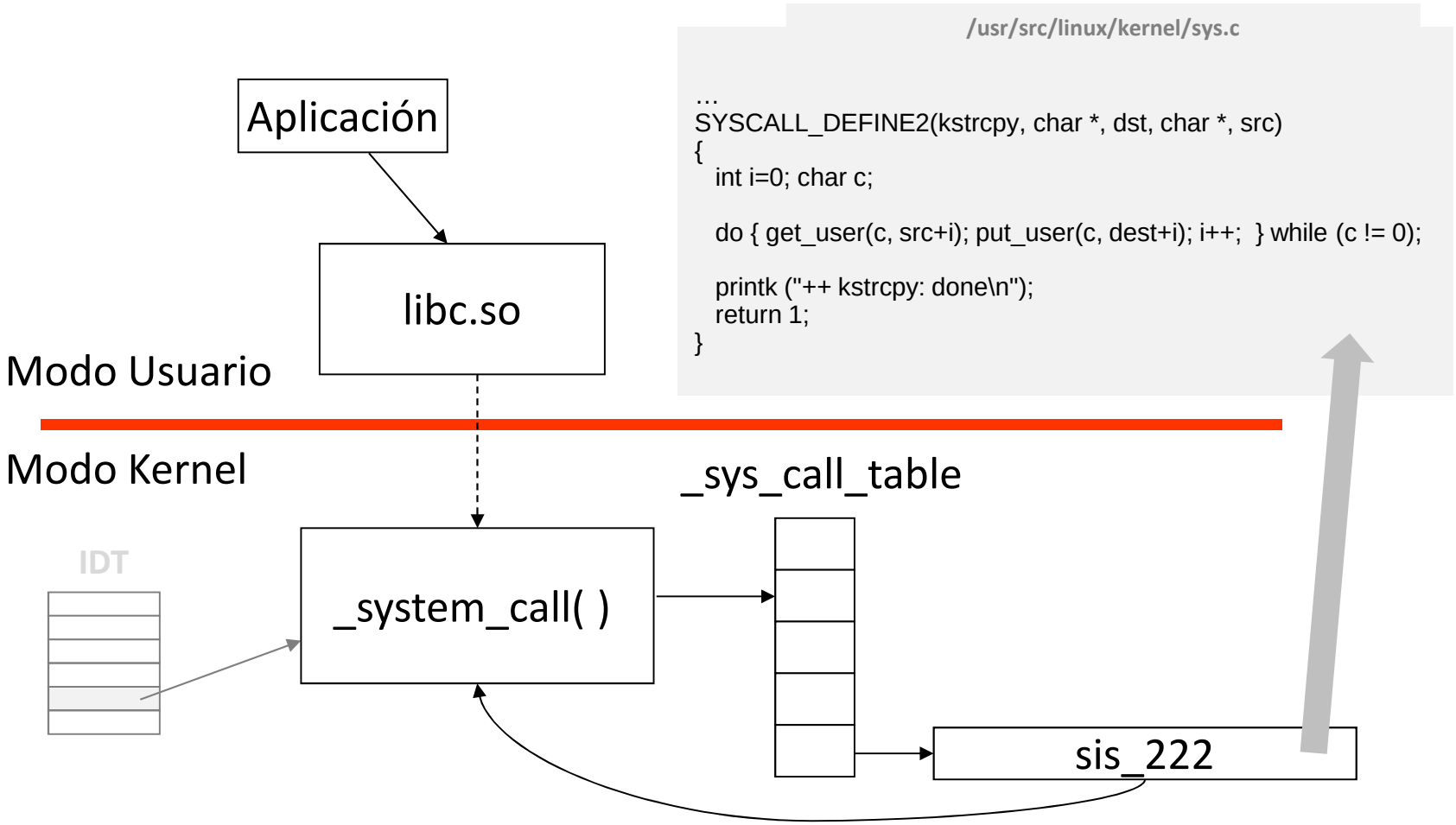
Llamadas al sistema
tratamiento en Linux (5/7)



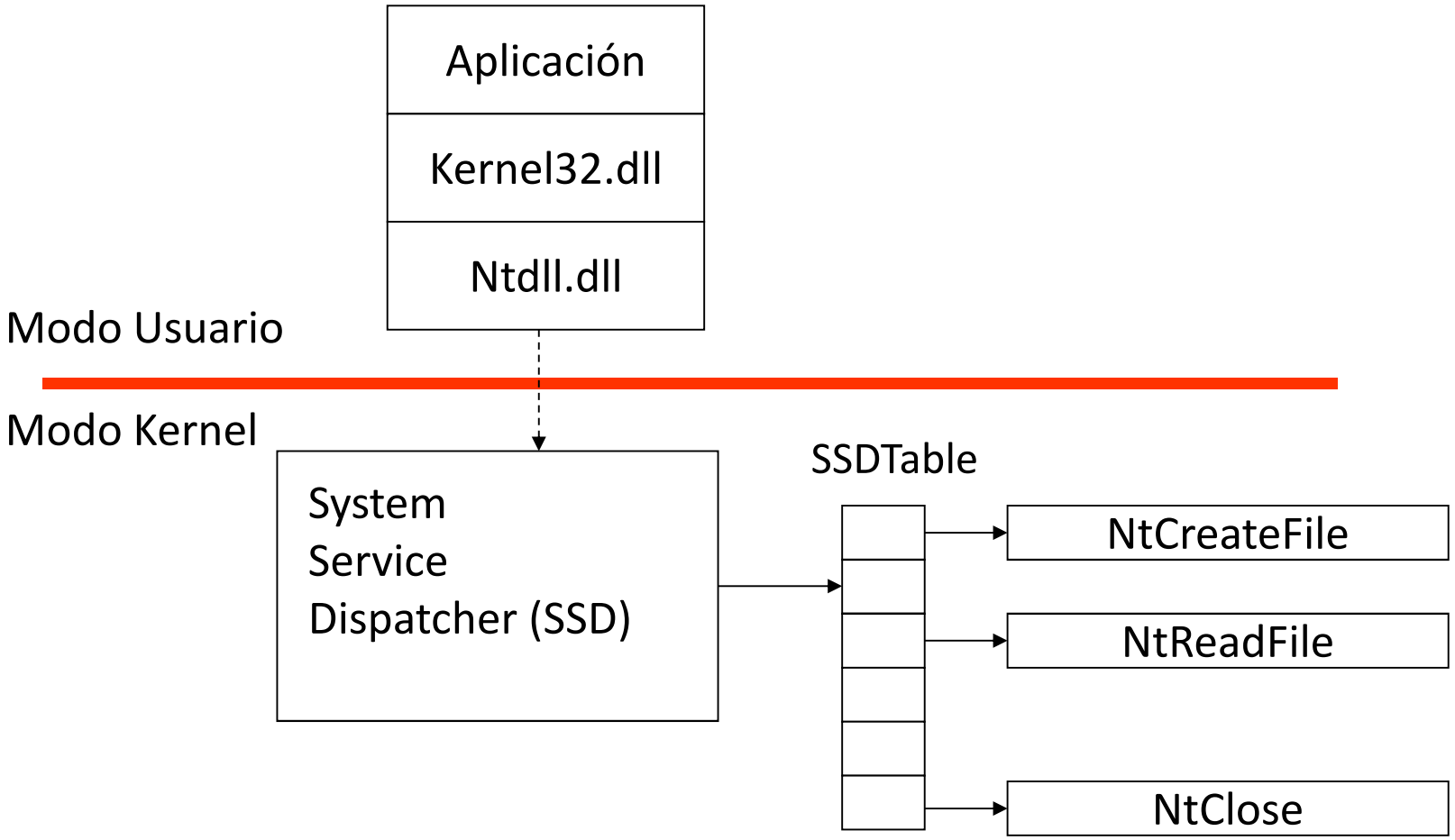
Llamadas al sistema
tratamiento en Linux (6/7)



