

Grupo ARCOS
Departamento de Informática
Universidad Carlos III de Madrid

Lección 2

Funcionamiento del sistema operativo

Diseño de Sistemas Operativos
Grado en Ingeniería Informática y Doble Grado I.I. y A.D.E.



Objetivos generales

1. Conocer **el rol** de un sistema operativo:
 1. Arranque del sistema
 2. Tratamiento de eventos y
 3. Procesos de núcleo.
2. Conocer **cómo es la concurrencia**.
3. Estudiar **cómo añadir nueva funcionalidad**.
 1. (síncrono, crítico, bloqueante) $\boxed{\rightarrow}$ posible lugar.

A recordar...

Antes de clase

Clase

Después de clase

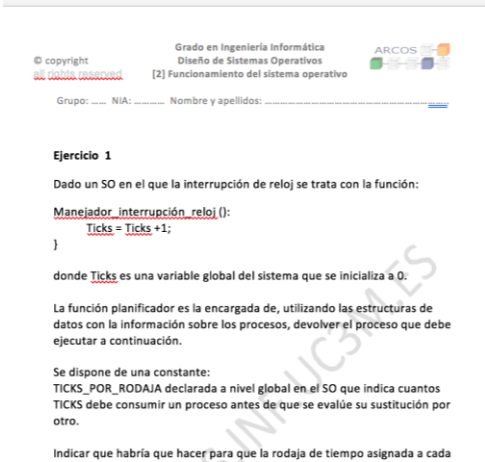
Preparar los pre-requisitos.

Estudiar el material asociado a la **bibliografía**:
las transparencias solo no son suficiente.
Preguntar dudas (especialmente tras estudio).

Ejercitar las competencias:

- ▶ Realizar todos los **ejercicios**.
- ▶ Realizar los **cuadernos de prácticas** y las **prácticas** de forma progresiva.

Ejercicios, cuadernos de prácticas y prácticas

Ejercicios ✓	Cuadernos de prácticas ✓	Prácticas ✗
 <p>Grado en Ingeniería Informática Diseño de Sistemas Operativos [2] Funcionamiento del sistema operativo</p> <p>Grupo: NIA: Nombre y apellidos:</p> <p>Ejercicio 1</p> <p>Dado un SO en el que la interrupción de reloj se trata con la función:</p> <pre>Manejador_interrupción_reloj (): Ticks = Ticks +1; }</pre> <p>donde <code>Ticks</code> es una variable global del sistema que se inicializa a 0.</p> <p>La función planificador es la encargada de, utilizando las estructuras de datos con la información sobre los procesos, devolver el proceso que debe ejecutar a continuación.</p> <p>Se dispone de una constante: TICKS_POR_RODAJA declarada a nivel global en el SO que indica cuantos TICKS debe consumir un proceso antes de que se evalúe su sustitución por otro.</p> <p>Indicar que habría que hacer para que la rodaja de tiempo asignada a cada</p>	<p>DISEÑO DE SISTEMAS OPERATIVOS</p> <p>GRADO EN INGENIERÍA INFORMÁTICA DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y ADMINISTRACIÓN DE EMPRESAS</p> <p>uc3m Universidad Carlos III de Madrid</p> <p>Añadir nuevas llamadas al sistema en Linux/Ubuntu</p>	

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

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

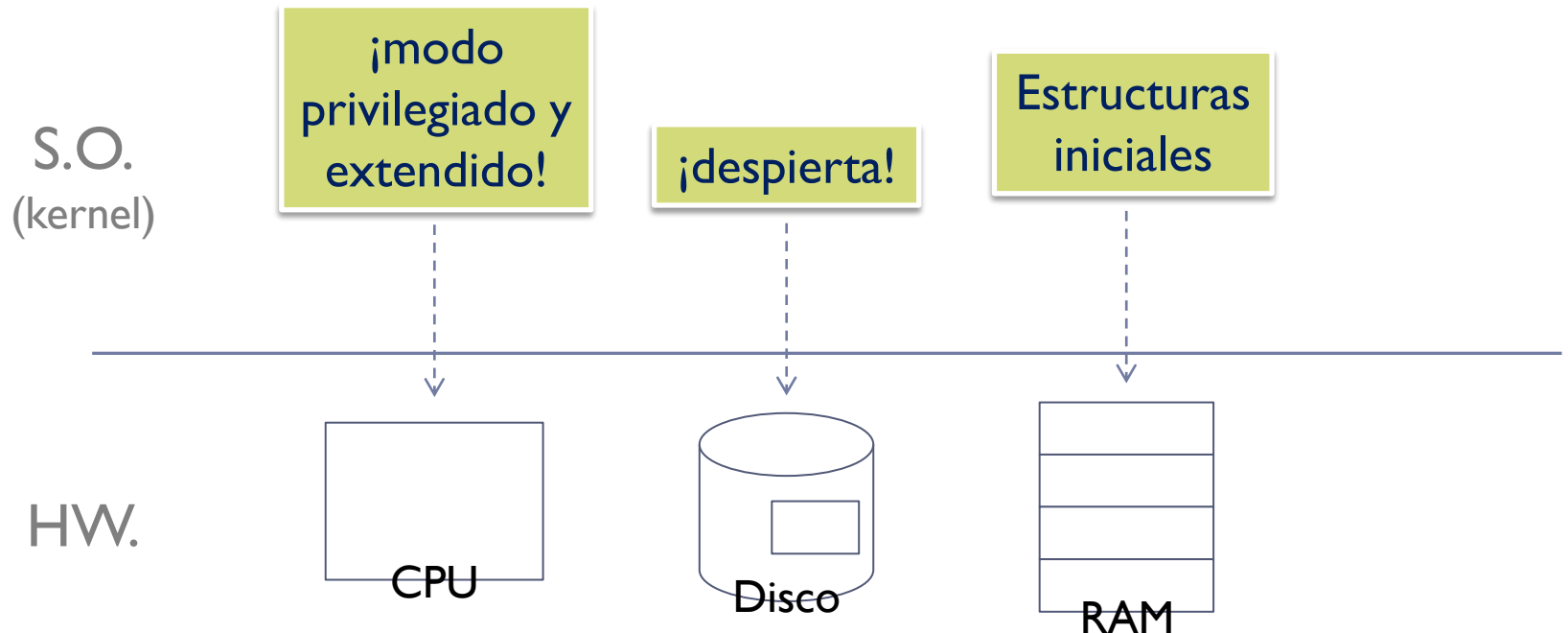
- ▶ **Introducción**
- ▶ **Funcionamiento del sistema operativo**
 - ▶ Arranque del sistema
 - ▶ Características y tratamiento de los eventos
 - ▶ Procesos de núcleo
- ▶ **Otros aspectos**
 - ▶ Concurrencia en los eventos
 - ▶ Añadir nuevas funcionalidades al sistema

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

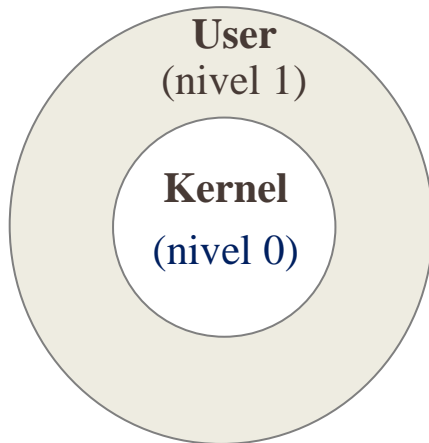


Modo kernel y usuario

repaso



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



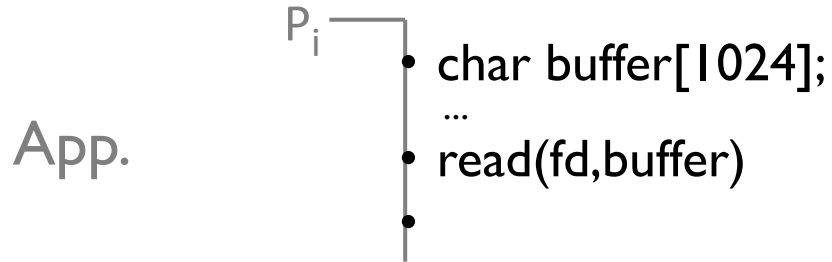
- ▶ Modo privilegiado (**kernel mode**)
 - ▶ Accede a todo el espacio de memoria
 - ▶ Uso de todo tipo de instrucciones de CPU
- ▶ Modo ordinario (**user mode**)
 - ▶ Accede solo al espacio de memoria del proceso asociado
 - ▶ No puede acceder a ciertos registros de la CPU o usar ciertas instrucciones

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

▶ Tratamiento de eventos

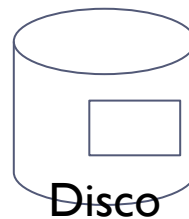
- ▶ Finalizado el arranque, el sistema operativo es una entidad pasiva.
 - ▶ Los procesos y el hardware son entidades activas (usan el kernel)
 - ▶ Excepto al inicio, siempre hay un proceso ejecutando (*idle*)
- ▶ Acceso a los servicios del S.O.
 - ▶ Interrupciones hardware
 - ▶ Interrupciones software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

Ejemplo simplificado

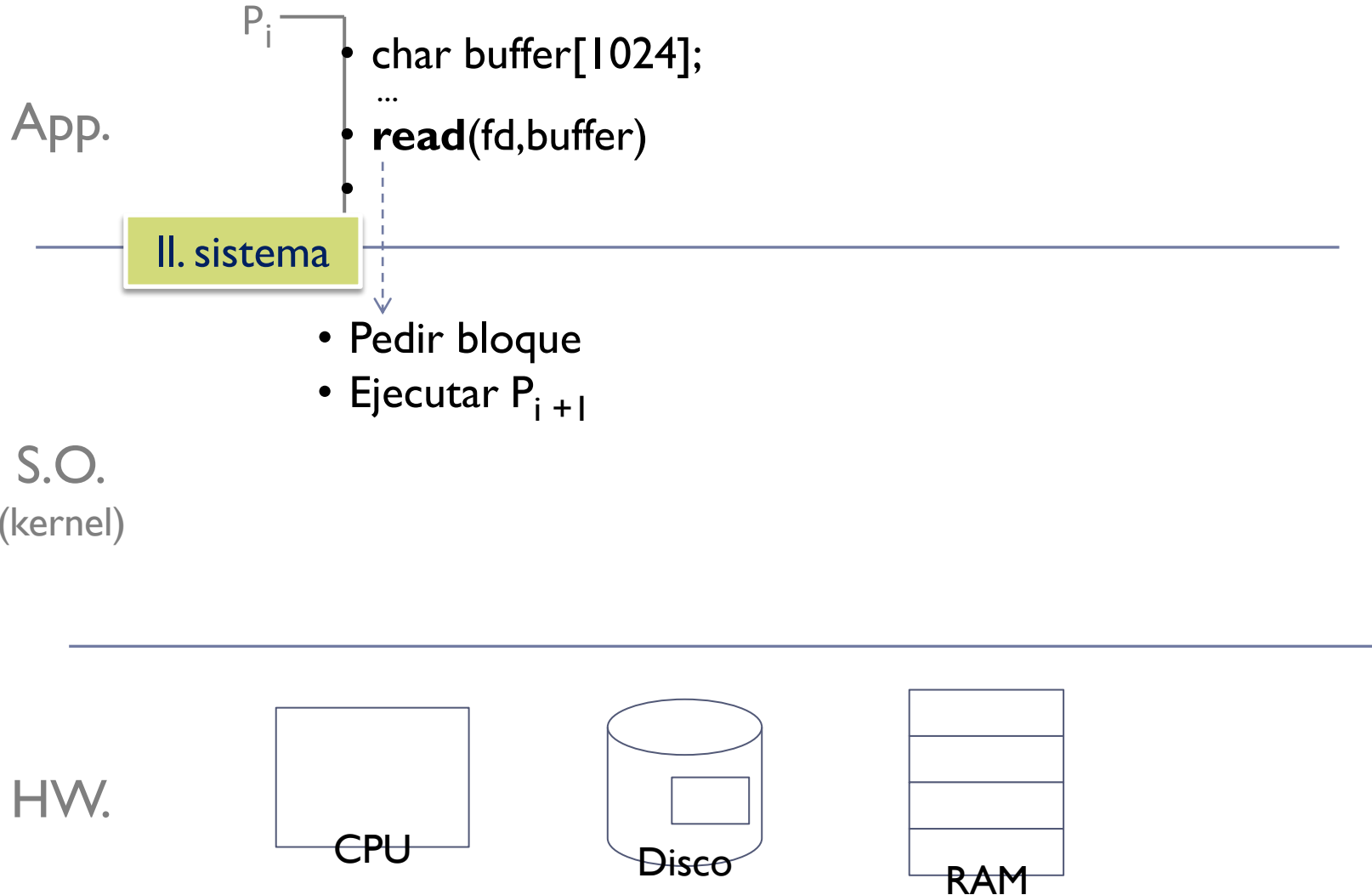


S.O.
(kernel)

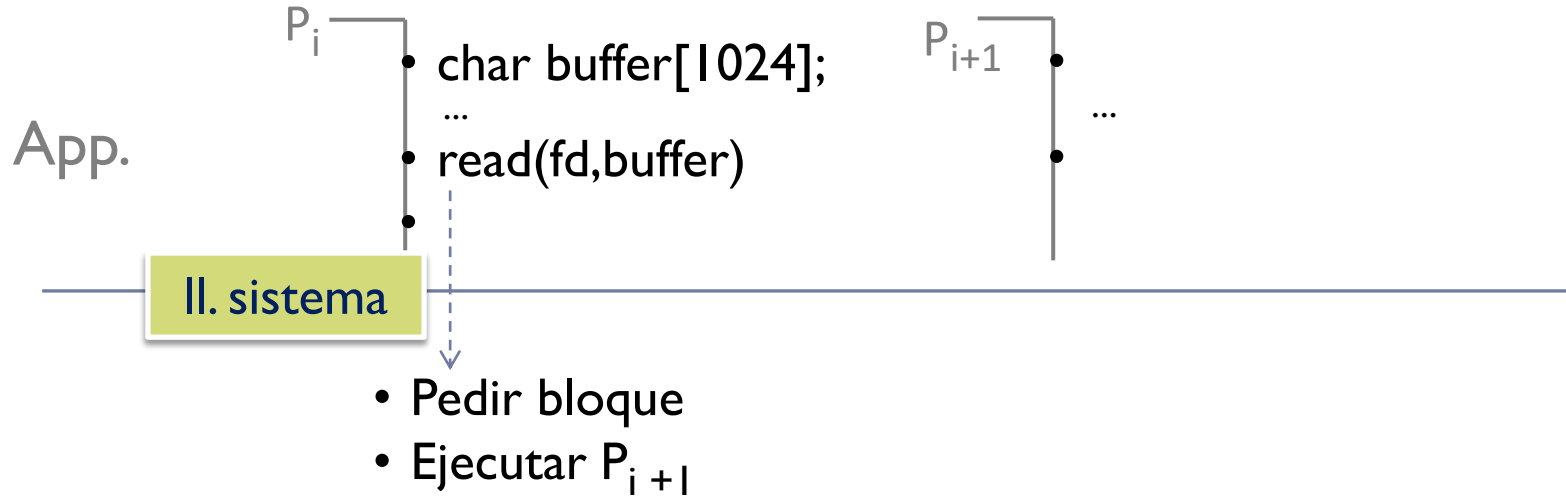
HW.



Ejemplo simplificado

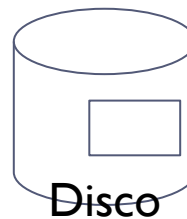
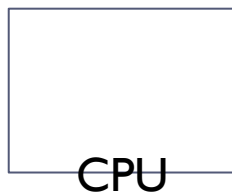


Ejemplo simplificado

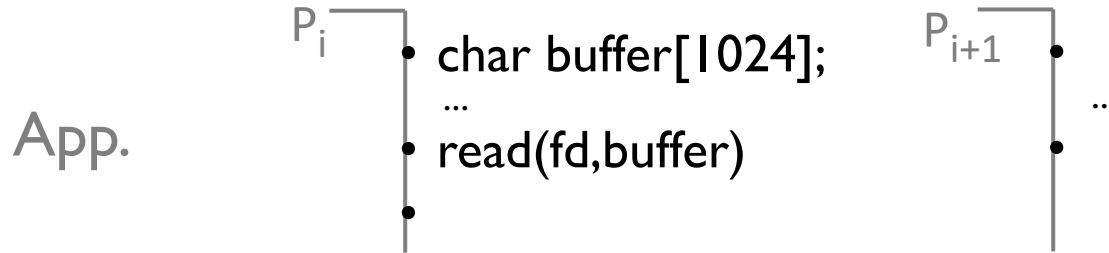


S.O.
(kernel)

HW.

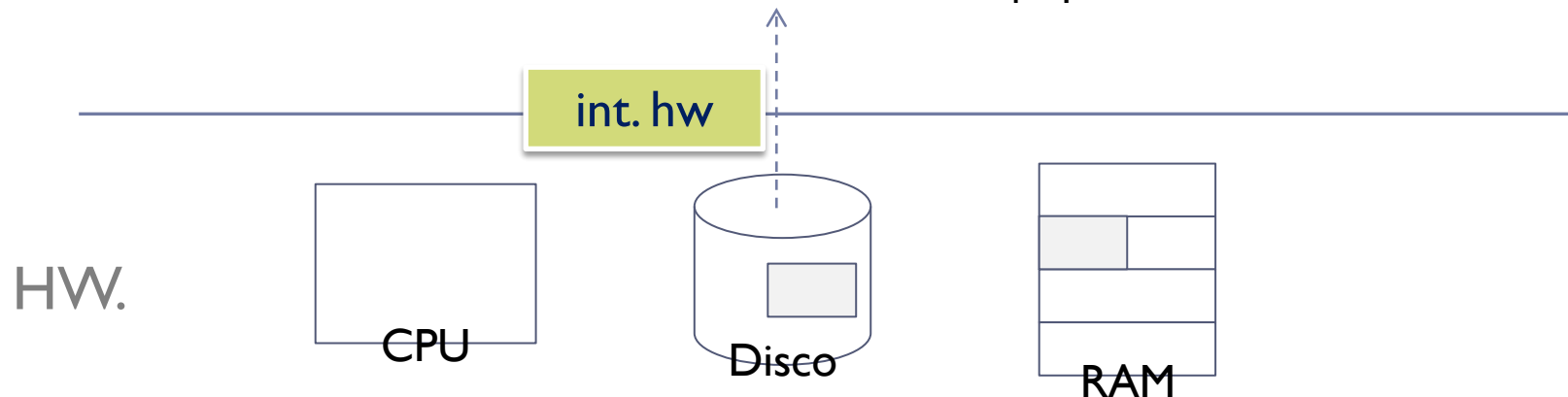


Ejemplo simplificado



S.O.
(kernel)

- Copiar a RAM
- P_i listo
- Continuar P_{i+1}



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

▶ Procesos de núcleo

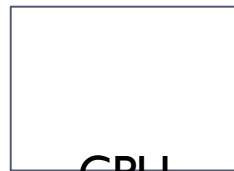
- ▶ Realiza labores del sistema operativo que se hacen mejor en el contexto de un proceso independiente
- ▶ Como **procesos prioritarios**, para tareas especiales.

Ejemplo simplificado

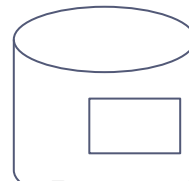
```
while (true) {  
  • sleep(1);  
  • Si (idle > 20m)  
    dormir disco  
}
```

S.O.
(kernel)

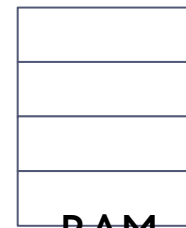
HW.