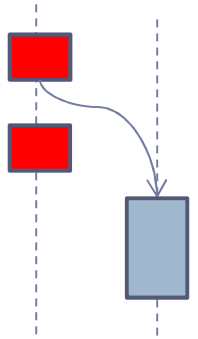
Llamadas al sistema
tratamiento en Linux (7/7)



Llamadas al sistema
tratamiento en Windows



Interrupción software
características

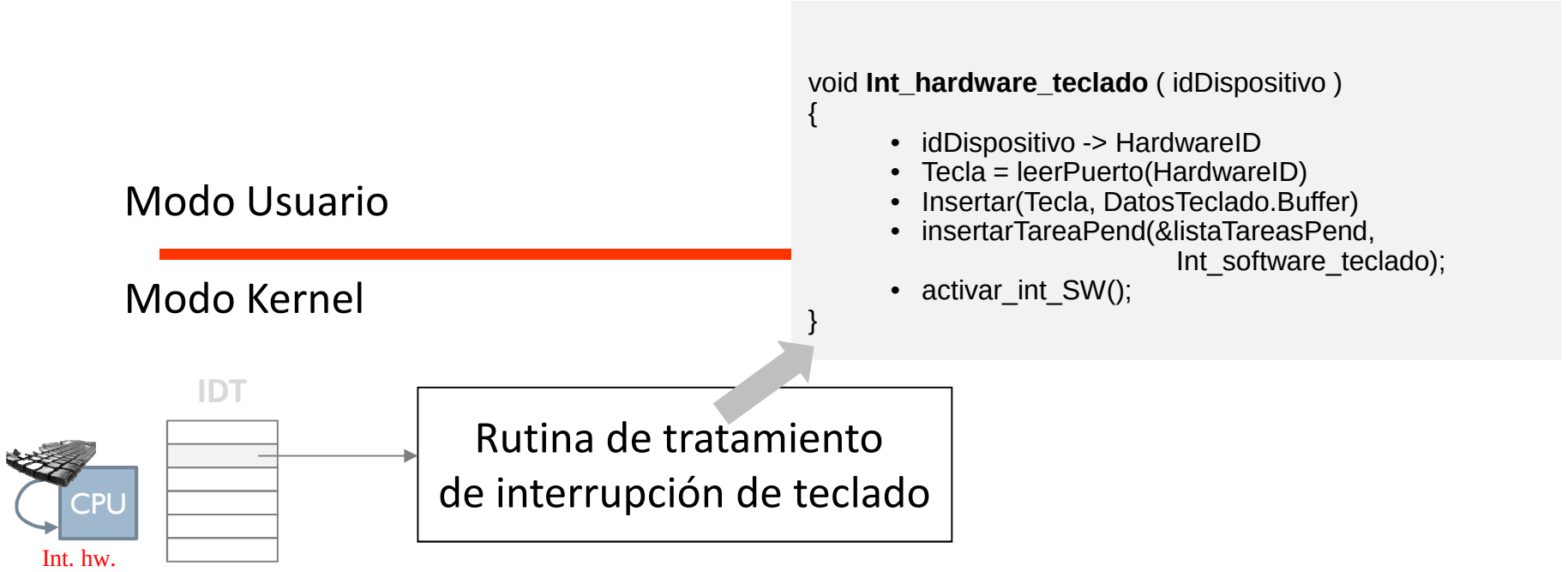


- Eventos asíncronos para tratar en diferido parte de la tarea asociada a un evento no crítica
  - Por esperar a circunstancias oportunas
  - Se hayan tratado el resto de eventos más urgentes
- Modo de ejecución previo:
  - Siempre sistema
- Generadas por:
  - El tratamiento de cualquiera de los eventos anteriores, se prepara una int. soft. para la parte no crítica

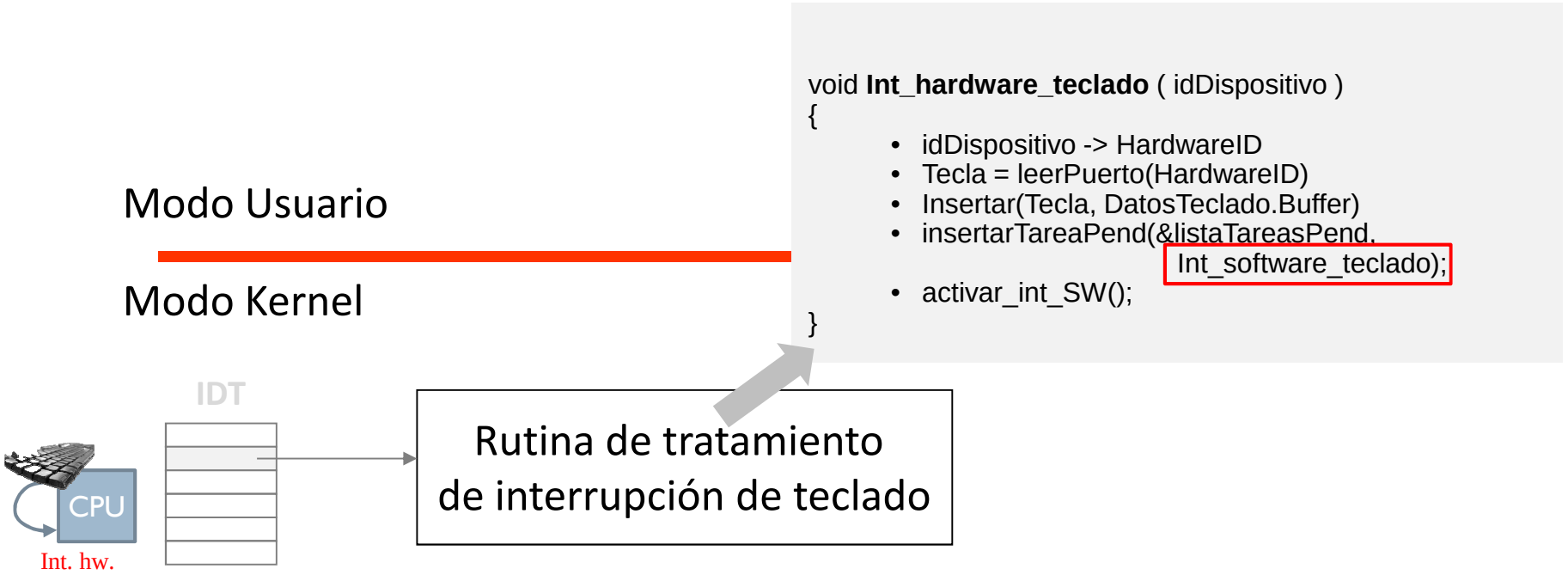
Interrupción software
tratamiento

```
int main (int argc, char **argv)
{
        ...

        /* instalar los manejadores para los vectores de interrupción */
        instal_man_int(EXC_ARITMETICA,    excepcionAritmetica);
        instal_man_int(EXC_MEMORIA,       excepcionMemoria);
        instal_man_int(INT_RELOJ,         interrupcionReloj);
        instal_man_int(INT_DISPOSITIVOS, interrupcionDispositivos);
        instal_man_int(LLAM_SISTEMA,      tratarLlamadaSistema);
        instal_man_int(INT_SW,            interrupcionSoftware);

        ...
```

Interrupción hardware
tratamiento (1/2)



Interrupción hardware
tratamiento (1/2)

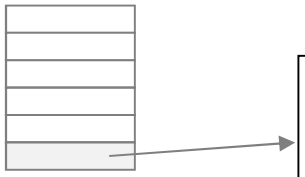


Interrupción software
tratamiento (1/2)

```
void Int_software_teclado ( idDispositivo )
{
    • idDispositivo -> DatosTeclado
    • P = ExtraerBCP(&(DatosTeclado.esperando))
    • Si P != NULL
        • P.estado = LISTO
        • Insertar(&ListaListos, P);

}
```

Modo Usuario

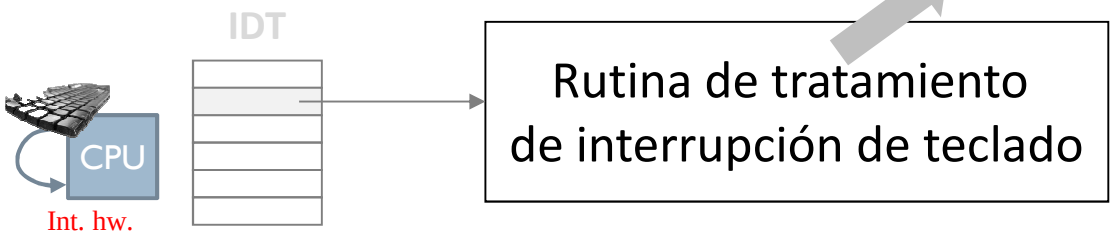
Modo Kernel



Rutina de tratamiento de interrupción software

Interrupción con la mínima prioridad: se ejecutará cuando no haya nada más urgente (crítico)

Interrupción hardware
tratamiento (2/2)



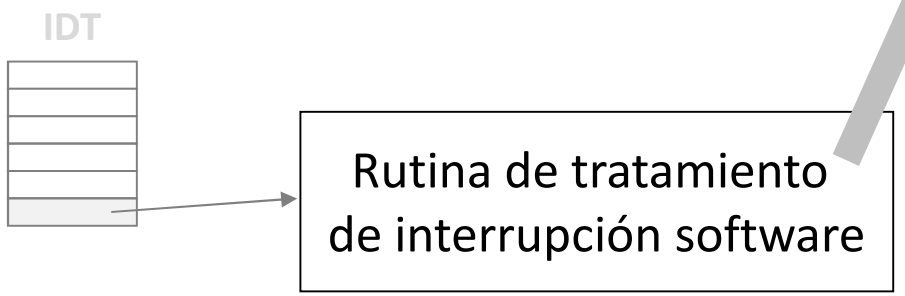

```
void Int_hardware_teclado ( idDispositivo )
{
        • idDispositivo -> HardwareID
        • Tecla = leerPuerto(HardwareID)
        • Insertar(Tecla, DatosTeclado.Buffer)
        • insertarTareaPend(&listaTareasPend,
                                    Int_software_teclado);
        • activar_int_SW();
}
```

Interrupción software
tratamiento (2/2)

```
/*  Tratamiento de interrupciones software */
void interrupcionSoftware ()
{
    void (*funcion)(void *);
    void *datos = NULL;

    Mientras ( hayTareasPend(ListaTareasPend) )
    {
        extraerPrimeraTareaPend(&(listaTareasPend), &(funcion), &(datos));
        funcion(datos);
    }
}
```

Modo Usuario

Modo Kernel



Interrupción con la mínima prioridad: se ejecutará cuando no haya nada más urgente (crítico)