CPU



Disco



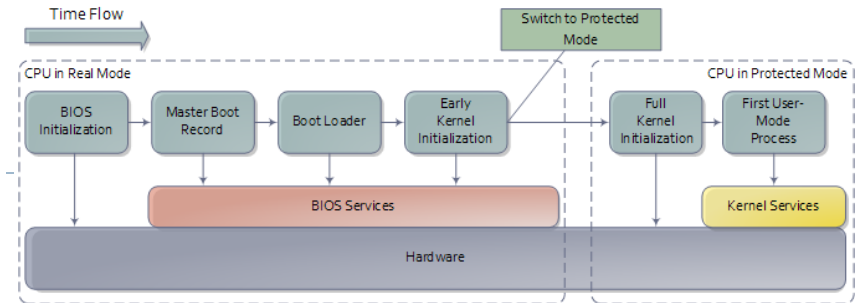
RAM

Contextos donde está presente el S.O.

resumen

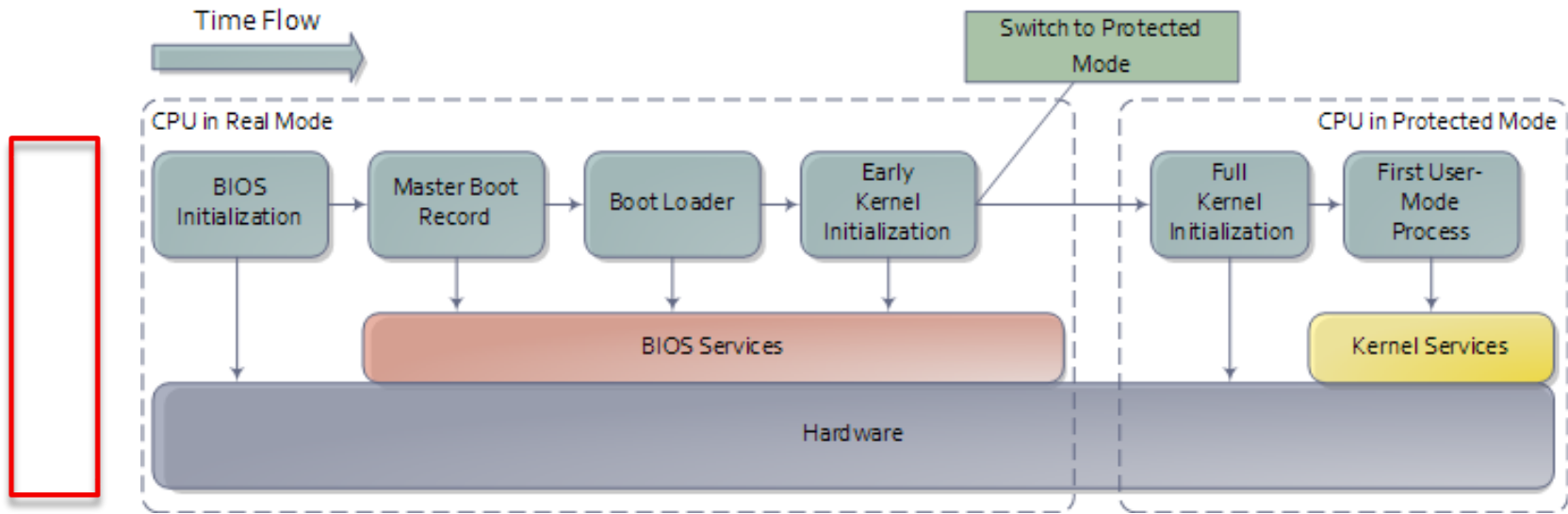
- ▶ **Arranque del sistema**
 - ▶ Realiza labores de iniciar el hardware y los procesos de núcleo, sistema y usuarios en el orden apropiados.
 - ▶ Ejecuta como **programa ejecutable**.
- ▶ **Tratamiento de eventos**
 - ▶ Finalizado el arranque, el sistema operativo es una entidad pasiva.
 - ▶ Los procesos 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

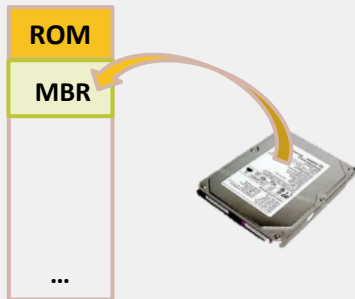
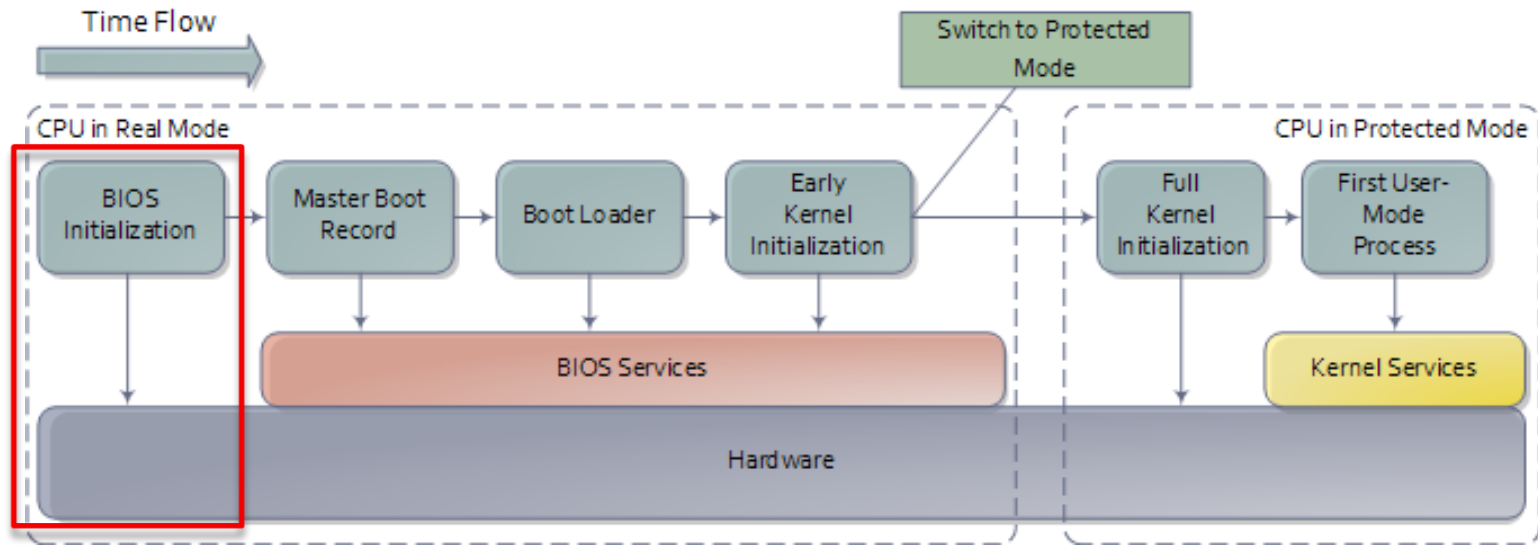
Proceso de arranque_{PC}



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

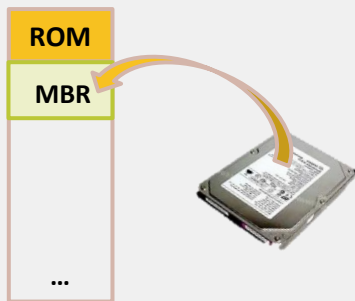
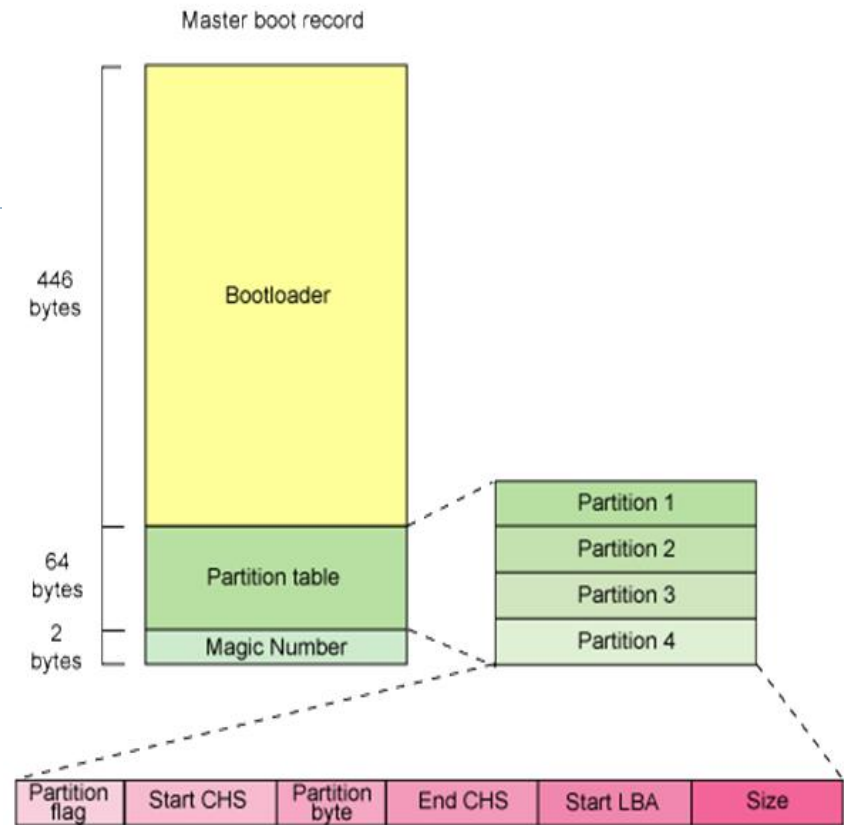


Proceso de arranque_{PC}



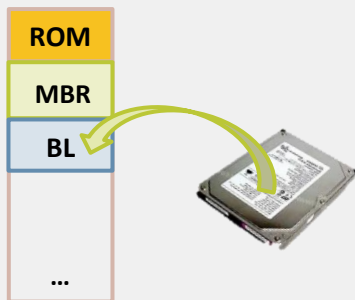
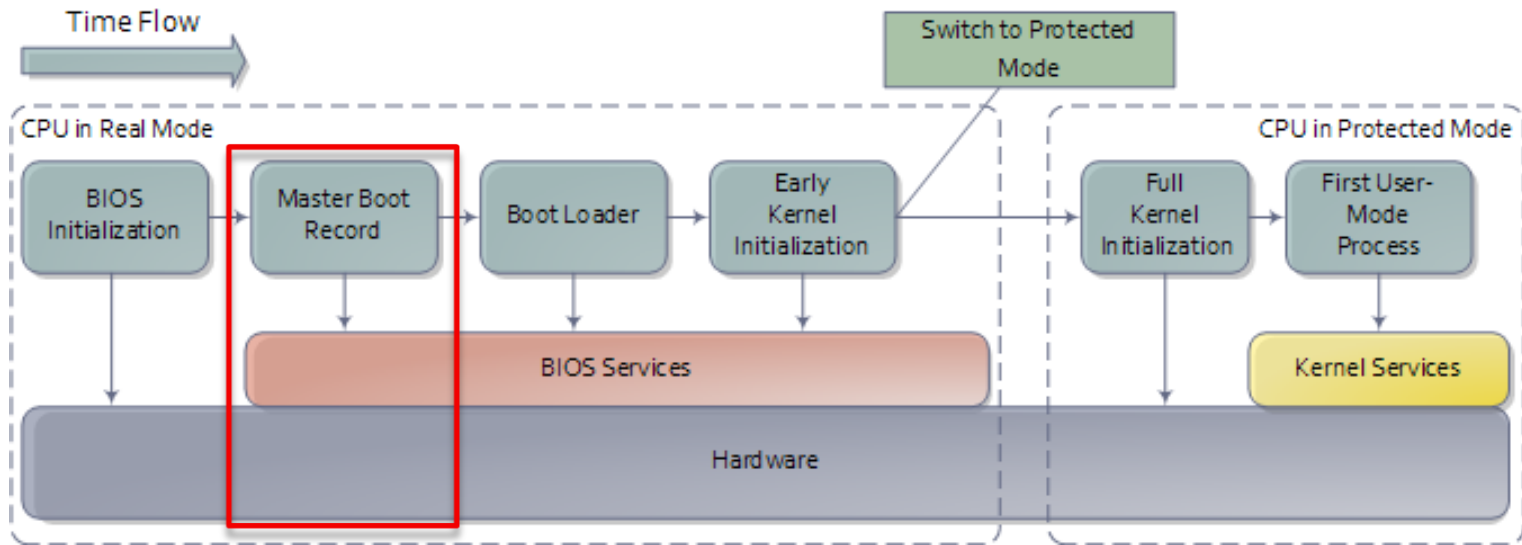
- Se ejecuta el **cargador de la ROM**
 - *Power-On SelfTest (POST)*
 - Carga en memoria (0000:7C00) el *Master Boot Record*

Proceso de arranque_{PC}



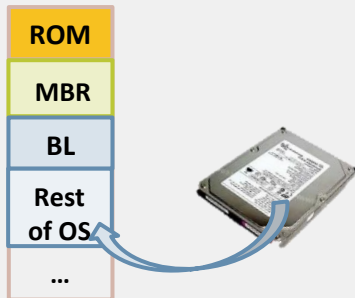
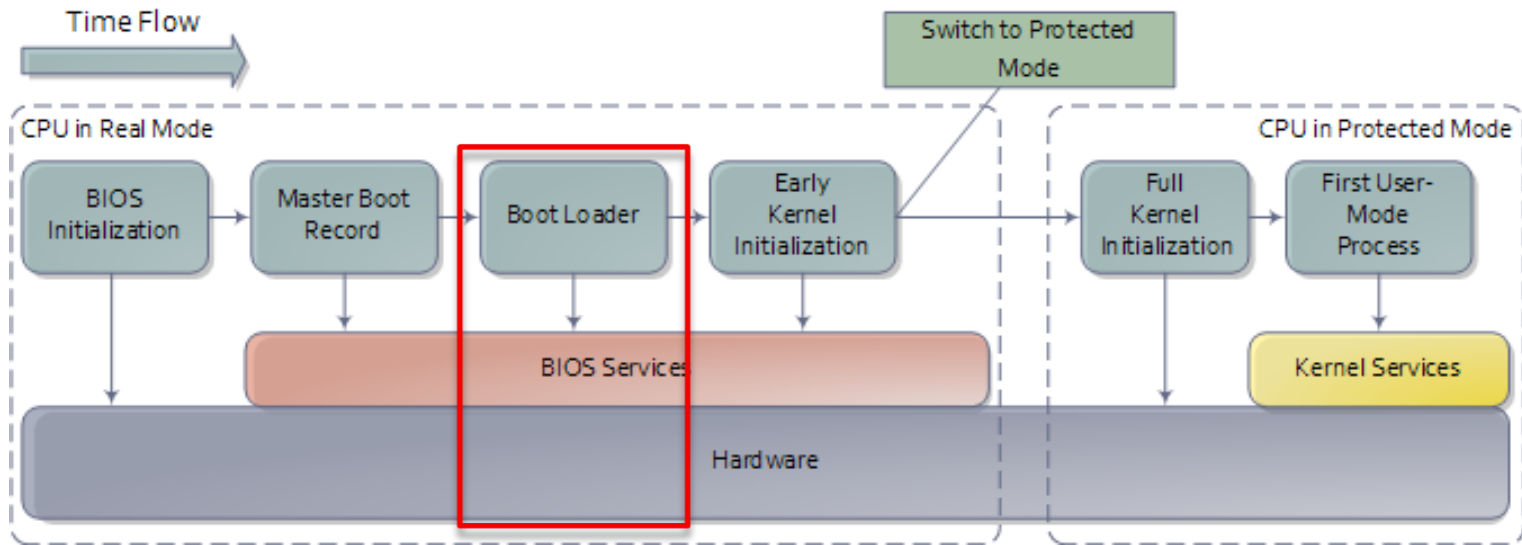
- Se ejecuta el **cargador de la ROM**
 - *Power-On SelfTest* (POST)
 - Carga en memoria (0000:7C00) el **Master Boot Record**

Proceso de arranque_{PC}



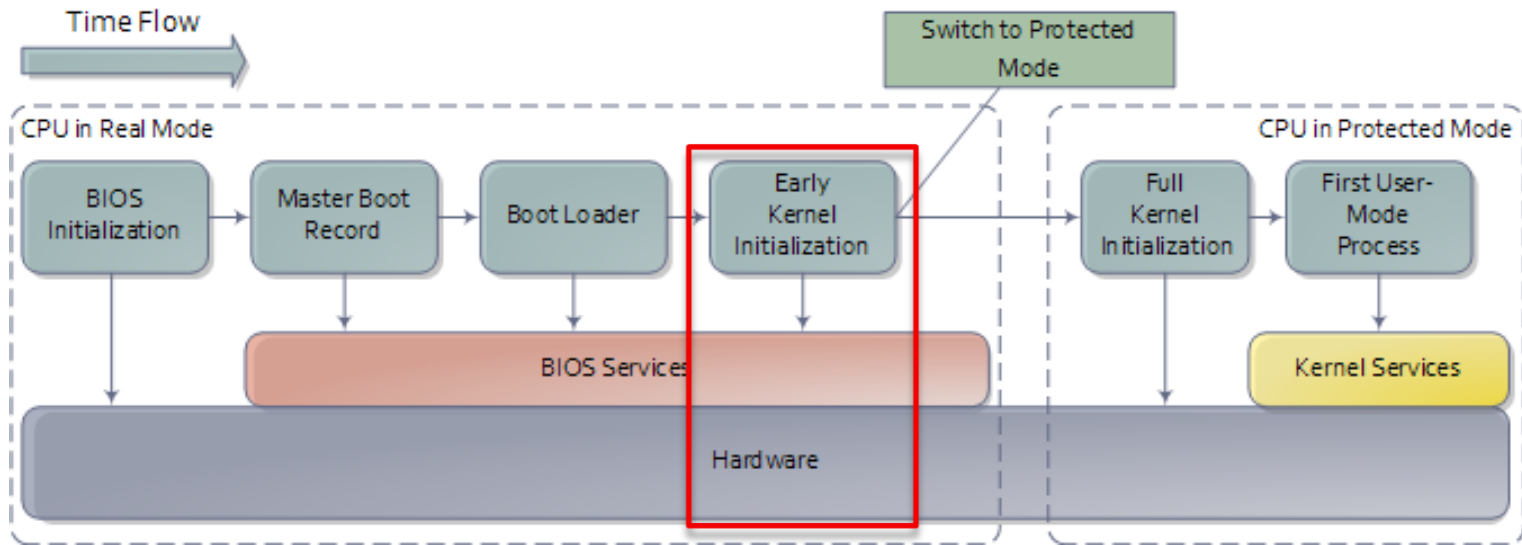
- Se ejecuta el *Master Boot Record*
 - (Es la primera parte del cargador del S.O.)
 - Busca una partición activa en la tabla de particiones
 - Carga el *Boot Record* en memoria desde esta partición

Proceso de arranque_{PC}



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

Proceso de arranque_{PC}

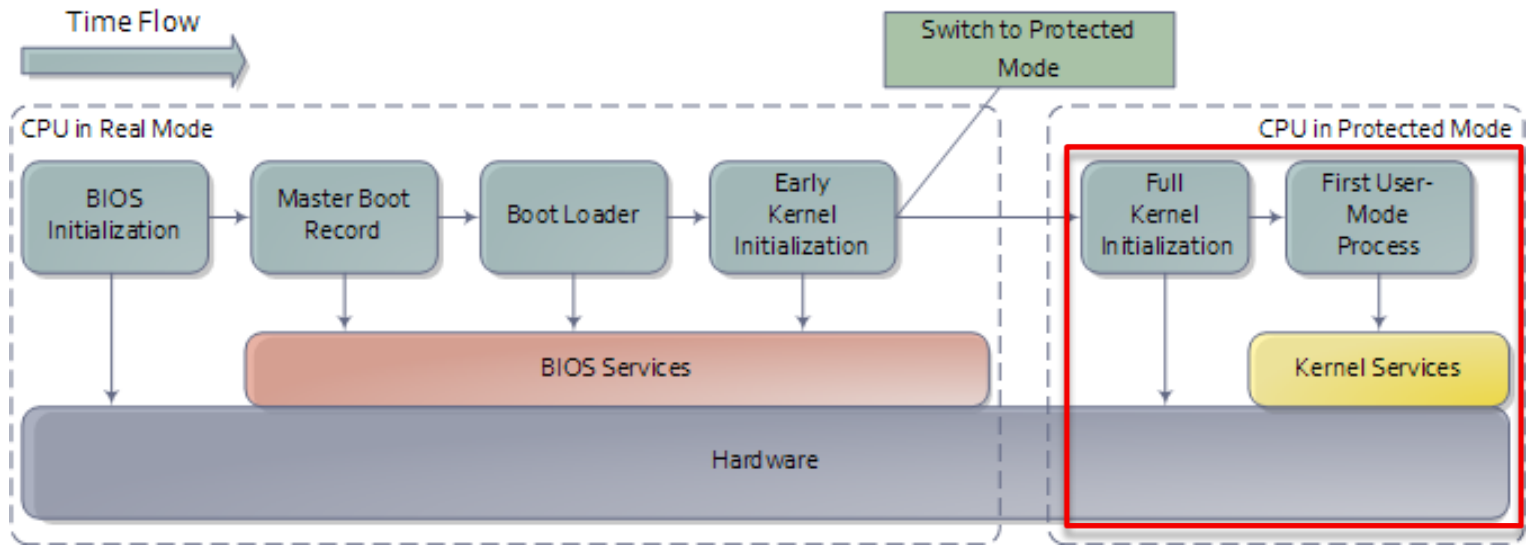


ROM
MBR
BL
Rest of OS
...



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

Proceso de arranque_{PC}

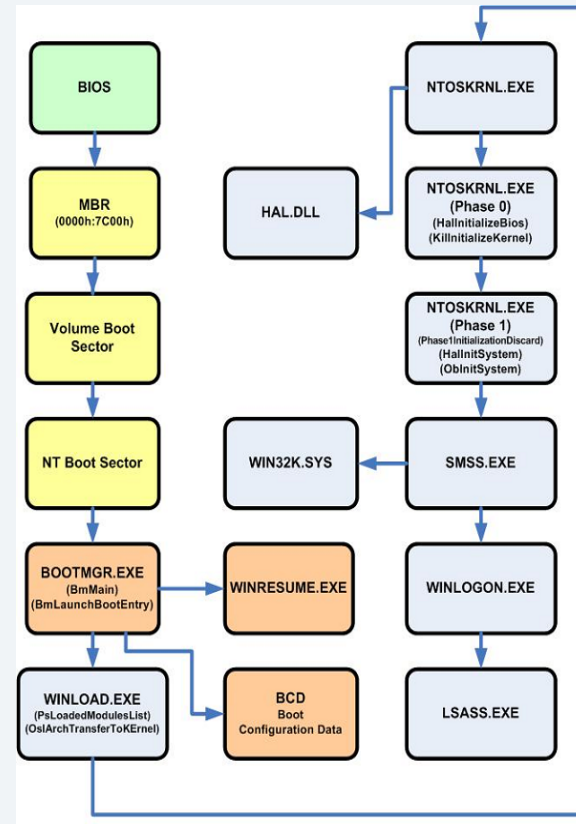
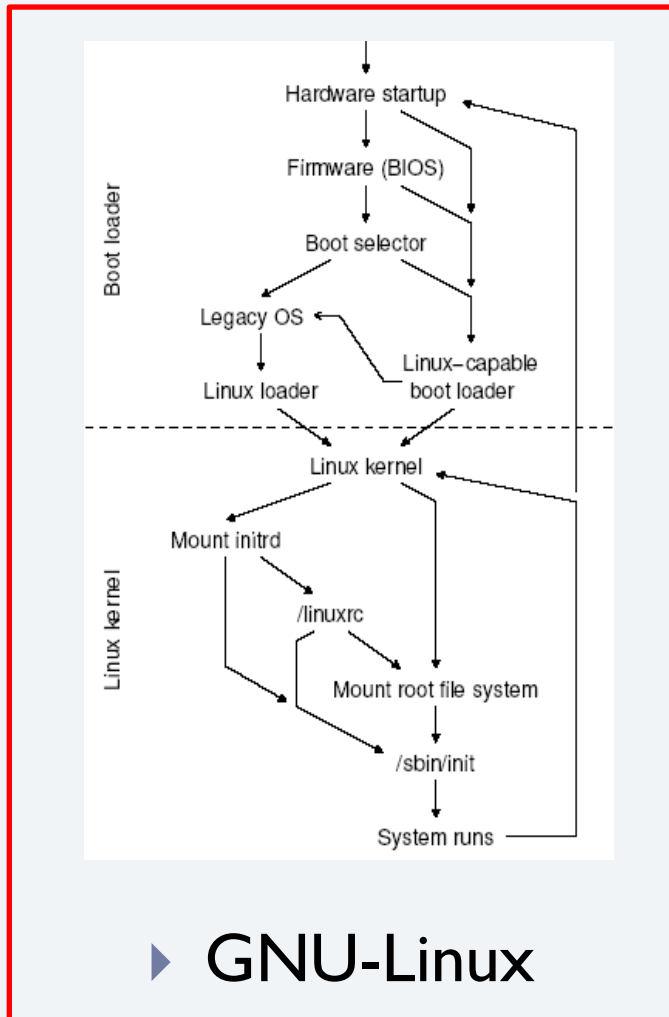


ROM
MBR
BL
Rest of OS
...

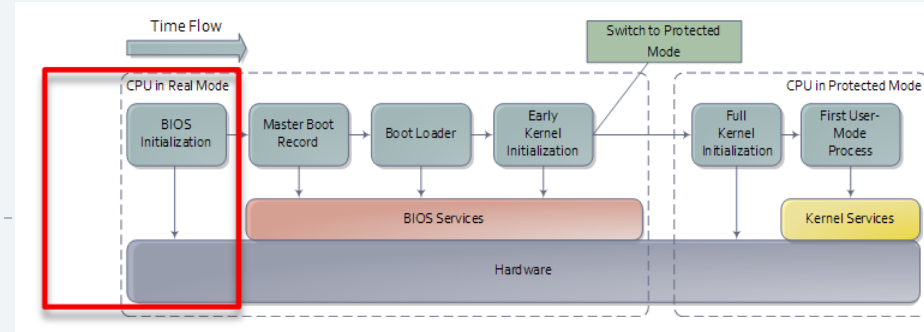


- Se ejecuta la **inicialización del kernel (2/2)**
 - Se establece el resto del S.O. en modo protegido
 - Se construye los procesos iniciales
 - Procesos de núcleo, servicios de sistema y terminales (*login*)

Ejemplo de secuencias de arranque



GNU-Linux_{PC}



```
Award Modular BIOS v6.00PG, An Energy Star Ally
Copyright (C) 1984-2007, Award Software, Inc.

Intel X38 BIOS for X38-DQ6 F4

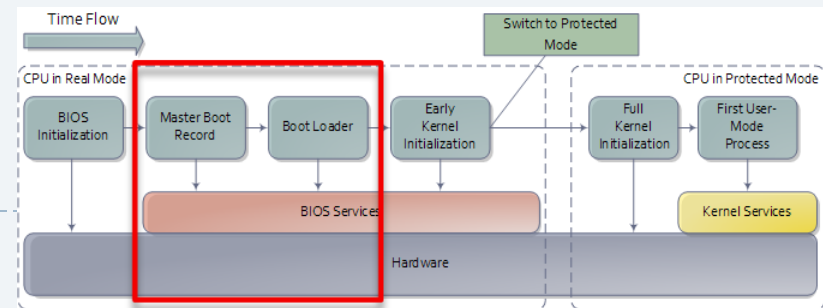
Main Processor : Intel(R) Core(TM)2 Extreme CPU X9650 @ 4.00GHz(333x12)
<CPUID:0676 Patch ID:0000>
Memory Testing : 2096064K OK

Memory Runs at Dual Channel Interleaved
IDE Channel 0 Slave : WDC WD3200AAJS-00RYA0 12.01B01
IDE Channel 1 Slave : WDC WD3200AAJS-00RYA0 12.01B01

Detecting IDE drives ...
IDE Channel 4 Master : None
IDE Channel 4 Slave : None
IDE Channel 5 Master : None
IDE Channel 5 Slave : None

<DEL>:BIOS Setup <F9>:XpressRecoveryZ <F12>:Boot Menu <End>:QFlash
09/19/2007-X38-ICH9-6A790G0QC-00
```

GNU-Linux_{PC}

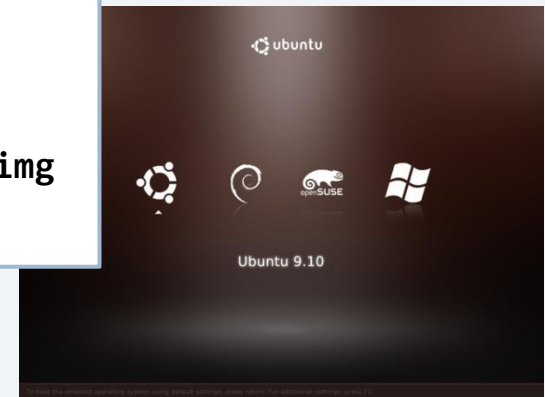


```
GNU GRUB version 0.97 (638K lower / 523200K upper memory)

Ubuntu, kernel 2.6.15-23-386
Ubuntu, kernel 2.6.15-23-386 (recovery mode)
Ubuntu, memtest86+
Other operating systems:
Microsoft Windows XP Home Edition

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the
commands before booting, or 'c' for a command-line.
The highlighted entry will be booted automatically in 9 seconds.
```

```
grub> set root=(hd0)/boot
grub> insmod linux
grub> linux /bzImage-2.6.14.2
grub> initrd /initrd-2.6.14.2.img
grub> boot
```

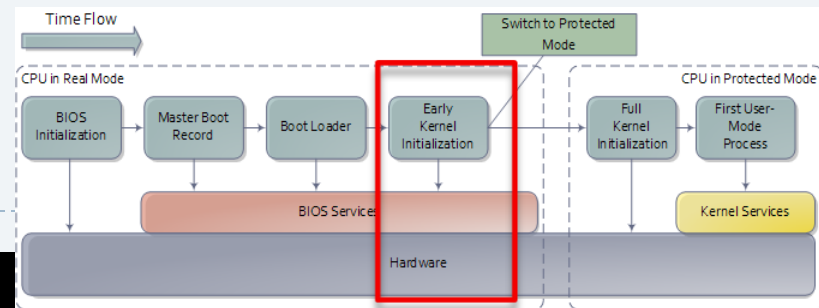


- Ejecución de **LILO** (*Linux Loader*) o **GRUB** (*Grand Unified Bootloader*).
 - Presenta un menú de opciones (/etc/grub.conf)
 - Se carga en memoria la imagen del kernel (**vmlinuz**) y se ejecuta con los parámetros/opciones que se indiquen.
 - También se puede “encadenar” otro cargador.

GNU-Linux_{PC}

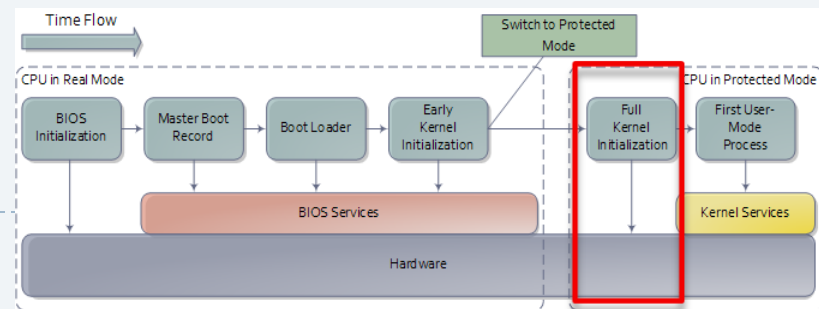


```
TURBOchannel rev. 0 at 20.0 MHz (without parity)
slot 0: DEC      PMAG-BA  V5.3a
slot 5: DEC      PMAZ-AA  V5.3a
slot 6: DEC      PMAD-AA  V5.3a
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Initializing RT netlink socket
Starting kswapd
Journalled Block Device driver loaded
devfs: v1.12c (20020818) Richard Gooch (rgooch@atnf.csiro.au)
devfs: boot_options: 0x0
PMAG-BA framebuffer in slot 0
Console: switching to colour frame buffer device 128x54
```



- Se ejecuta el kernel (**vmlinuz**): base
 - Si se necesita, descomprime el kernel
 - Este realiza el reconocimiento de hardware (y la inicialización de aquel cuyos *drivers* están compilados dentro del kernel)

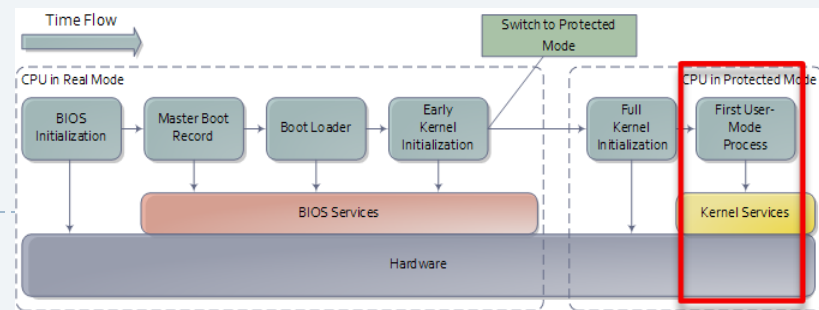
GNU-Linux_{PC}



- Se ejecuta el kernel (**initrd**): módulos
 - El **initrd** es el sistema de ficheros inicial con los drivers necesarios para continuar.
 - Se ejecuta el shell-script **/linuxrc**
 - Se inicializa los drivers con la configuración básica del sistema.
 - En el se **initrd** pasa a ‘pivotar’ al sistema raíz definitivo:
 - El mismo (sistema empotrado), participación de un disco duro, NFS, etc.

```
Initializing basic system settings ...
Updating shared libraries
Setting hostname: engpc23.murdoch.edu.au
```

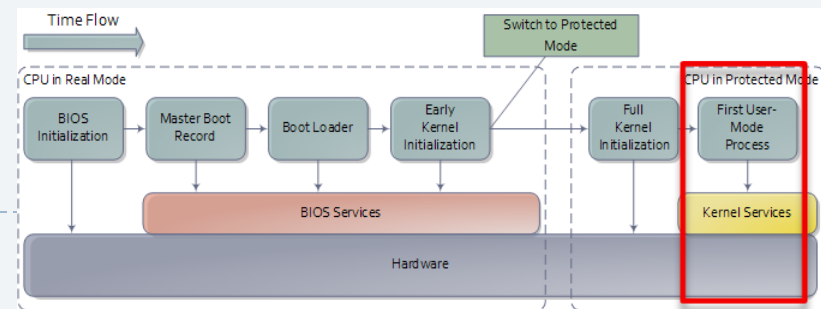
GNU-Linux_{PC}



- Se ejecuta el proceso **init**
 - El proceso **init** (pid 1) arranca todos los procesos del sistema...
 - ... y los procesos de terminal (*login* o *xlogin*) para que el usuario se autentique.
 - Pasa a quedarse dormido esperando la llegada de eventos (*cpu_idle*)

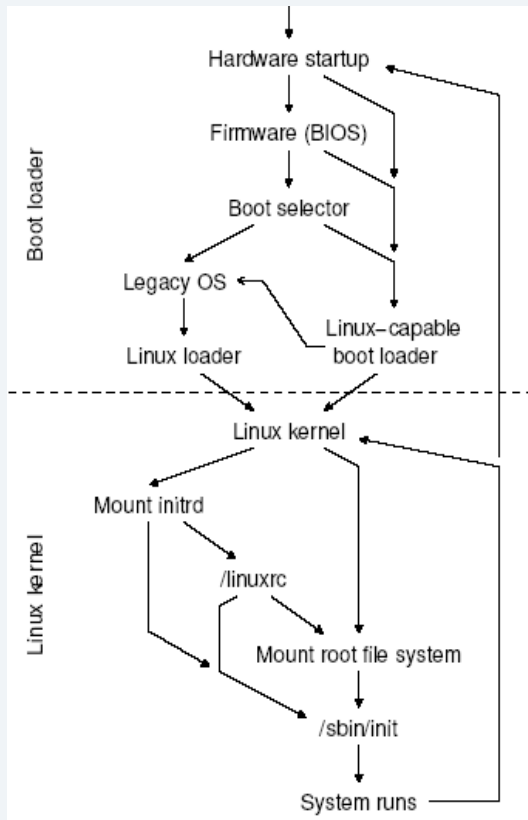
```
INIT: Entering runlevel: 4
rc.M ==> Going multiuser...
Starting system logger ... [ OK ]
Initialising advanced hardware
Setting up modules ... [ OK ]
Initialising network
Setting up localhost ... [ OK ]
Setting up inet1 ... [ OK ]
Setting up route ... [ OK ]
Setting up fancy console and GUI
Loading fc-cache ... [ OK ]
rc.vlinit ==> Going to runlevel 4
Starting services of runlevel 4
Starting dnsmasq ... [ OK ]
==> rc.X Going to multiuser GUI mode ...
XFree86 Display Manager
Framebuffer /dev/fb0 is 307200 bytes.
Grabbing 640x480 ...
```


GNU-Linux_{PC}

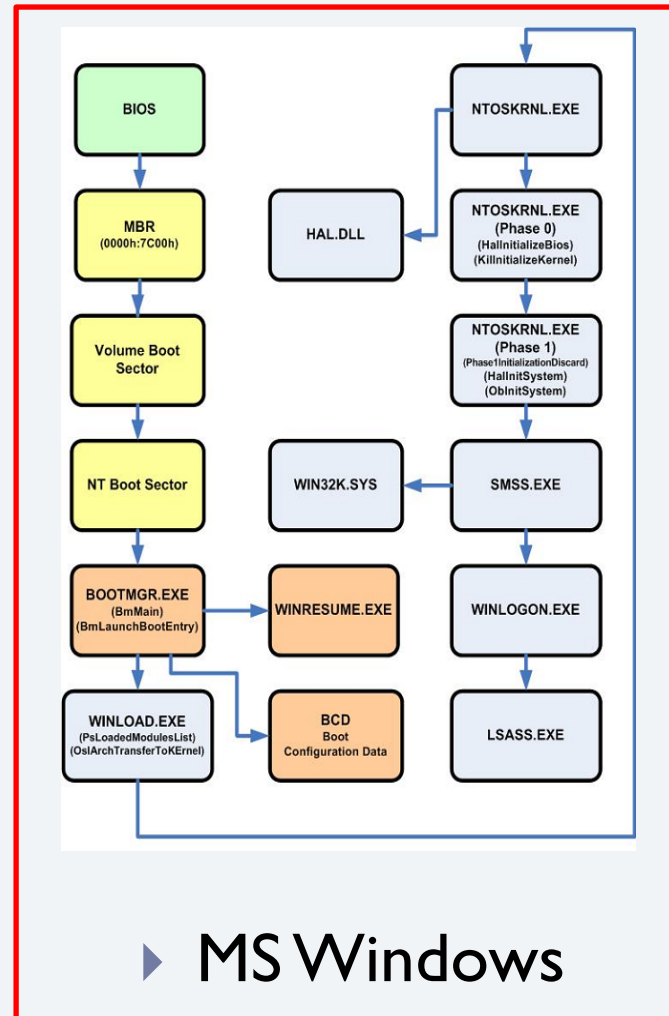


- Se ejecuta el proceso **init**
 - El proceso **init** (pid 1) arranca todos los procesos del sistema.
 - Según las instrucciones del fichero **inittab**.
 - Los procesos a arrancar están ordenados en diferentes directorios de arranque
 - Uno para el arranque inicial (**rcS.d**)
 - Varios para los tipos de arranque completo del sistema (**rc[1-5].d**)
 - Uno para apagar (**rc0.d**) y otro para reiniciar (**rc6.d**)
 - El proceso **init** permanece ejecutando todo el tiempo (huérfanos, parada, etc.)
 - Durante el arranque inicial (**rcS.d**) habitualmente:
 - Se carga los drivers restantes
 - Se comprueba el sistema de ficheros raíz (si se aplica) y remonta para lectura-escritura
 - Se monta el resto de sistemas de ficheros
 - Durante el arranque completo del sistema (**rc1.d, rc2.d, ...**)
 - Arranca los servicios y inicia el sistema de *login/xlogin*.