Interrupción software
tipos de tratamiento en Linux

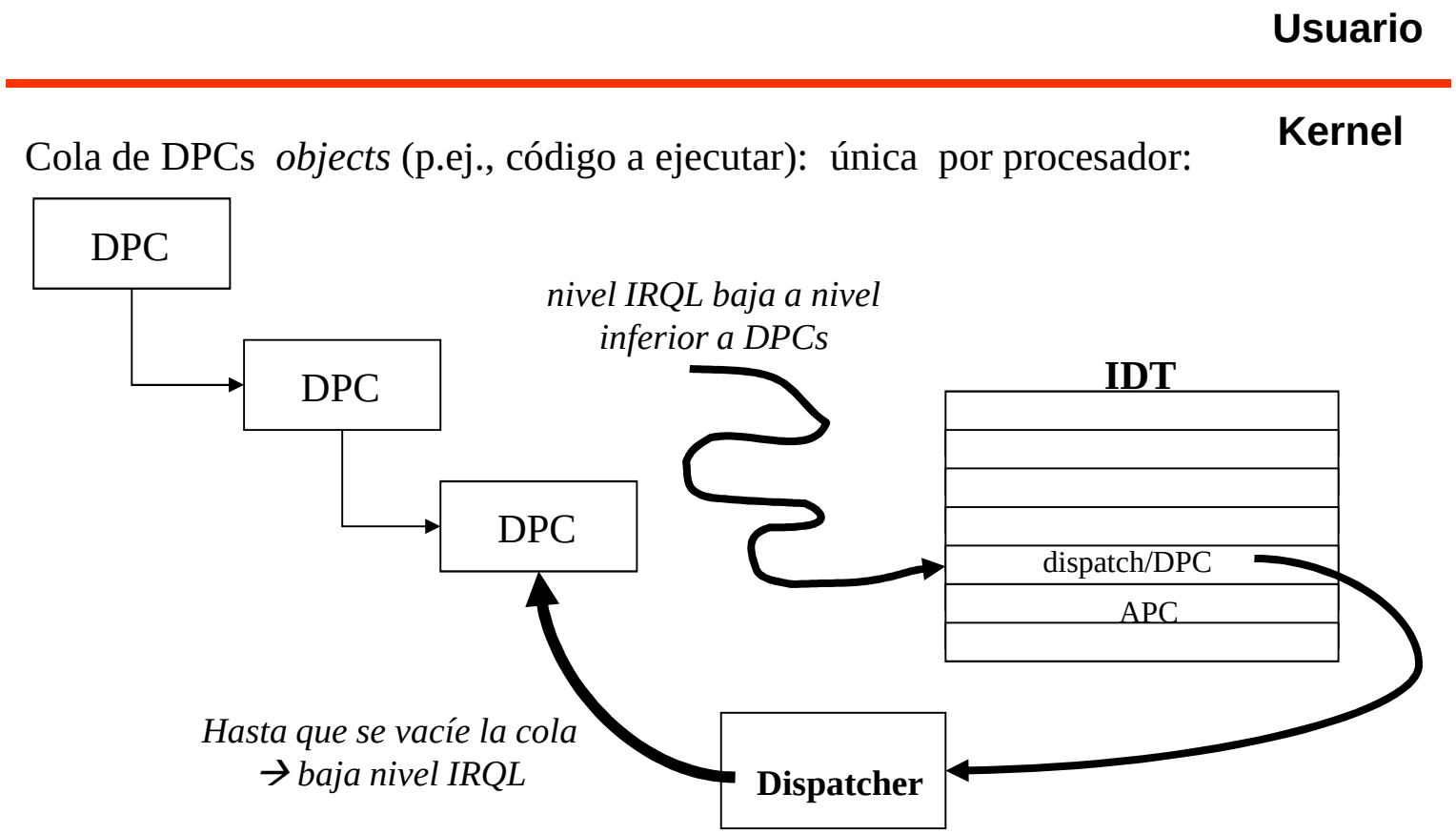
- Bottom-Halves (BH):
  - Es la 1° implementación de int. software en Linux. (eliminada en k2.6.x)
  - Se ejecutan siempre en serie (da igual # CPU).
    Solo hay 32 manejadores (registrados previamente).
- Softirqs:
  - *Softirq* del mismo tipo se pueden ejecutar en paralelo en diferentes CPU.
    Solo hay 32 manejadores (registrados previamente).
  - El *system timer* usa *softirqs*.
- Tasklets
  - Similar a *softirqs* salvo que no existe límite y más fácil de programar.
  - Todos los *tasklets* se canalizan a través de un *softirq*, de forma que un mismo *tasklet* no puede ejecutarse a la vez en varias CPU.
- Work queues
  - El *top-half* se dice que ejecuta en contexto de una interrupción => no está asociado a un proceso.
    Sin dicha asociación el código no puede dormir o bloquearse.
  - Las *Work queues* ejecutan en contexto de un proceso y tienen habilidades de un hilo de kernel.
    Tienen un conjunto de funciones útiles para creación, planificación, etc.

http://www.ibm.com/developerworks/linux/library/l-tasklets/index.html

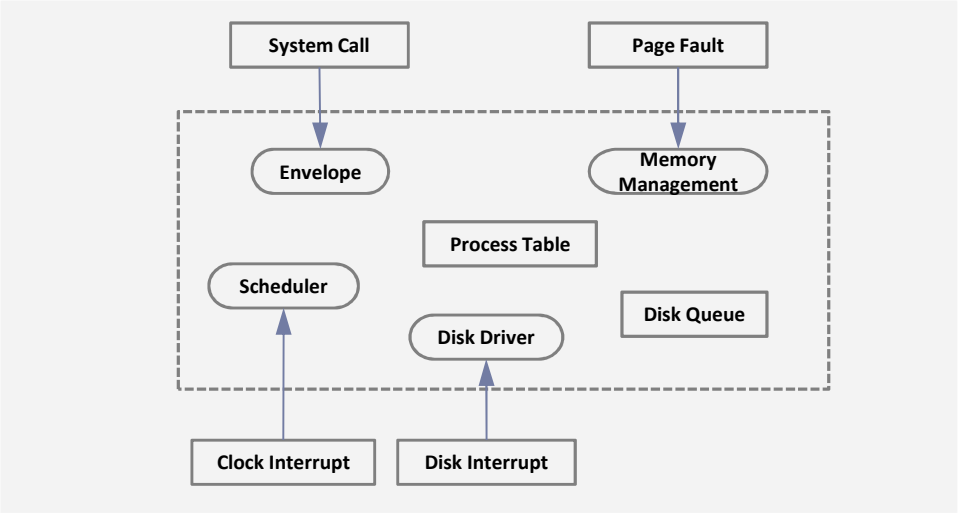
Interrupción software
tipos de tratamiento en Windows

- Deferral Procedure Calls (DPCs):
  - Comunes a todo el sistema operativo (una sola cola por CPU)
  - Realizan labores diferidas que han sido programadas:
    - Completar operaciones de E/S de los controladores.
    - Procesamiento expiración de *timers*.
    - Liberación de *threads* en espera.
    - Forzar la replanificación al expirar una rodaja de tiempo.
- Asynchronous Procedure Calls (APCs):
  - Particulares a cada thread (cada hilo tiene su propia cola).
    - El thread debe dar su permiso para que se ejecuten sus APC.
  - Pueden ejecutarse desde modo sistema o modo usuario.
    - Sistema: permite ejecutar código del sistema operativo en el contexto de un thread.
    - Usuario: utilizado por algunas API de E/S en Win32

Interrupción software
tipos de tratamiento en Windows: las DPC



Contenidos



- Introducción
- **Funcionamiento del sistema operativo**
  - Arranque del sistema
  - Características y tratamiento de los eventos
  - **Procesos de núcleo**
- Otros aspectos
  - Concurrencia en los eventos
  - Añadir nuevas funcionalidades al sistema

Contextos donde está presente el S.O.

- Arranque del sistema
- Tratamiento de eventos
  - Interrupciones hardware
  - Excepciones
  - Llamadas al sistema
  - Interrupciones software
- Procesos de núcleo
  - Realiza labores del sistema operativo que se hacen mejor en el contexto de un proceso independiente
    - Ej.: pueden realizar operaciones de bloqueo
  - Compiten con el resto de procesos por la CPU
    - El planificador suele otorgarles una prioridad mayor

Distintos tipos de procesos

- Procesos de usuario
  - Con los privilegios de un usuario no administrador
  - Solo ejecuta en modo privilegiado si:
    - Procesa una llamada al sistema que ha invocado (fork, exit, etc.)
    - Trata una excepción que ha generado (0/0, *(p=null), etc.)
    - Trata una interrupción que se ha producido mientras ejecutaba (TCPpk, …)
- Procesos de sistema
  - Con privilegios de un usuario administrador
  - Ejecuta en modo privilegiado igual que un proceso de usuario
- Procesos de núcleo
  - Pertenecen al kernel (no a un usuario)
  - Siempre se ejecutan en modo privilegiado

Contenidos

- Introducción
- **Funcionamiento del sistema operativo**
  - Arranque del sistema
  - Características y tratamiento de los eventos
  - Procesos de núcleo
- Otros aspectos
  - **Concurrencia en los eventos**
  - Añadir nuevas funcionalidades al sistema

Concurrencia en multiprocesadores

- **UP**: Uni-Processing.
  - El sistema operativo y aplicaciones se ejecuta solo en una CPU.
  - Sencillo pero mal rendimiento.
- **ASMP**: Asymmetric MultiProcessing.
  - El sistema operativo se ejecuta en la misma CPU.
  - Sencillo pero rendimiento mejorable.
- **SMP**: Symmetric MultiProcessing.
  - El sistema operativo se puede ejecutar en cualquier procesador.
  - Dificultad al necesidad mecanismos de sincronización especiales para la protección de secciones críticas.
    - Ej.: subir el nivel de interrupción no impide ejecutar sección en otra CPU.