Ejemplo de secuencias de arranque

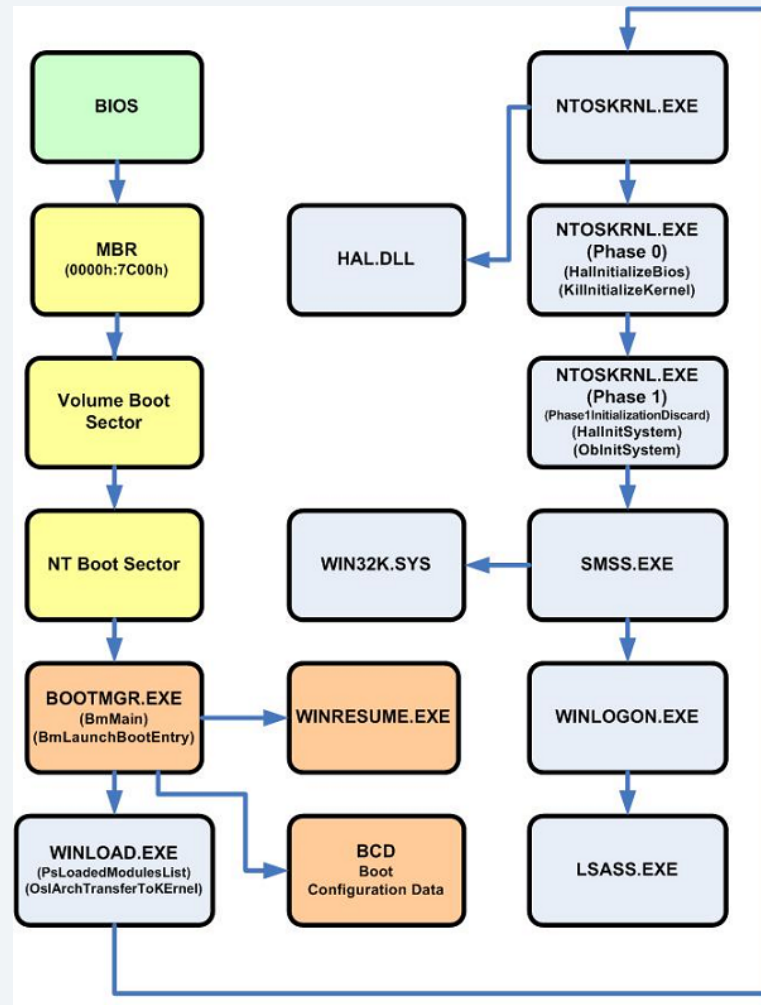


► GNU-Linux



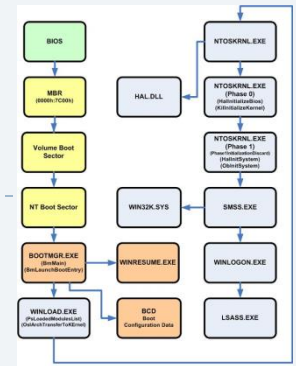
► MS Windows

Arranque Windows 7_{PC}

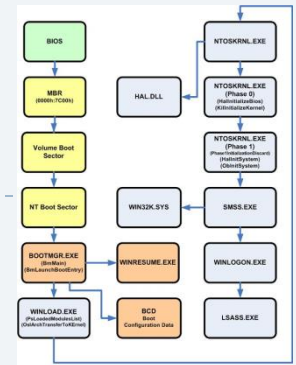


Arranque Windows 7_{PC} (1 / 2)

- La CPU comienza a ejecutar el **MBR**
 - Busca y carga en memoria el sector de arranque del volumen y el sector de arranque de NT (8 KB en tamaño, entiende FAT32 y NTFS)
- La CPU comienza a ejecutar el **sector de arranque de NT (NTBS)**
 - Busca y carga en memoria **BOOTMGR.EXE**
- La CPU comienza a ejecutar **BOOTMGR.EXE**
 - Comprueba si hay estado de hibernación. Si es así, ejecuta **WINRESUME.EXE**
 - Monta y extrae la información básica del BCD (*Boot Configuration Data*)
 - Muestra un menú al usuario con distintas opciones de arranque.
 - Pasa a modo 64 bits (si se aplica) y carga en memoria **WINLOAD.EXE**
- La CPU comienza a ejecutar **WINLOAD.EXE**
 - Carga en memoria **NTOSKRNL.EXE**, **HAL.DLL**, drivers para el arranque y la rama **SYSTEM** del registro



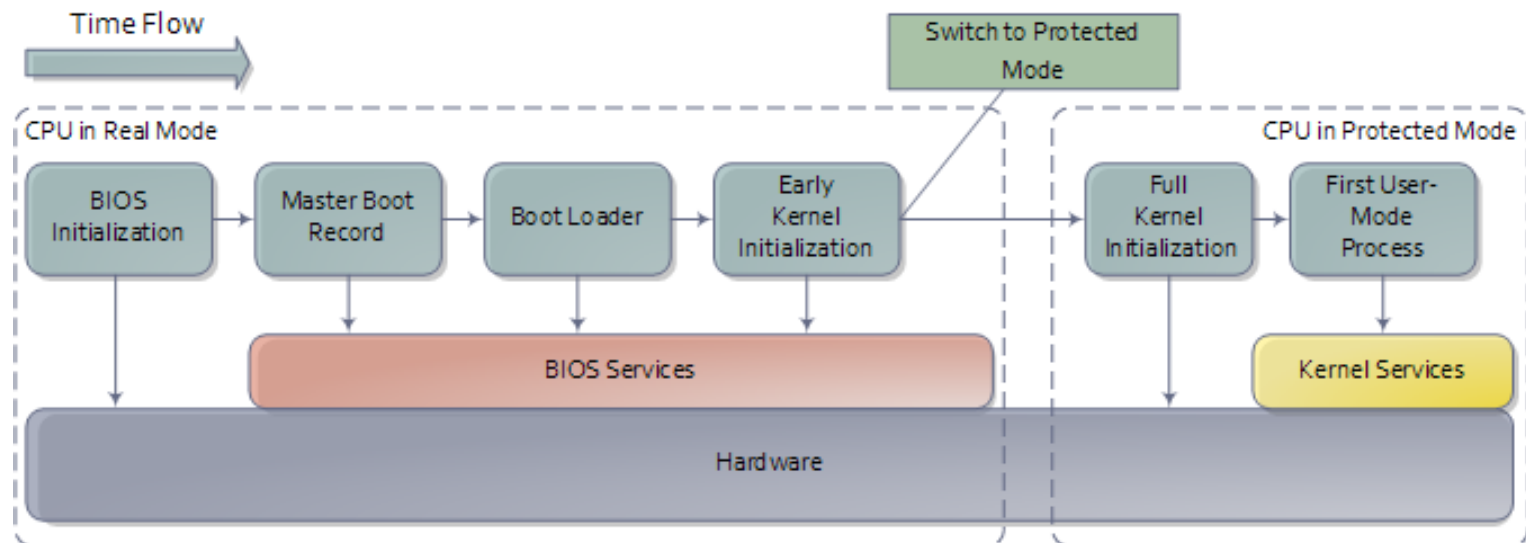
Arranque Windows 7_{PC} (2/2)



- La CPU comienza a ejecutar **NTOSKRNL.EXE**, que se inicializa el sistema en dos fases:
 - Fase 0: Inicializa el kernel en si mismo
 - Inicializa HAL, del driver de pantalla, arranca el depurador.
 - Fase 1: Inicializa el sistema.
 - Carga todos los drivers necesarios y para el depurador.
 - Al finalizar carga el primer procesos de usuario (smss.exe).
- La CPU comienza a ejecutar **SMSS.EXE**
 - El gestor de sesiones carga el resto del registro.
 - Configura el entorno para ejecutar el subsistema Win32 (**WIN32K.SYS**)
 - Carga en memoria el proceso **WINLOGON.EXE**
 - Carga el resto de servicios y drivers no esenciales (mostrar el escritorio antes)
 - Carga el subsistema de seguridad **LSASS.EXE**

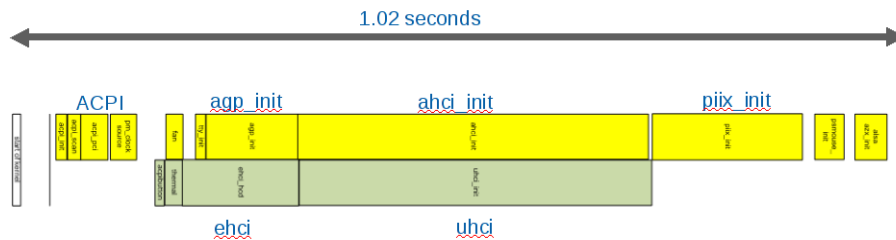
Proceso de arranque_{PC}

resumen

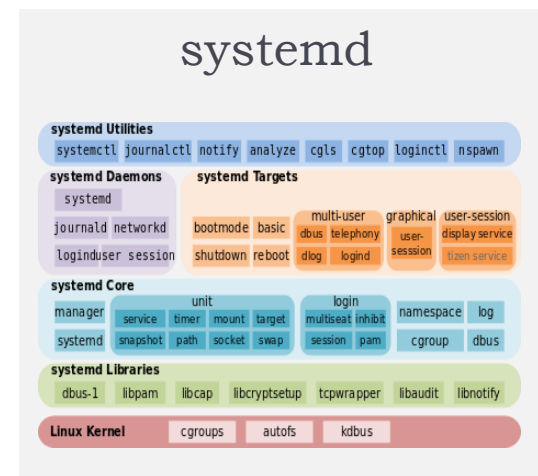
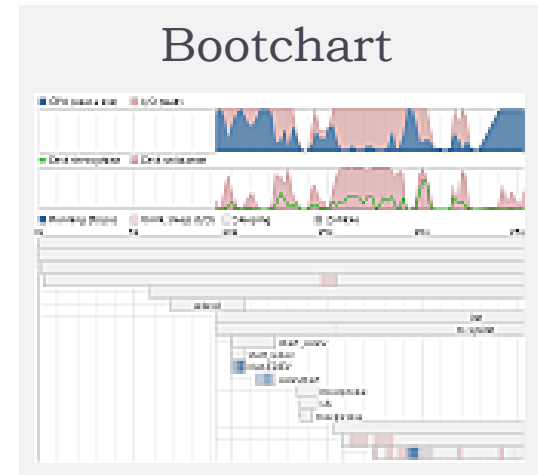
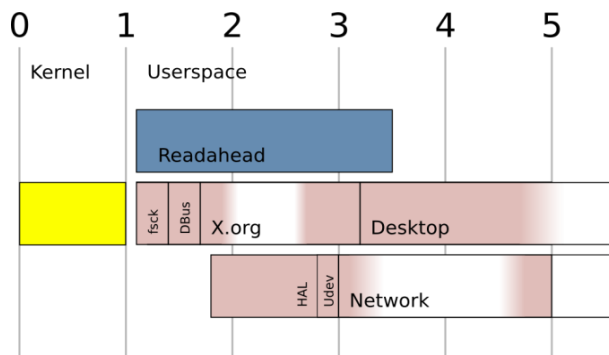


Acelerando el arranque en Linux

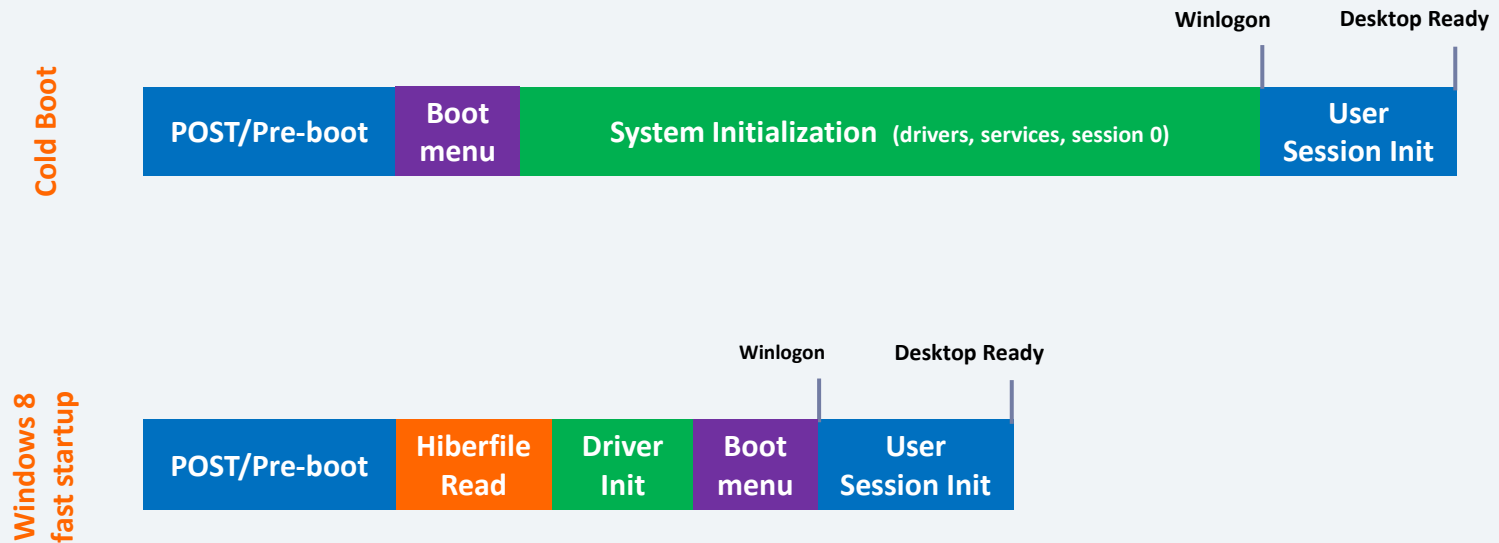
▶ Inicialización asíncrona del hardware



▶ Inicialización asíncrona de servicios



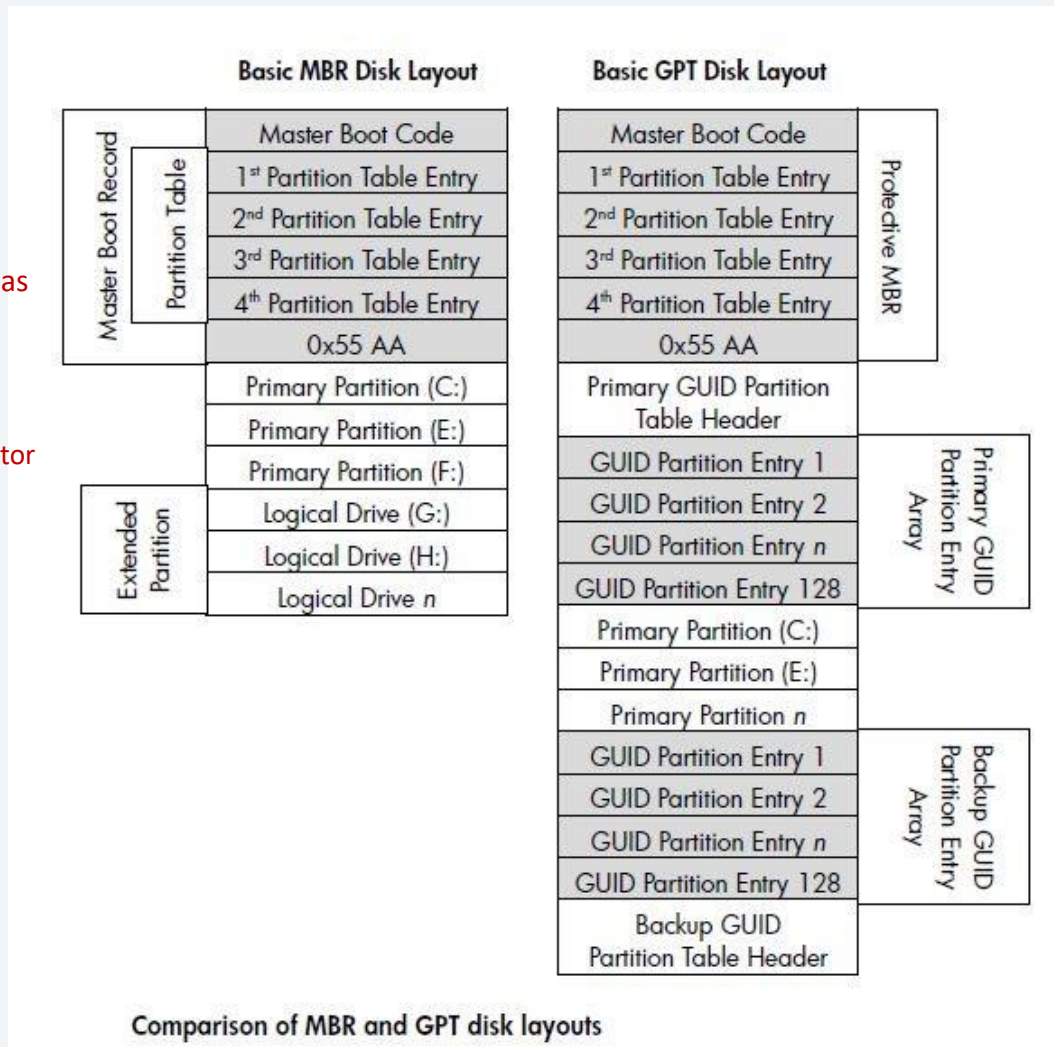
Acelerando el arranque en "Windows 8"