Ejemplo de mecanismos básicos…

Linux

Técnica	Ámbito	Ejemplo de esqueleto

Deshabilitar interrupciones	• Una CPU solo	`unsigned long flags;`
		`local_irq_save(flags);`
		`/* ... SC: sección crítica ... */`
		`local_irq_restore(flags);`

Spin Locks	• Todas las CPU
	• Espera activa:
	  • NO se puede dormir, planificar, etc. en SC
		`#include <linux/spinlock.h>`
		`spinlock_t l1 = SPIN_LOCK_UNLOCKED;`
		`spin_lock(&l1);`
		`/* ... SC: sección crítica ... */`
		`spin_unlock(&l1);`

Mutex	• Todas las CPU
	• Espera bloqueante:
	  • NO usar en int. HW
		`#include <linux/mutex.h>`
		`static DEFINE_MUTEX(m1);`
		`mutex_lock(&m1);`
		`/* ... SC: sección crítica ... */`
		`mutex_unlock(&m1);`

Operaciones Atómicas	• Todas las CPU	`atomic_t a1 = ATOMIC_INIT(0);`
		`atomic_inc(&a1);`
		`printk("%d\n", atomic_read(&a1));`

Ejemplo de mecanismos compuestos…

Linux

Técnica	Ámbito	Ejemplo de esqueleto

RW locks	• Todas las CPU
	• Espera activa:
	  • NO se puede dormir, planificar, etc. en SC
		`rwlock_t x1 = RW_LOCK_UNLOCKED;`
		`read_lock(&x1);`
		`/* ... SC: sección crítica ... */`
		`read_unlock(&x1);`
		`write_lock(&x1);`
		`/* ... SC: sección crítica ... */`
		`write_unlock(&x1);`

Spin Locks + irq	• Todas las CPU
	• Espera activa y no interrup.:
	  • NO se puede dormir, planificar, etc. en SC
		`spinlock_t l1 = SPIN_LOCK_UNLOCKED;`
		`unsigned long flags;`
		`spin_lock_irqsave(&l1, flags);`
		`/* ... SC: sección crítica ... */`
		`spin_unlock_irqrestore(&l1, flags);`

RW locks + irq	• Todas las CPU
	• Espera activa y no interrup.:
	  • NO se puede dormir, planificar, etc. en SC
		`read_lock_irqsave();`
		`read_lock_irqrestore();`
		
		`write_lock_irqsave();`
		`write_lock_irqrestore();`

Ejecución anidada de tratamiento de evento

Evento en ejecución	Evento que llega	Tratamiento habitual
Int. Hw. / Excepción	Int. Hw. / Excepción	• Se permite todas, ninguna o solo de más prioridad (si S.C., deshabilitadas).
Ll. sist. / Int. Sw.	Int. Hw. / Excepción	• Interrumpe siempre (si S.C., deshabilitadas).
Int. Hw. / Excepción	Ll. sist. / Int. Sw.	• No pueden interrumpirlas.

Ll. sist. / Int. Sw.	Ll. sist. / Int. Sw.	• Kernel no expulsivo
		  • No pueden interrumpir (se encolan).
		  • Muchos UNIX y Linux antes.
		• Kernel expulsivo.
		  • Hay que proteger secciones críticas.
		  • Solaris, Windows 2000, etc.

Ejecución anidada de tratamiento de evento

Linux

Kernel Control Path	Protección en UP	Protección en *MP
Excepciones	Mutex	-
Int. HW.	Deshabilitar Int.	Spin Lock
Int. SW.	-	Spin Lock (SoftIrq, N Tasklets)
Excepciones + Int. HW.	Deshabilitar Int.	Spin Lock
Excepciones + Int. SW.	Encolar Int. SW.	Spin Lock
Int. HW. + Int. SW.	Deshabilitar Int.	Spin Lock
Exc. + Int HW. + Int. SW.	Deshabilitar Int.	Spin Lock

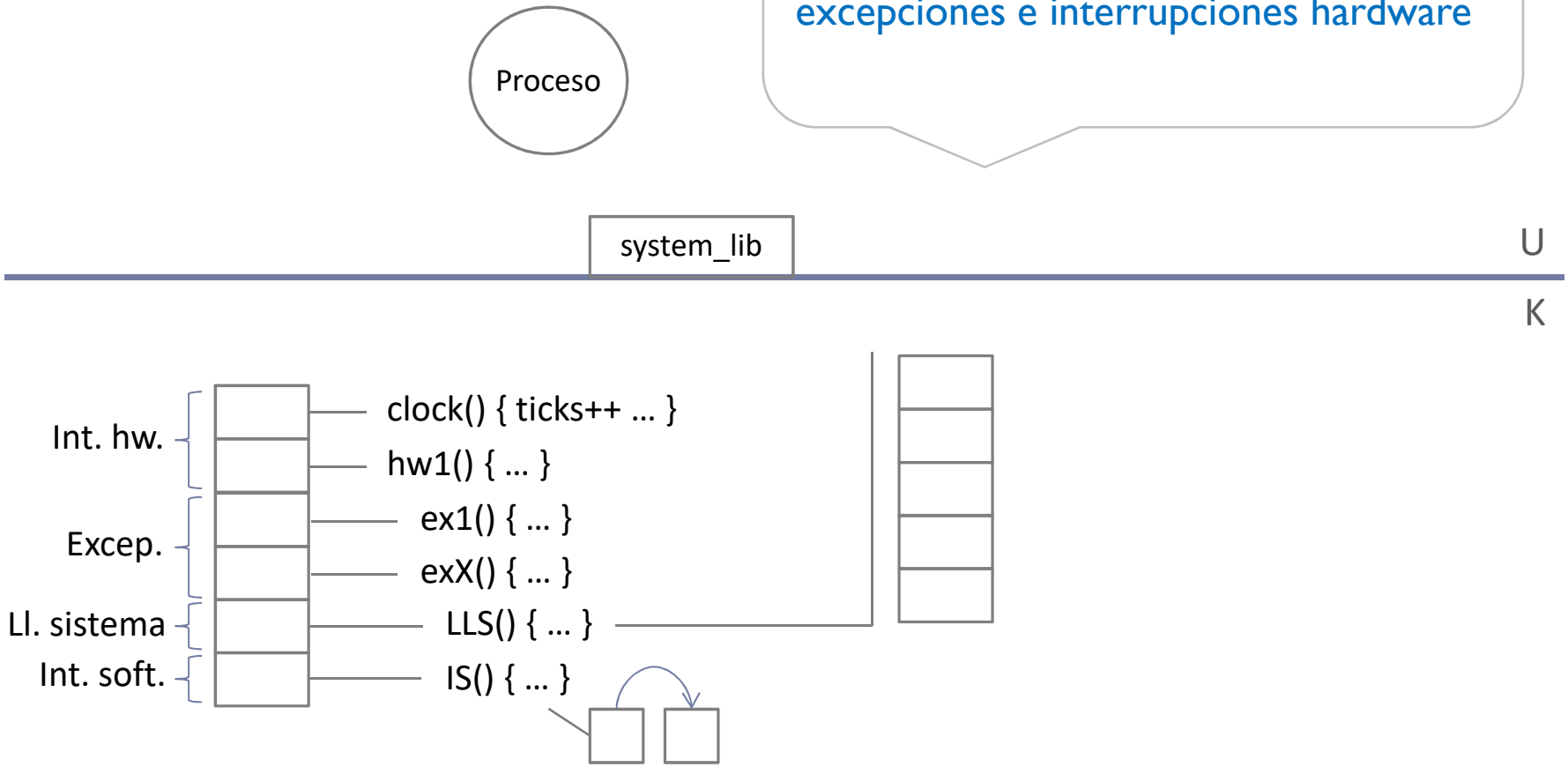
Understanding the Linux Kernel (página 218)