MBR → GPT

Master Boot Record

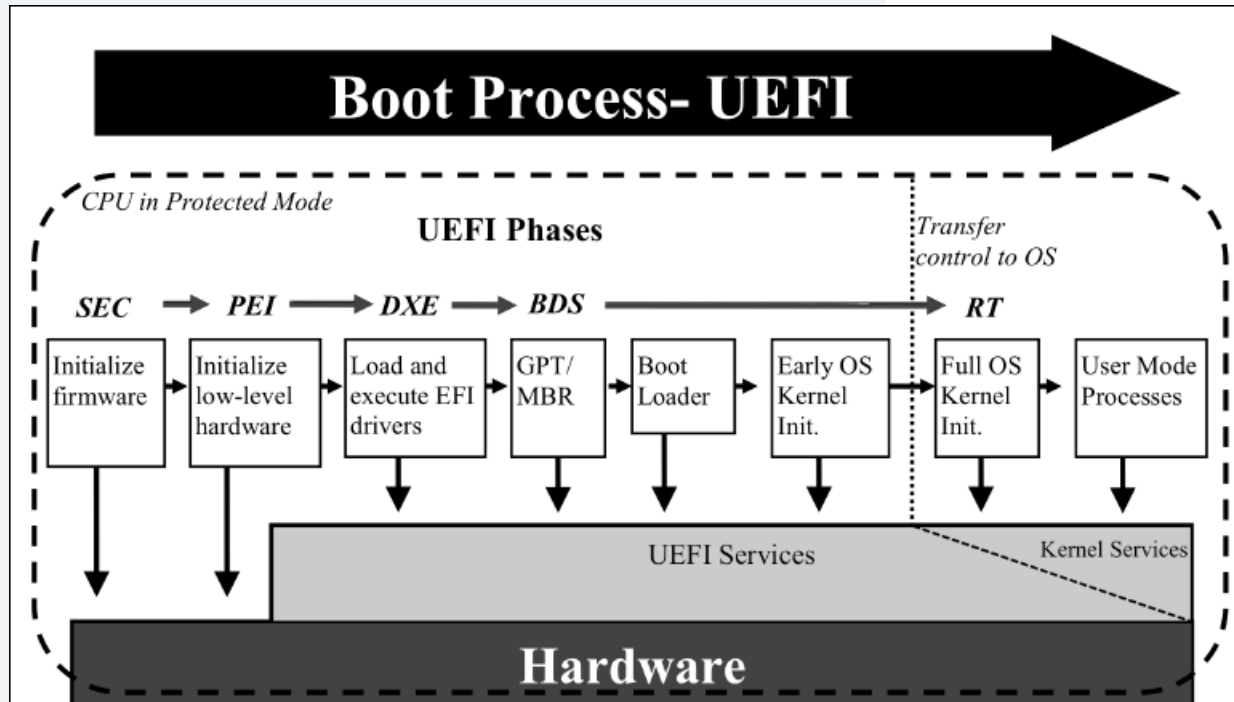
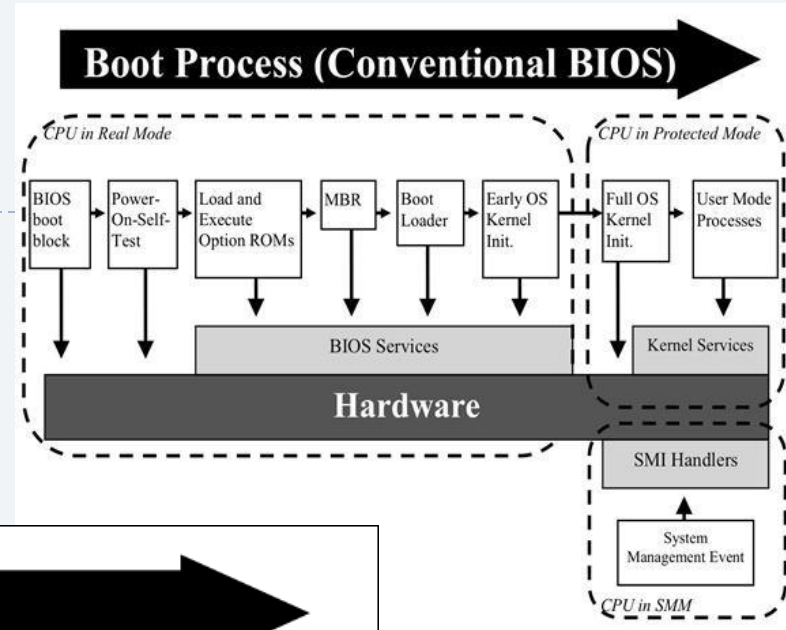
- 4 part. primarias
3P. + 1E. (+n U.L.)
- 32 bits
- 2 TB/part.
 $2^{32} * 512$ bytes/sector
- BIOS
- Viejo S.O.
- 1 MBR +
no CRC32



GUID Partition Table

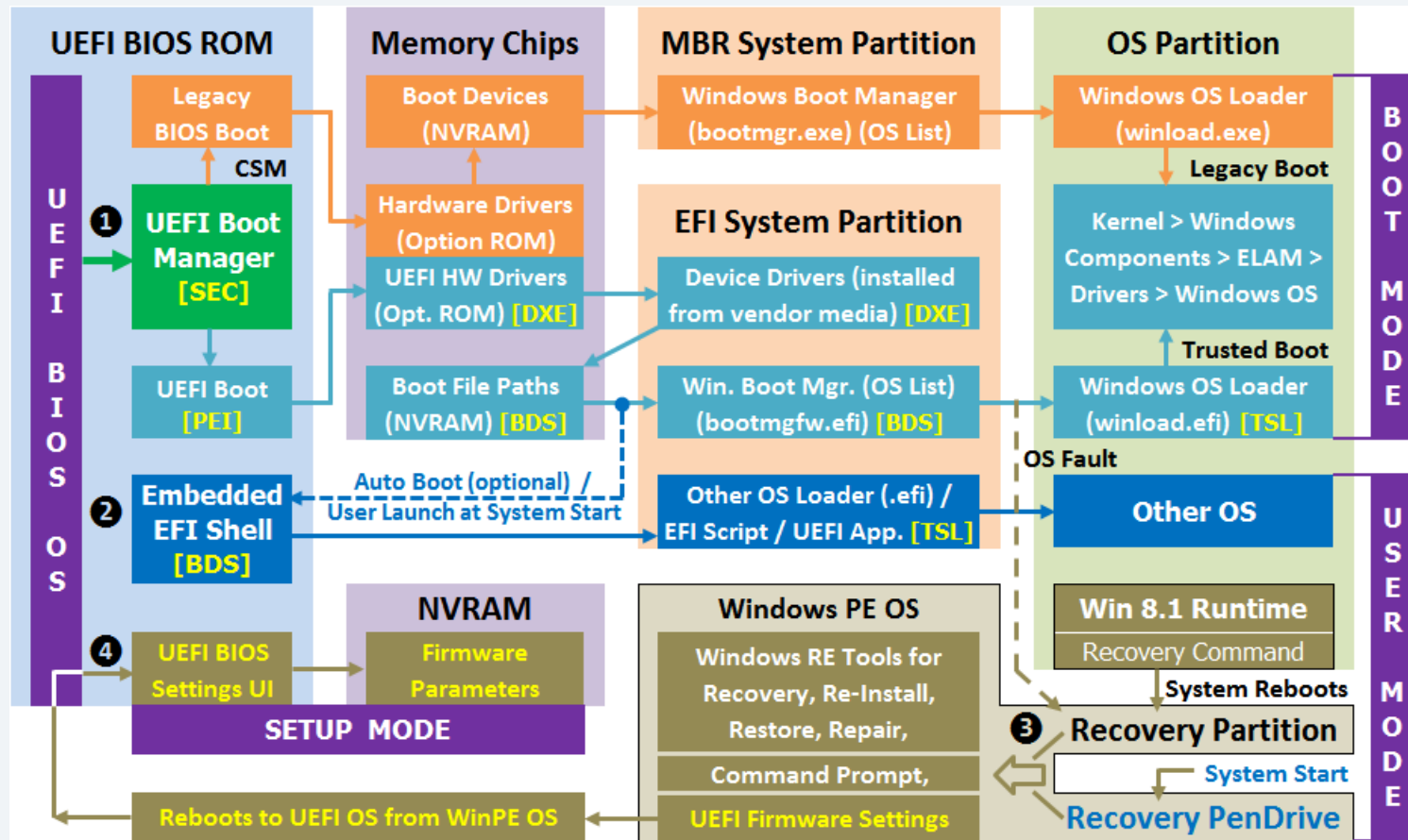
- 128 part.
128 en varios SS.OO.
- 64 bits
- 9 ZB/part.
 $2^{64} * 512$ bytes/sector
- UEFI
- Nuevo S.O.
- 2 GPT +
CRC32
más seguridad

BIOS → UEFI



UEFI

Máquina de Estados Finitos para MBR + GPT



GPT + UEFI

Ejemplo de particiones obligatorias con arranque dual

Partition	File System	Mount Point	Label	Size	Used	Unused	Flags
/dev/sda1	fat32	/boot/efi	SYSTEM	300.00 MiB	34.99 MiB	265.01 MiB	boot
/dev/sda2	ntfs		Recovery	600.00 MiB	307.02 MiB	292.98 MiB	hidden, diag
/dev/sda3	unknown			128.00 MiB	--	--	msftres
/dev/sda4	ntfs		OS	45.50 GiB	38.69 GiB	6.80 GiB	
/dev/sda7	ext4	/		41.09 GiB	29.66 GiB	11.43 GiB	
/dev/sda8	linux-swap			3.89 GiB	--	--	
/dev/sda5	unknown			4.00 GiB	--	--	hidden
/dev/sda6	ntfs		Restore	20.00 GiB	10.01 GiB	9.99 GiB	hidden, diag

0 operations pending

EFI

W-recovery

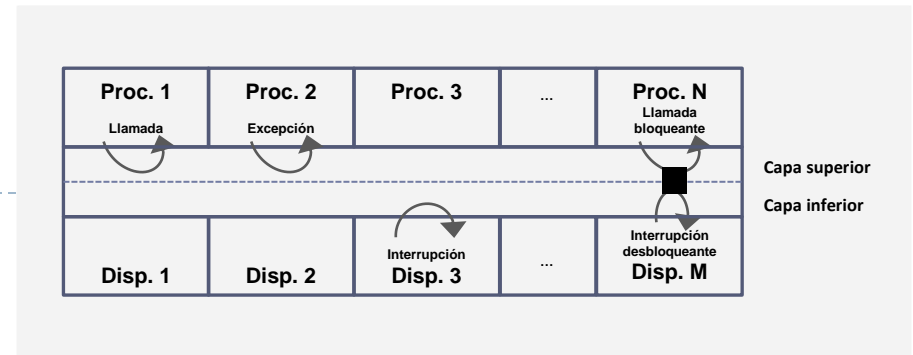
W-MSR

W-sistema

L-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

Tipos de eventos

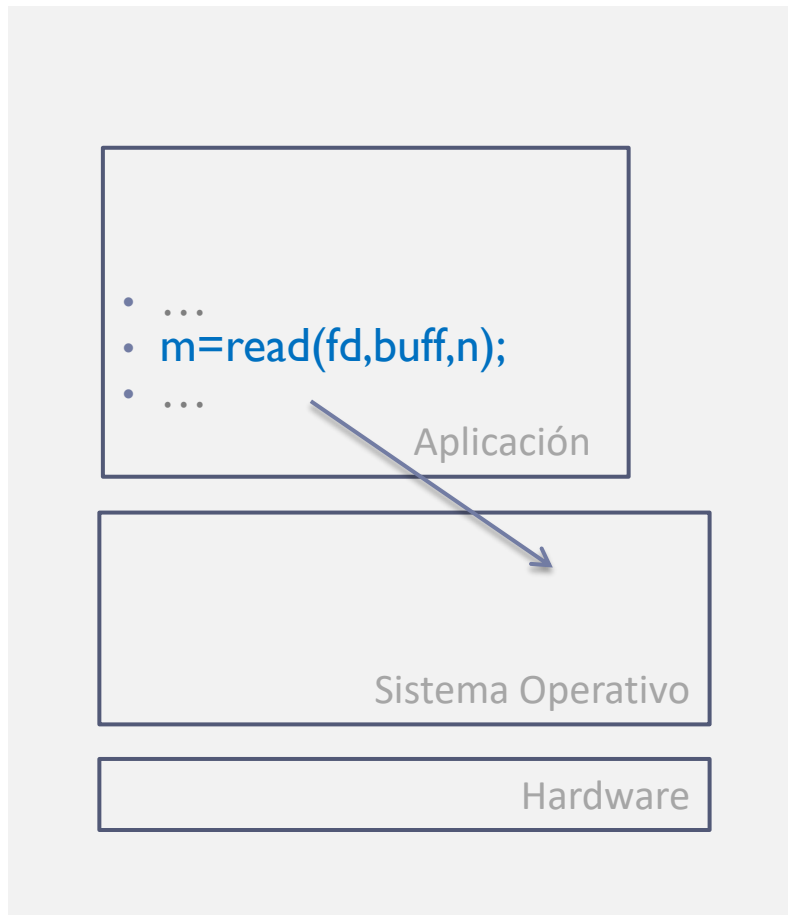
 Usuario

- ▶ **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

Hardware

Tipos de eventos

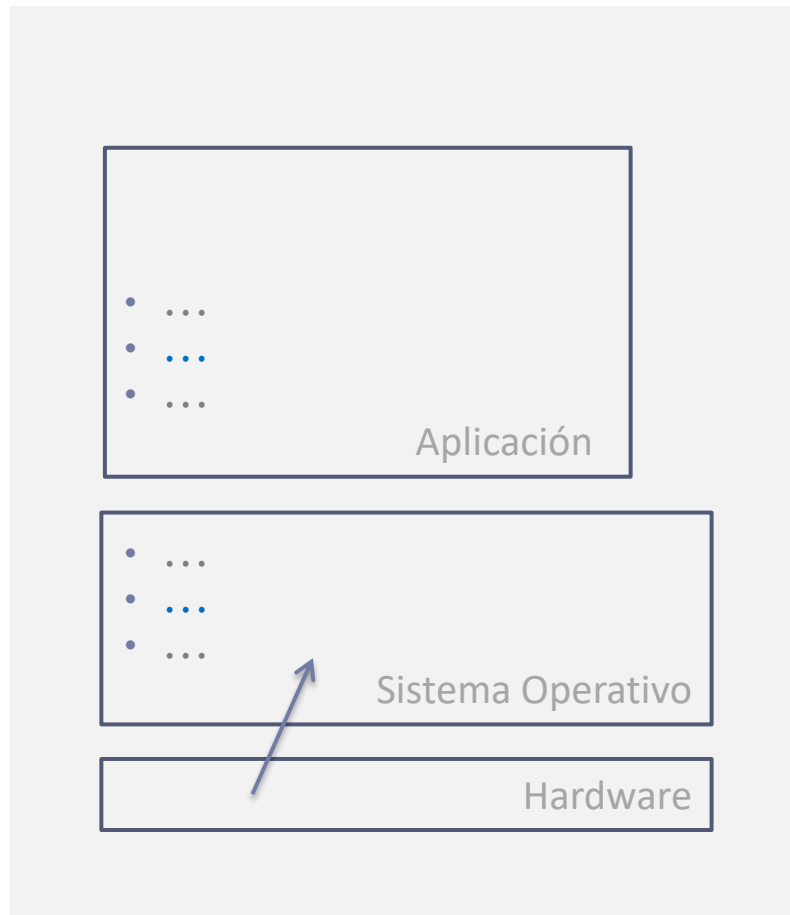
Llamadas al sistema



- ▶ 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

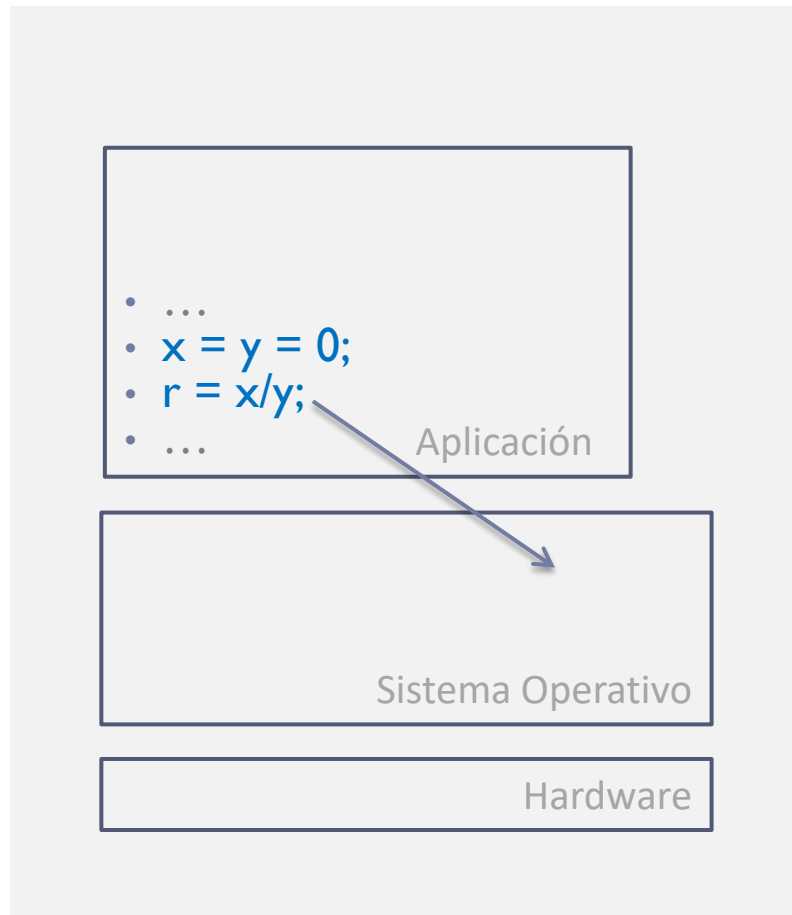
Interrupciones hardware



- ▶ 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

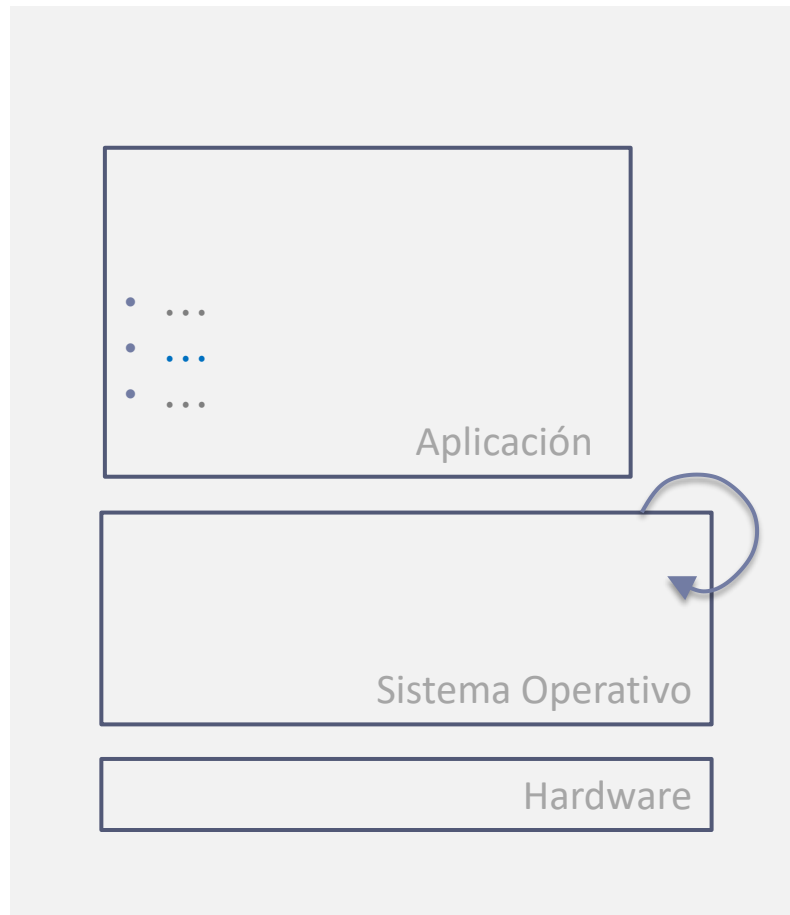
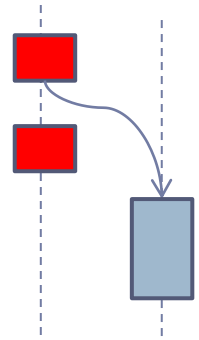
Excepciones



- ▶ 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

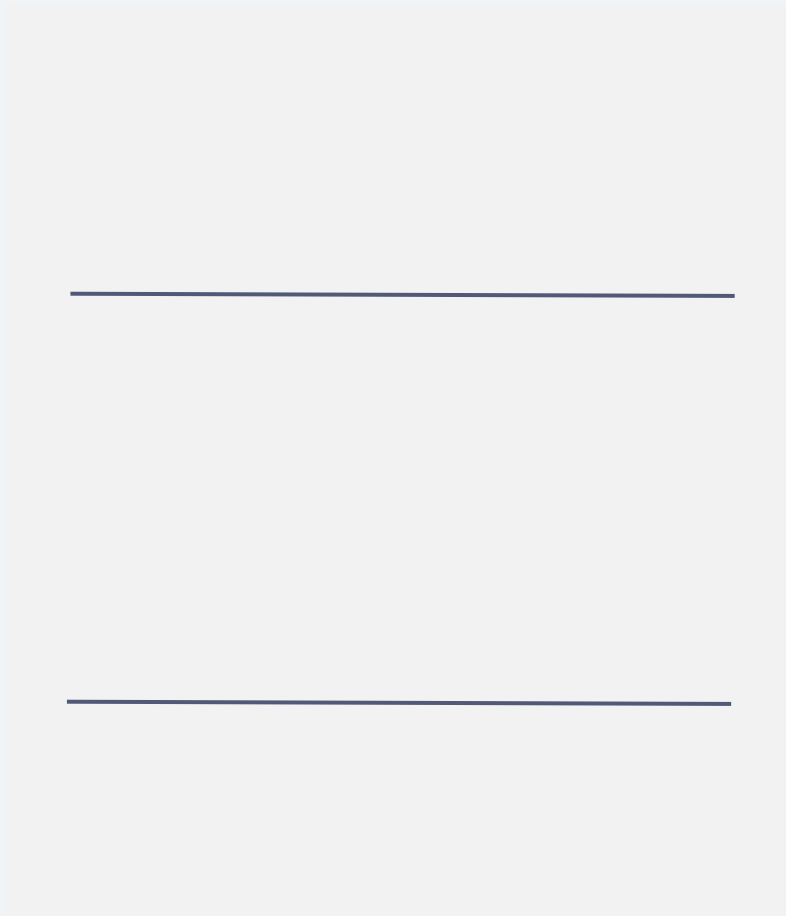
Interrupciones software



- ▶ 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.