Contenidos

- Introducción
- **Funcionamiento del sistema operativo**
  - Arranque del sistema
  - Características y tratamiento de los eventos
  - Procesos de núcleo
- Otros aspectos
  - Concurrencia en los eventos
  - **Añadir nuevas funcionalidades al sistema**

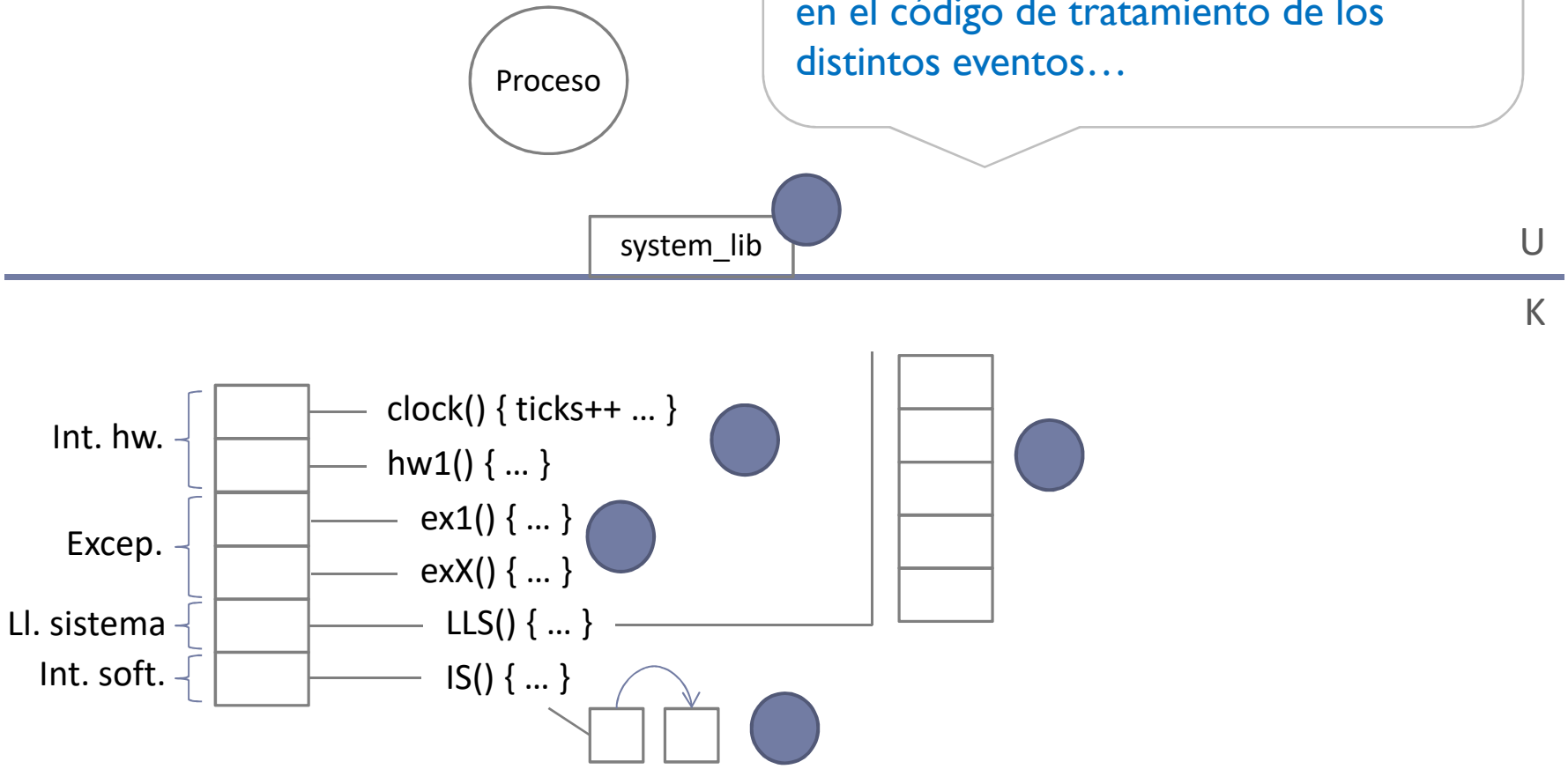
Contexto…

Funcionamiento interno del núcleo repartido entre: interrupciones software, llamadas al sistema, excepciones e interrupciones hardware



Contexto…

Una funcionalidad del sistema operativo (existente o a añadir)
está repartida en distintas localizaciones, en el código de tratamiento de los distintos eventos…



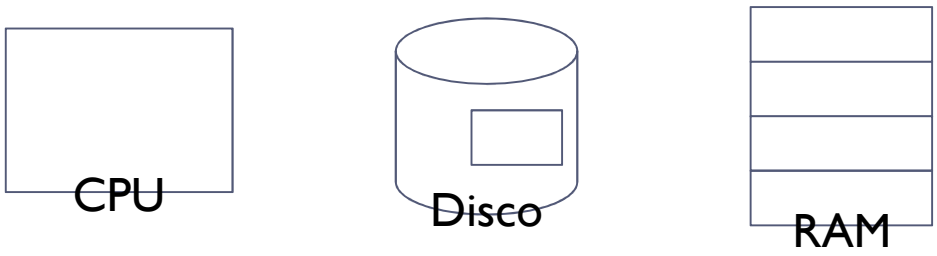
Ejemplo…

App.

- char buffer[1024];
  …
- read(fd,buffer)
- buffer[2048]='\0';

S.O.
(kernel)

HW.



Ejemplo…

App.