Metáfora: la tienda de libros...

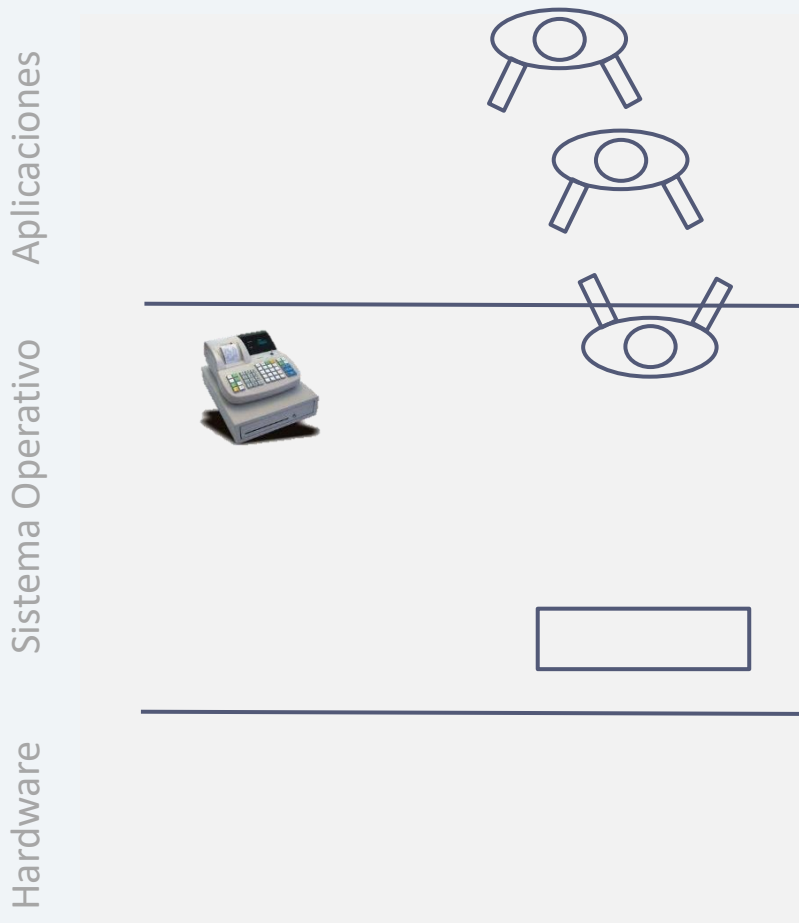
Aplicaciones
Sistema Operativo
Hardware



- ▶ **Dispositivos hardware**
proveedores.
 - ▶ **Ordenador**
la tienda de libros.
 - ▶ **CPU y RAM**
vendedor y estanterías.
- ▶ **Sistema operativo**
Libro de instrucciones que
ha de seguir el vendedor.
- ▶ **Procesos**
compradores.

Metáfora: la tienda de libros...

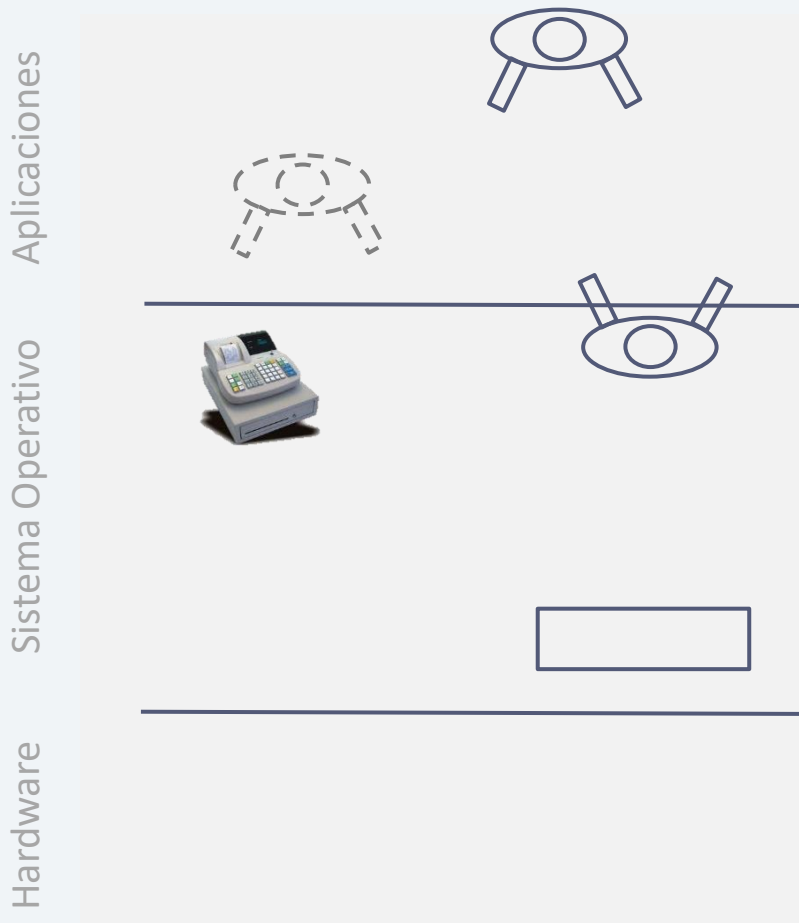
Llamada al sistema



- ▶ El comprador pide un libro
- ▶ El proceso solicita una llamada al sistema
- ▶ El vendedor pide el libro al proveedor correspondiente
- ▶ El sistema operativo solicita al disco un bloque de datos
- ▶ El vendedor pone en espera al comprador hasta tener el libro para atender a otras situaciones
- ▶ El sistema operativo bloquea el proceso y ejecuta mientras otro proceso (o tarea pendiente)

Metáfora: la tienda de libros...

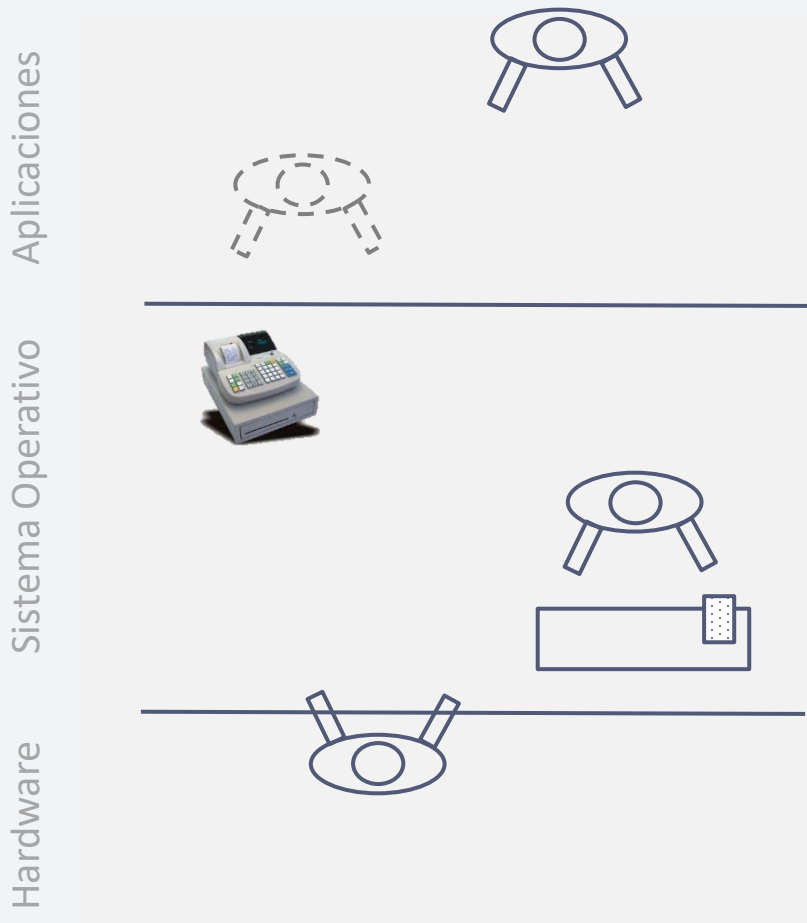
Llamada al sistema



- ▶ El comprador pide un libro
- ▶ El proceso solicita una llamada al sistema
- ▶ El vendedor pide el libro al proveedor correspondiente
- ▶ El sistema operativo solicita al disco un bloque de datos
- ▶ El vendedor pone en espera al comprador hasta tener el libro para atender a otras situaciones
- ▶ El sistema operativo bloquea el proceso y ejecuta mientras otro proceso (o tarea pendiente)

Metáfora: la tienda de libros...

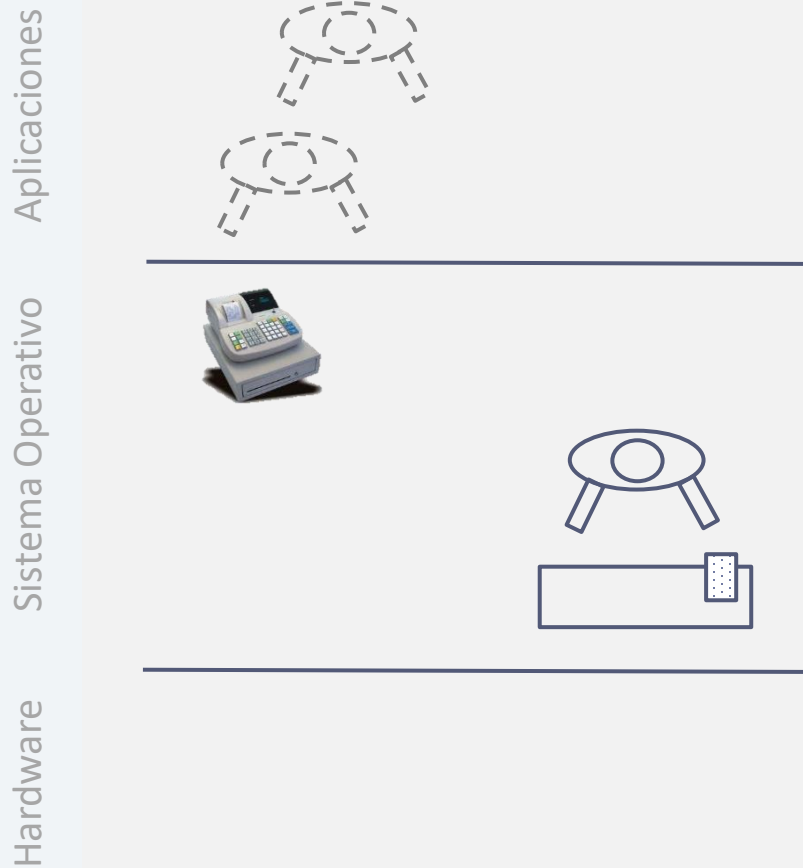
interrupción hardware



- ▶ El proveedor avisa por teléfono que está en la puerta pero no puede aparcar (ha de salir el vendedor)
- ▶ El disco manda una interrupción hardware
- ▶ El vendedor se hace con el libro que colocará en una estantería temporal, junto con un post-it que lo etiqueta como 'a entregar'
- ▶ El sistema operativo copia el bloque de disco a memoria y activa una interrupción software

Metáfora: la tienda de libros...

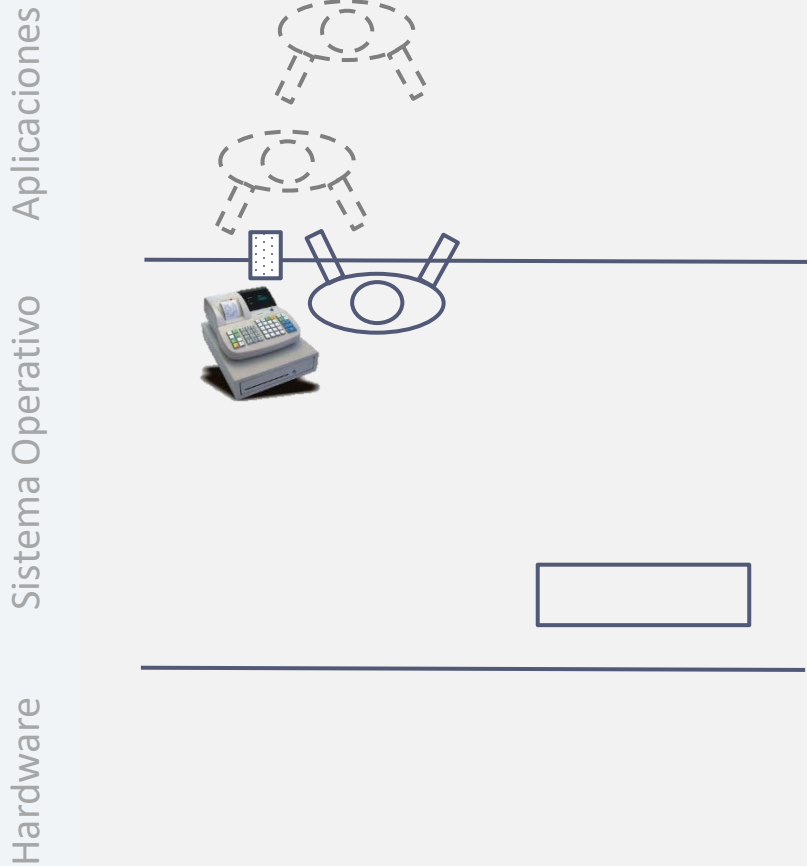
interrupción software



- ▶ Cuando nada más prioritario hay que hacer, se atiende a las entregas pendientes
- ▶ Si no hay evento prioritario, se atiende a las interrupciones software
- ▶ Para cada elementos pendientes de entregar se avisa al comprador de que puede recogerlo
- ▶ Se pasa el proceso a listo para ejecutar, cuando se ejecute copiará los datos de memoria

Metáfora: la tienda de libros...

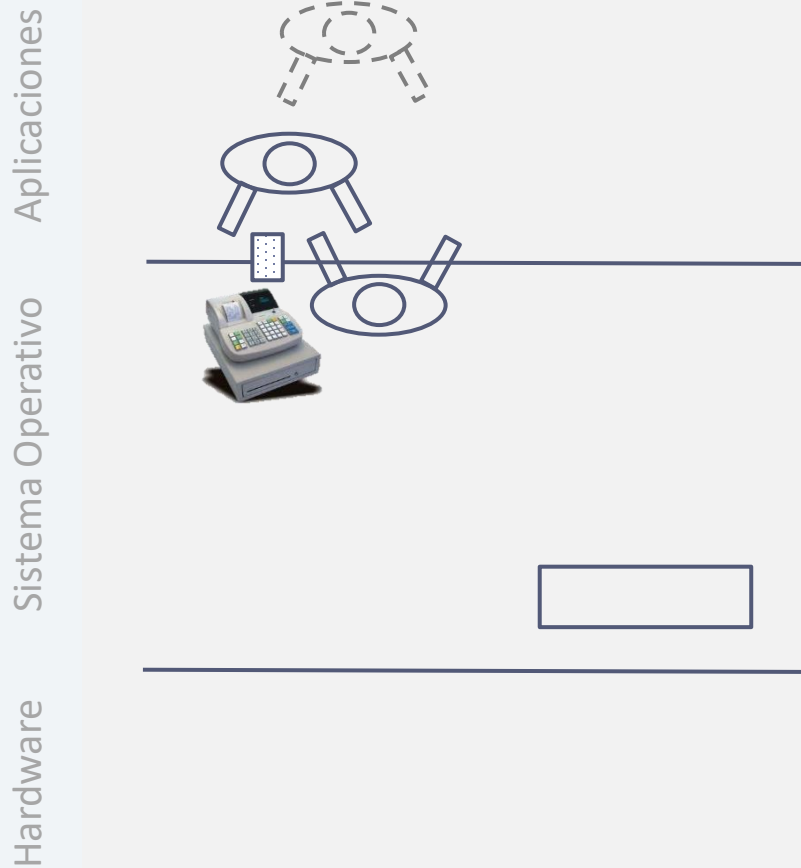
interrupción software



- ▶ Cuando nada más prioritario hay que hacer, se atiende a las entregas pendientes
- ▶ Si no hay evento prioritario, se atiende a las interrupciones software
- ▶ Para cada elementos pendientes de entregar se avisa al comprador de que puede recogerlo
- ▶ Se pasa el proceso a listo para ejecutar, cuando se ejecute copiará los datos de memoria

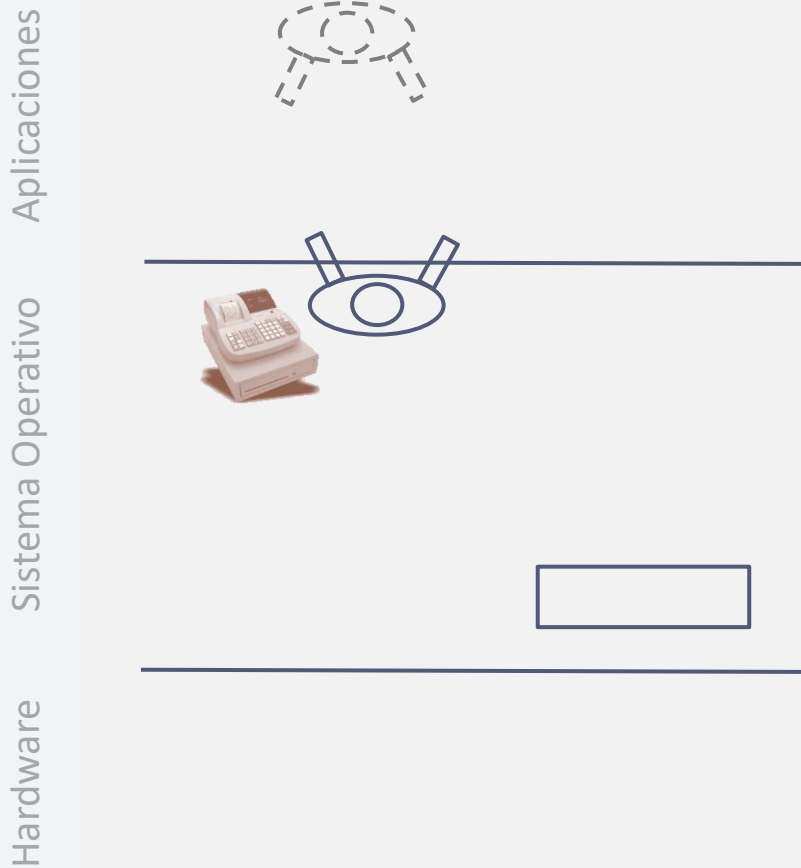
Metáfora: la tienda de libros...

interrupción software



- ▶ Cuando nada más prioritario hay que hacer, se atiende a las entregas pendientes
- ▶ Si no hay evento prioritario, se atiende a las interrupciones software
- ▶ Para cada elementos pendientes de entregar se avisa al comprador de que puede recogerlo
- ▶ Se pasa el proceso a listo para ejecutar, cuando se ejecute copiará los datos de memoria

Metáfora: la tienda de libros... excepción



- ▶ Un comprador quiere café, se le invita a abandonar el local y luego se continúa atendiendo al resto
- ▶ Se produce una excepción mientras se ejecuta un proceso, se mata al proceso
- ▶ Se estropea la caja registradora, hay que cerrar la tienda
- ▶ Se produce una excepción grave mientras se ejecuta el sistema operativo, *kernel-panic*

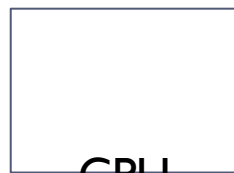
Ejemplo simplificado

App.