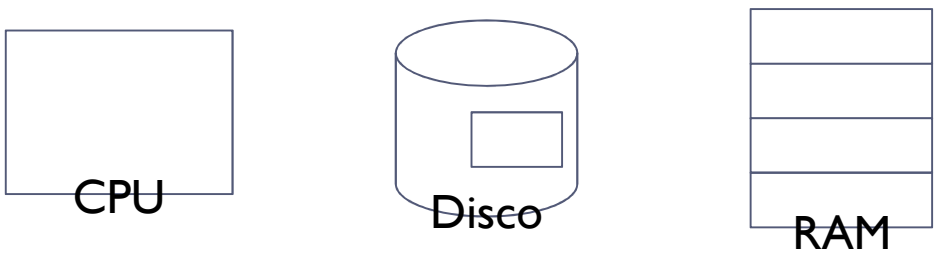
- char buffer[1024];
  …
- **read**(fd,buffer)
- buffer[2048]='\0';

ll. sistema

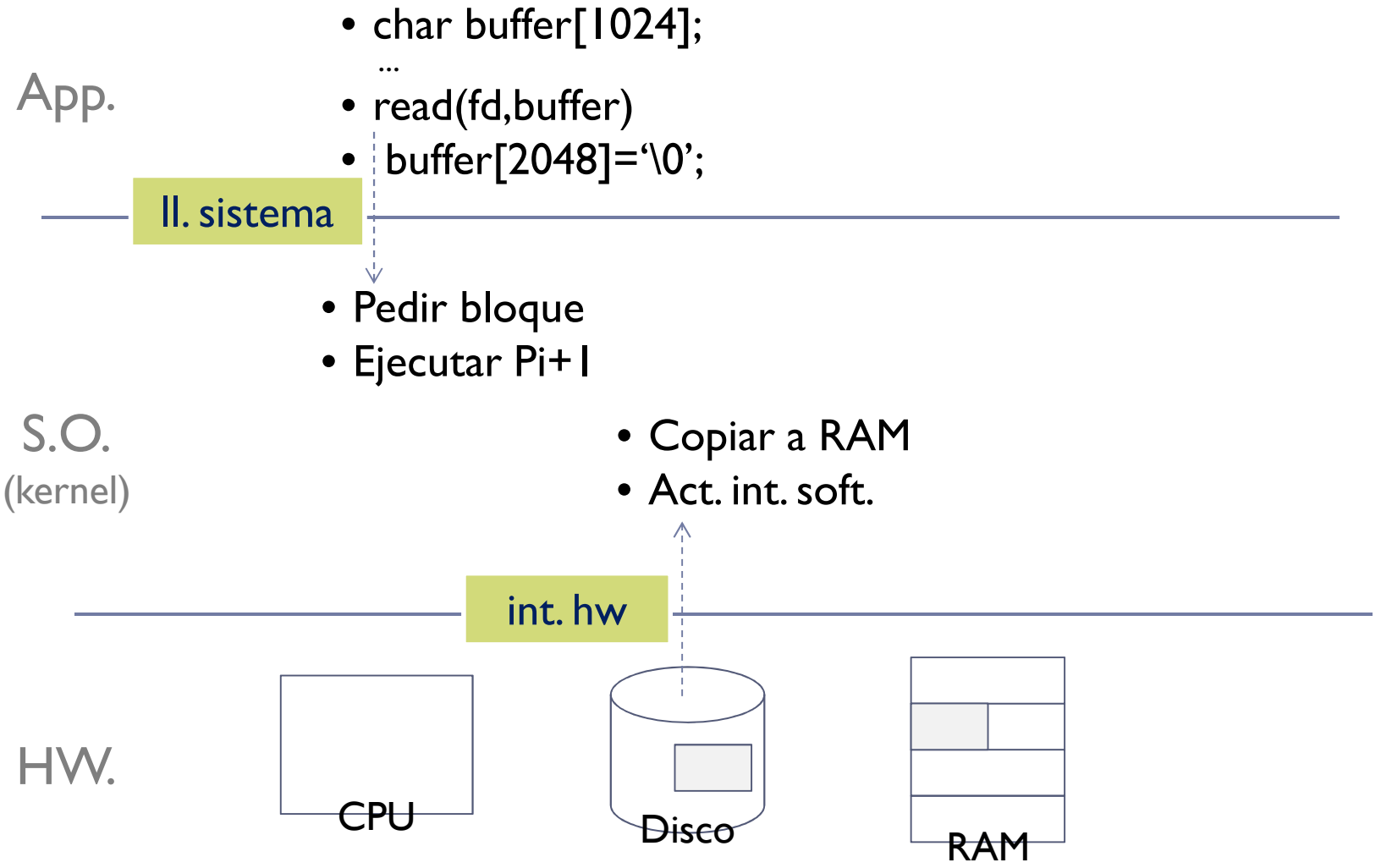
- Pedir bloque
- Ejecutar Pi+1

S.O.
(kernel)

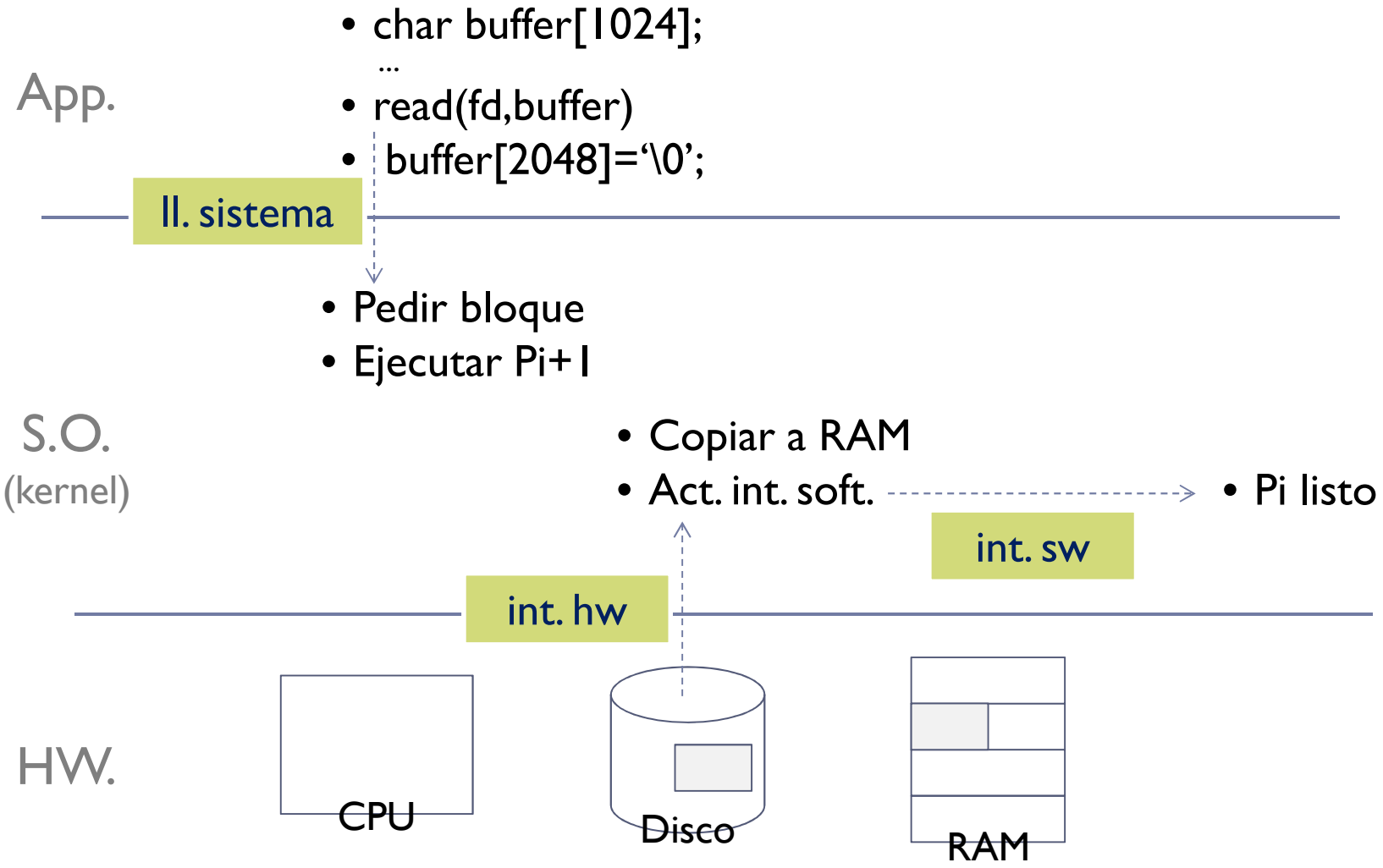
HW.