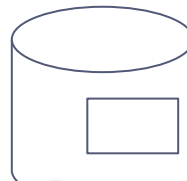
- `char buffer[1024];`
 - ...
 - `read(fd,buffer)`
 - `buffer[2048]='\0';`
-

S.O.
(kernel)

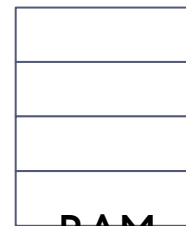
HW.



CPU



Disco



RAM

Ejemplo simplificado

App.

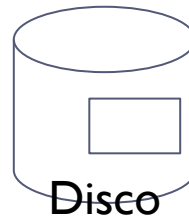
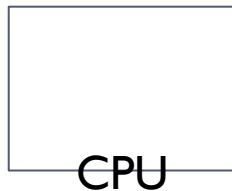
- char buffer[1024];
- ...
- **read**(fd,buffer)
- buffer[2048]='\0';

Il. sistema

- Pedir bloque
- Ejecutar Pi+I

S.O.
(kernel)

HW.



Ejemplo simplificado

App.

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

Il. sistema

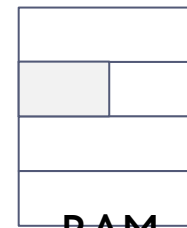
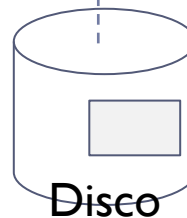
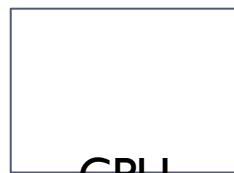
- Pedir bloque
- Ejecutar Pi+I

S.O.
(kernel)

- Copiar a RAM
- Act. int. soft.

int. hw

HW.



Ejemplo simplificado

App.

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

Il. sistema

- Pedir bloque
- Ejecutar Pi+I

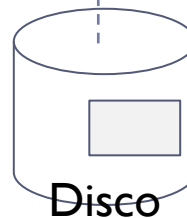
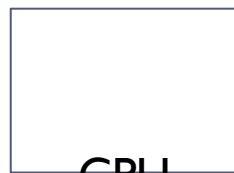
S.O.
(kernel)

- Copiar a RAM
- Act. int. soft. → • Pi listo

int. sw

int. hw

HW.



Ejemplo simplificado

App.

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

Il. sistema

- Pedir bloque
- Ejecutar Pi+I

S.O.
(kernel)

• SIGSEGV

- Copiar a RAM
- Act. int. soft. →

• Pi listo

excep.

int. hw

int. sw

HW.

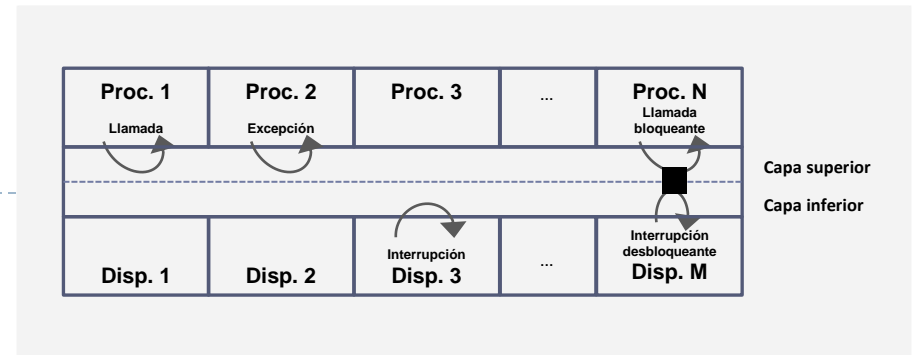
CPU

Disco

RAM



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

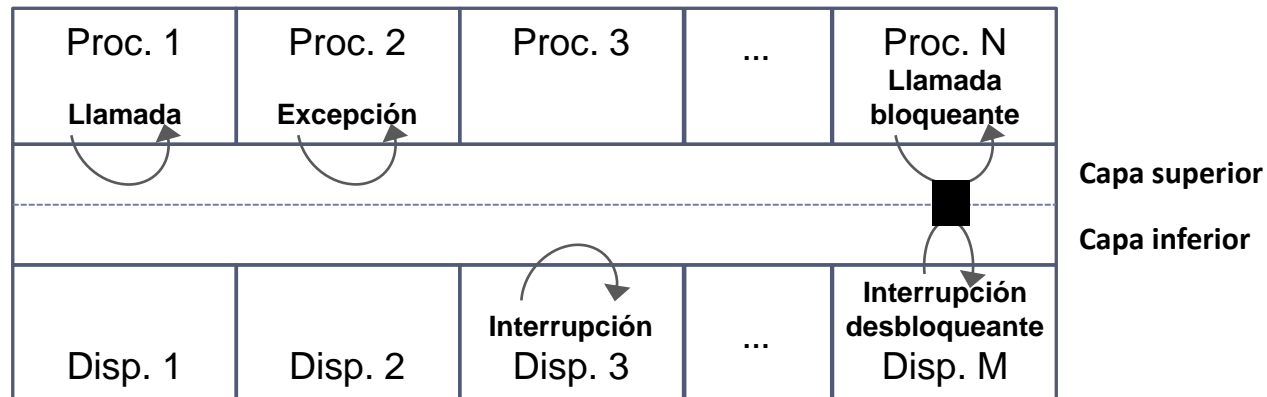


Características básicas...

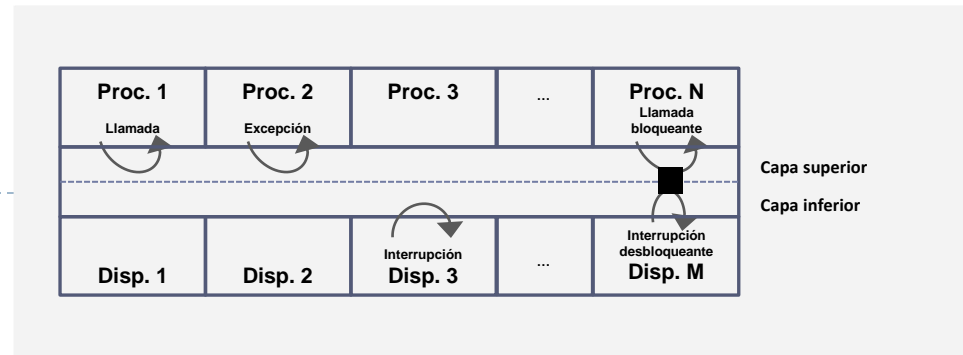
	Modo de ejecución previo	Generadas por
Interrupción hardware	<ul style="list-style-type: none"> • Puede ser usuario o sistema • NO influye en el tratamiento 	<ul style="list-style-type: none"> • Dispositivos de E/S • Interrupción entre las CPU (IPI)
Excepciones	<ul style="list-style-type: none"> • Puede ser usuario o sistema • SI influye en el tratamiento 	<ul style="list-style-type: none"> • La CPU (int. hw. de CPU) • Normalmente errores de programación, NO siempre (fallo de página, depuración, etc.)
Llamadas al sistema	<ul style="list-style-type: none"> • Siempre usuario 	<ul style="list-style-type: none"> • Las aplicaciones
Software	<ul style="list-style-type: none"> • Siempre sistema 	<ul style="list-style-type: none"> • 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

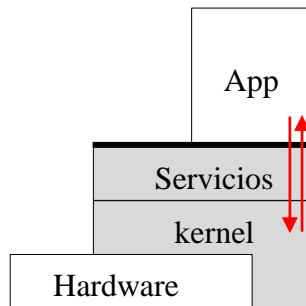
- ▶ El esquema del S.O. busca ser genérico e independiente de la arq. de hw.
 - ▶ Linux sin prioridad (SPARC si soporta) y Windows con prioridad (Intel no lo soporta)

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

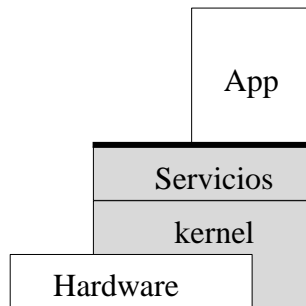
- ▶ El esquema del S.O. busca ser genérico e independiente de la arq. de hw.
 - ▶ Linux sin prioridad (SPARC si soporta) y Windows con prioridad (Intel no lo soporta)
- ▶ Todos los eventos se tratan de forma similar (~int. hw.)
 - ▶ Se ha ido introduciendo ya la gestión de eventos



- ▶ 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

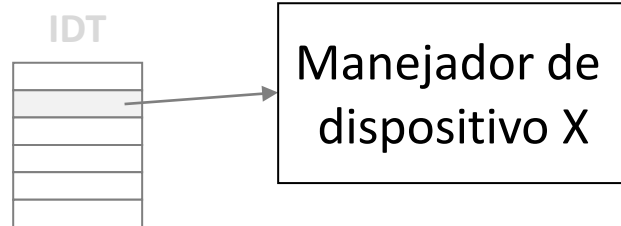
- ▶ Notifican eventos asíncronos que viene del hardware
- ▶ **Modo de ejecución previo:**
 - ▶ Puede ser usuario o sistema (NO influye en el tratamiento)
- ▶ **Generadas por:**
 - ▶ Dispositivos de E/S
 - ▶ Condiciones críticas en el sistema (ej.: corte energía)
 - ▶ Interrupción entre procesadores (IPI)

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, interrupcionRelej);
    instal_man_int(INT_DISPOSITIVOS, interrupcionDispositivos);
    instal_man_int(LLAM_SISTEMA, tratarLlamadaSistema);
    instal_man_int(INT_SW, interrupcionSoftware);
    ...
}
```

Interrupciones hardware

tratamiento (2/5)

Aplicación

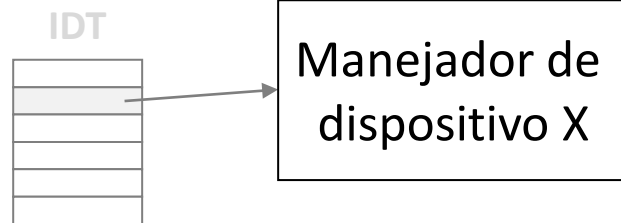


```
#include "servicios.h"

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

Modo Usuario

Modo Kernel



Interrupciones hardware

tratamiento (3/5)

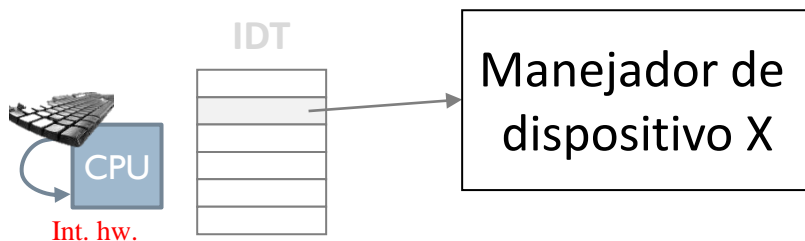
Aplicación

```
#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



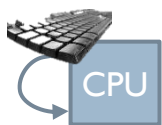
- ▶ Primero **salva estado básico (PC, RE, SP) en la pila de sistema**
- ▶ La CPU pasa en **modo privilegiado** y salta a la **rutina de tratamiento asociada**

Interrupciones hardware

tratamiento (4/5)

Modo Usuario

Modo Kernel



Int. hw.



Manejador de dispositivo X

```
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)

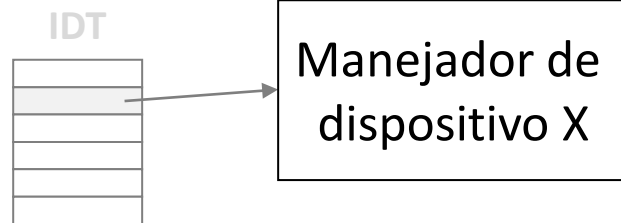
Aplicación

```
#include "servicios.h"

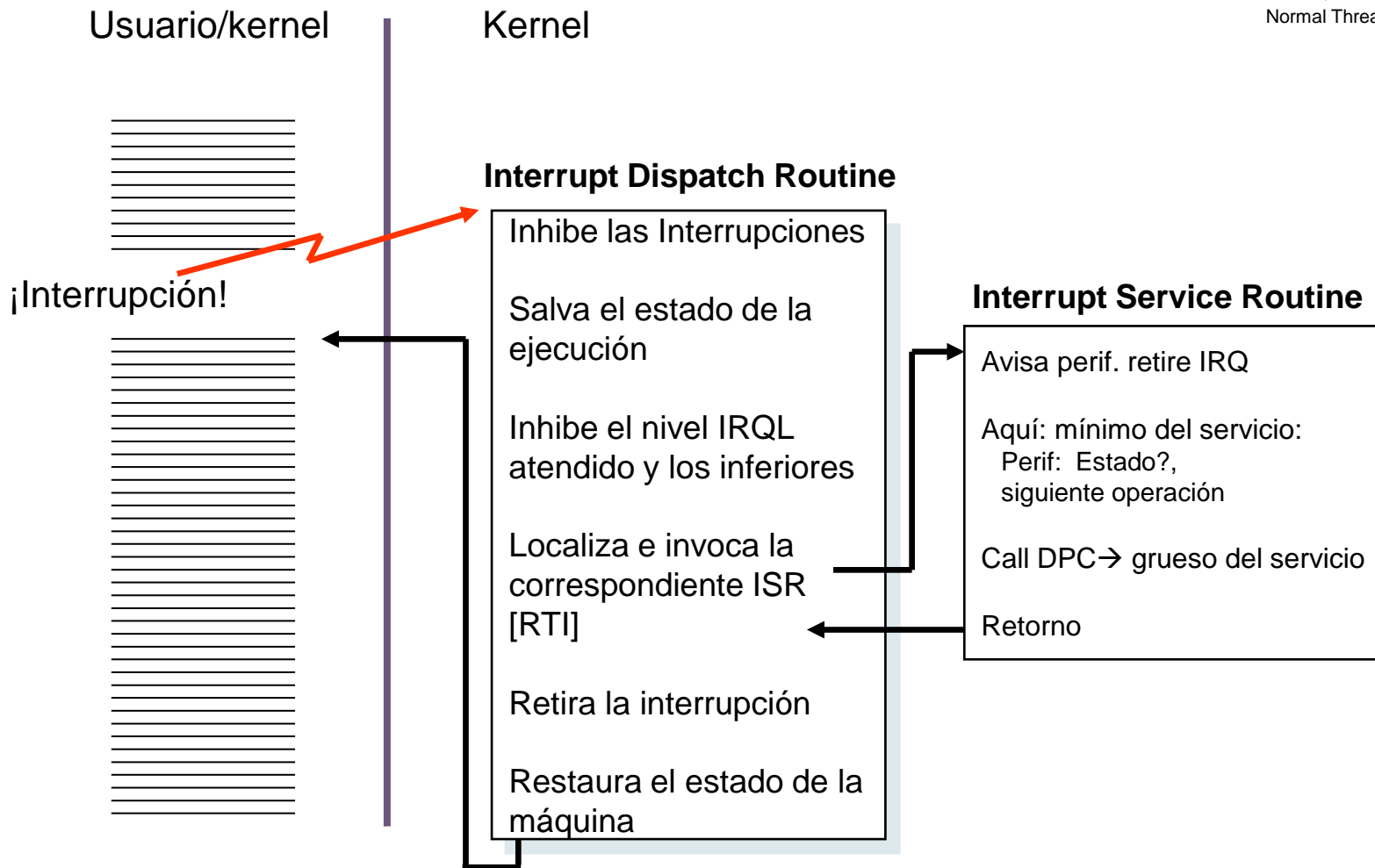
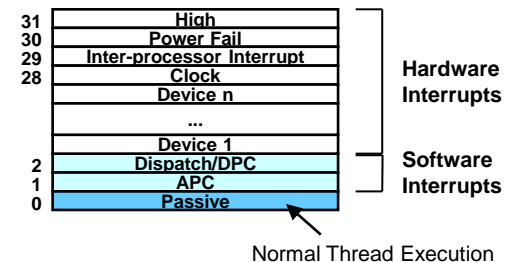
int main ()
{
    for (int i=0; i<1000000; i++)
        printf("resultado = %d\n",calculo_complejo(i));
    return 0;
}
```

Modo Usuario

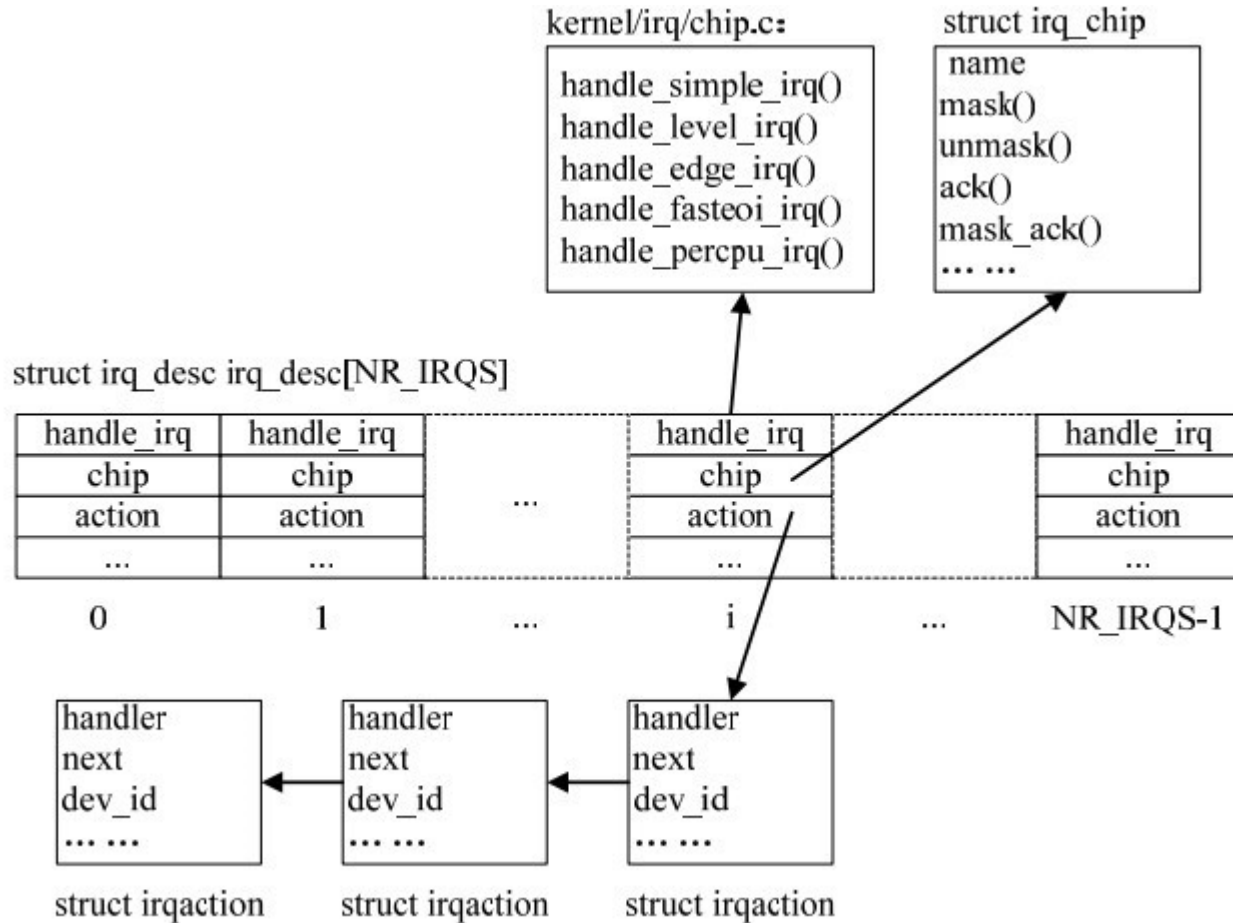
Modo Kernel



Interrupciones hardware tratamiento en Windows



Interrupciones hardware tratamiento en Linux



Excepciones

características

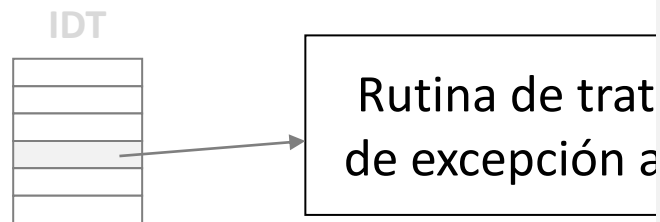
- ▶ Eventos síncronos de carácter excepcional al ejecutar una instrucción
- ▶ **Modo de ejecución previo:**
 - ▶ Puede ser usuario o sistema (SI influye en el tratamiento)
- ▶ **Generadas por:**
 - ▶ El hardware, normalmente errores de programación
 - ▶ NO siempre son errores (fallo de página, depuración,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