Ejemplo…

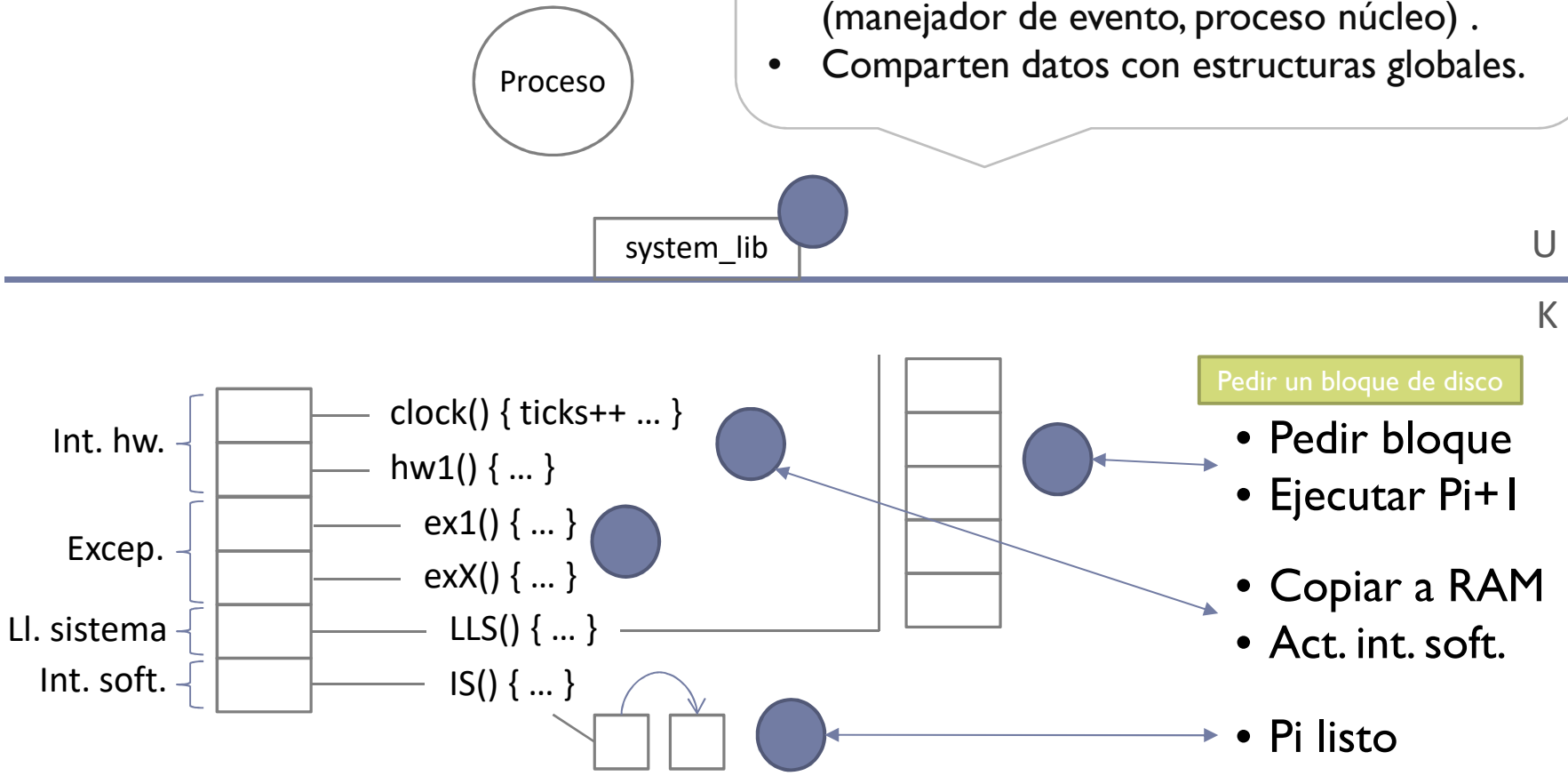

App.
• char buffer[1024];
…
• read(fd,buffer)
• buffer[2048]='\0';
ll. sistema
• Pedir bloque
• Ejecutar Pi+1
S.O.
(kernel)
• Copiar a RAM
• Act. int. soft.
int. hw
HW.
CPU
Disco
RAM

Ejemplo…


App.
• char buffer[1024];
…
• read(fd,buffer)
• buffer[2048]='\0';
ll. sistema
• Pedir bloque
• Ejecutar Pi+1
S.O.
(kernel)
• Copiar a RAM
• Act. int. soft.
• Pi listo
int. sw
int. hw
HW.
CPU
Disco
RAM

Contexto…

Una funcionalidad es una secuencia de tareas
- Pueden darse en distintos momentos, se asignan al contexto correspondiente (manejador de evento, proceso núcleo) .
- Comparten datos con estructuras globales.



Criterios para asignar una tarea a un contexto de ejecución

¿La acción está vinculada a un evento síncrono o asíncrono?

síncrono
(excepción o ll. al sistema)

asíncrono
(Int. HW e Int. SW.)

- Incluir en la rutina del evento (exc. o ll. sist.)

¿La acción es crítica?

crítica

- Incluir en la rutina de interrupción

no

¿La acción requiere bloqueos?

no

- Incluir en la interrupción software

si

- Se ejecutará dentro de un proceso de núcleo

Lección 2

Funcionamiento del sistema operativo

Grupo ARCOS

Diseño de Sistemas Operativos

Grado en Ingeniería Informática

Universidad Carlos III de Madrid