Excepciones

tratamiento (1/4)

Modo Usuario

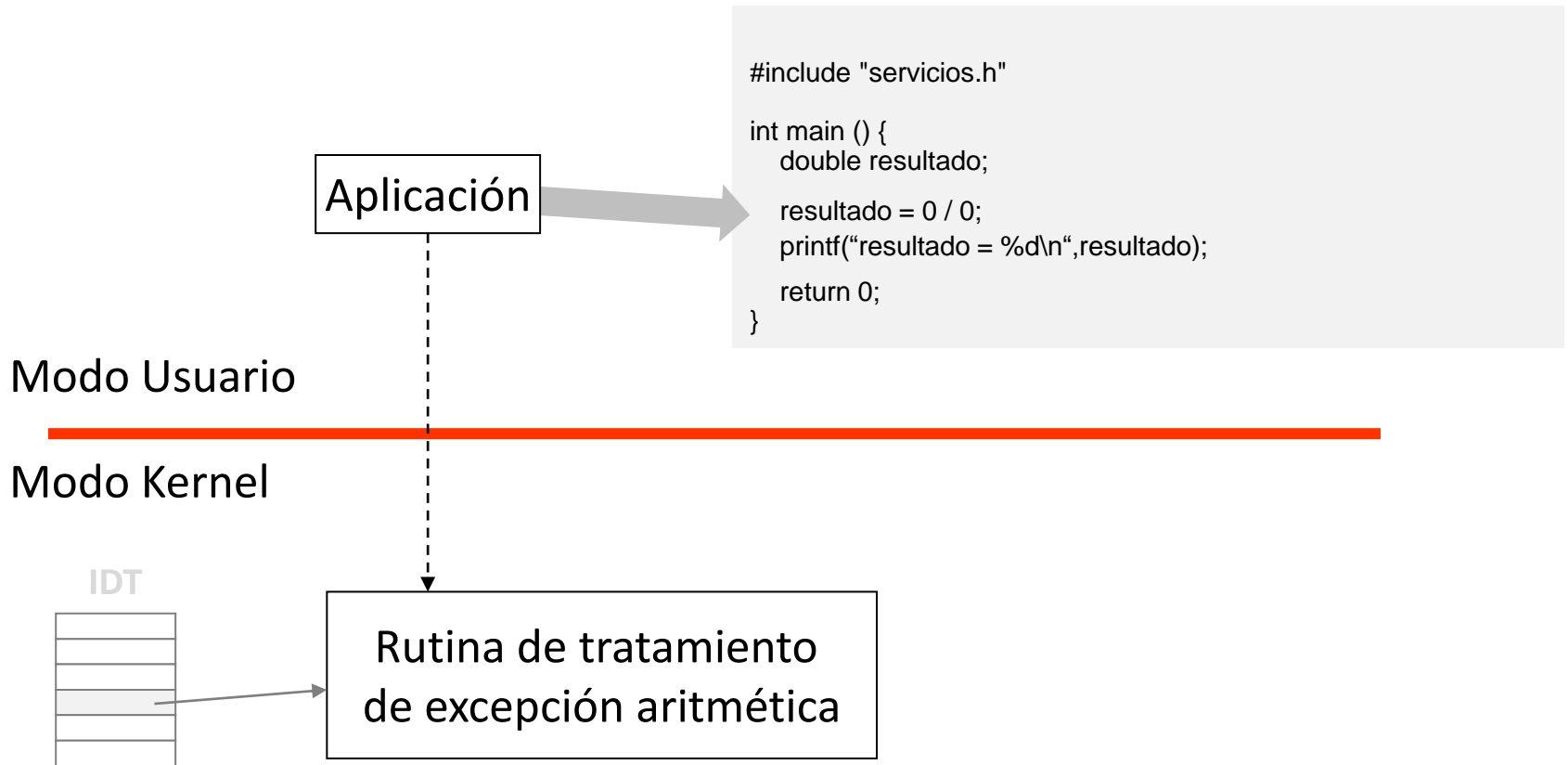
Modo Kernel



```
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, interrupcionRelej);
    instal_man_int(INT_DISPOSITIVOS, interrupcionDispositivos);
    instal_man_int(LLAM_SISTEMA, tratarLlamadaSistema);
    instal_man_int(INT_SW, interrupcionSoftware);
    ...
}
```

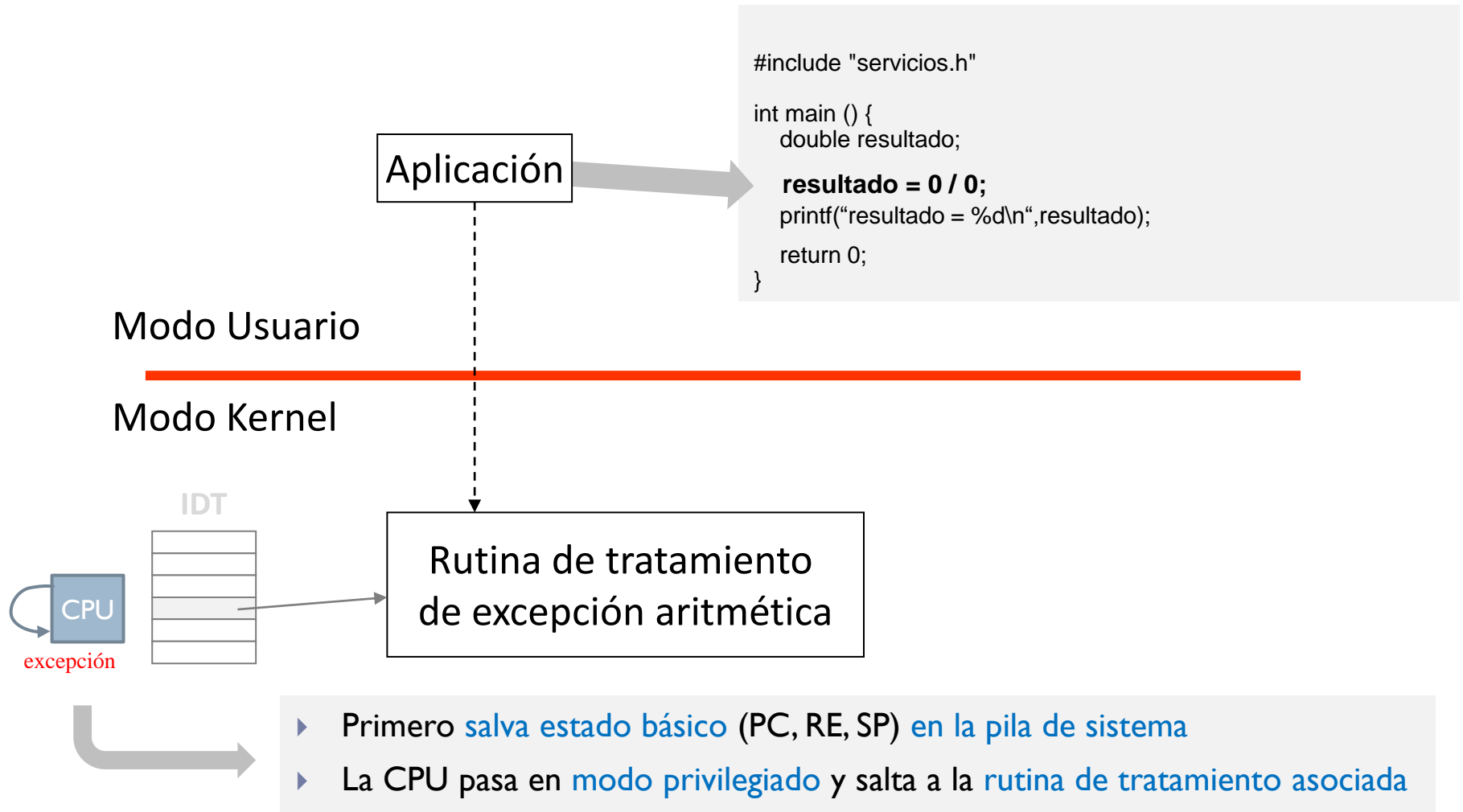
Excepciones

tratamiento (2/4)



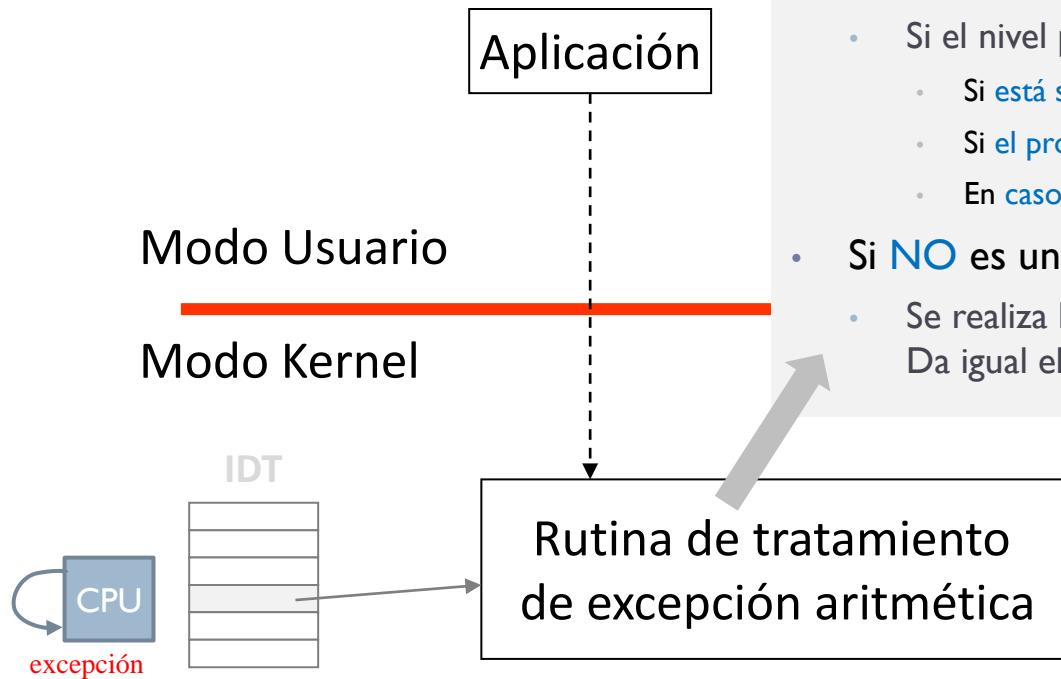
Excepciones

tratamiento (3/4)



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

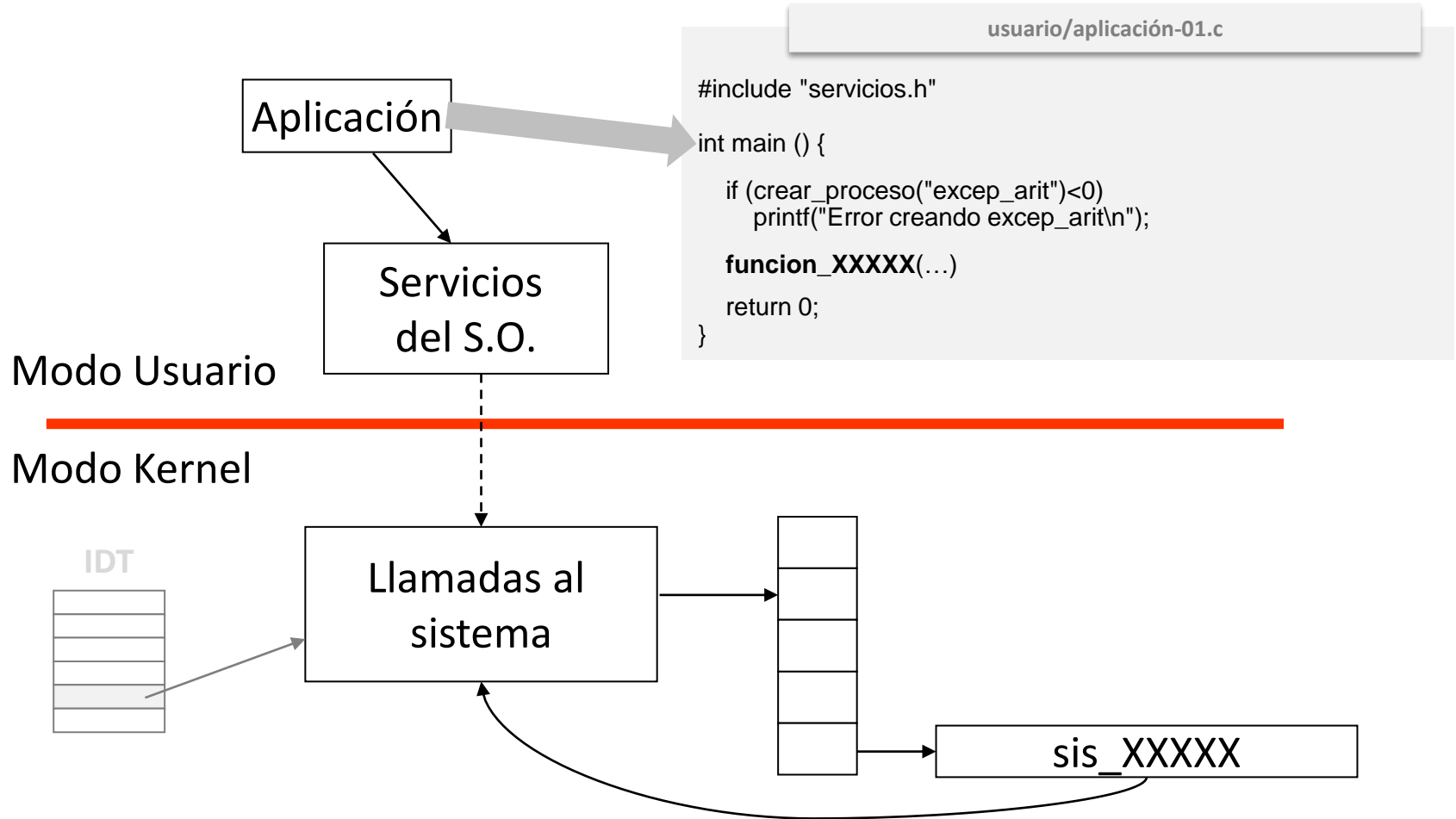
tratamiento

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

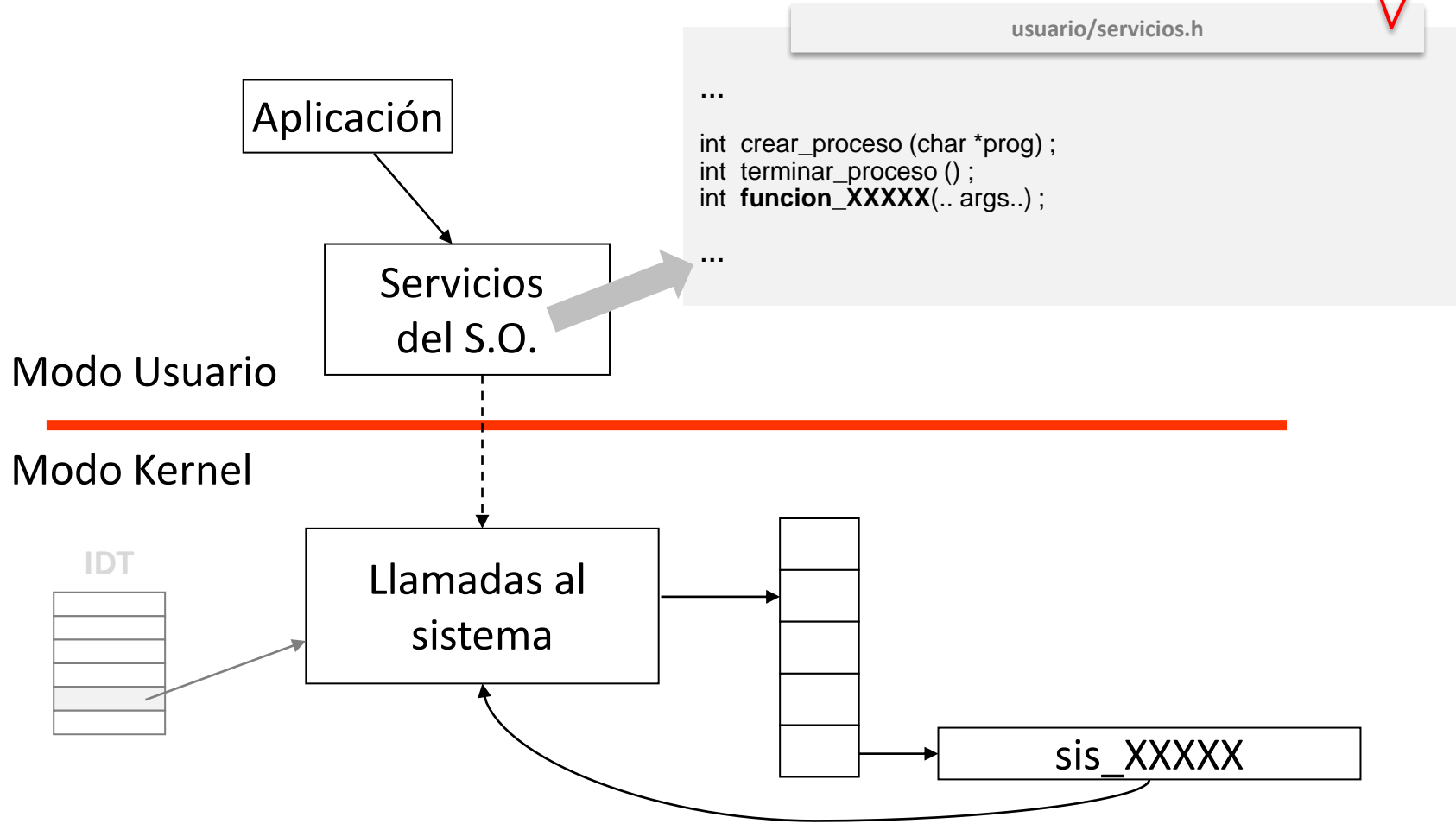
    /* instalar los manejadores para los vectores de interrupción */
    instal_man_int(EXC_ARITMETICA, excepcionAritmetica);
    instal_man_int(EXC_MEMORIA, excepcionMemoria);
    instal_man_int(INT_RELOJ, interrupcionReloj);
    instal_man_int(INT_DISPOSITIVOS, interrupcionDispositivos);
    instal_man_int(LLAM_SISTEMA,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

    ...
}
```

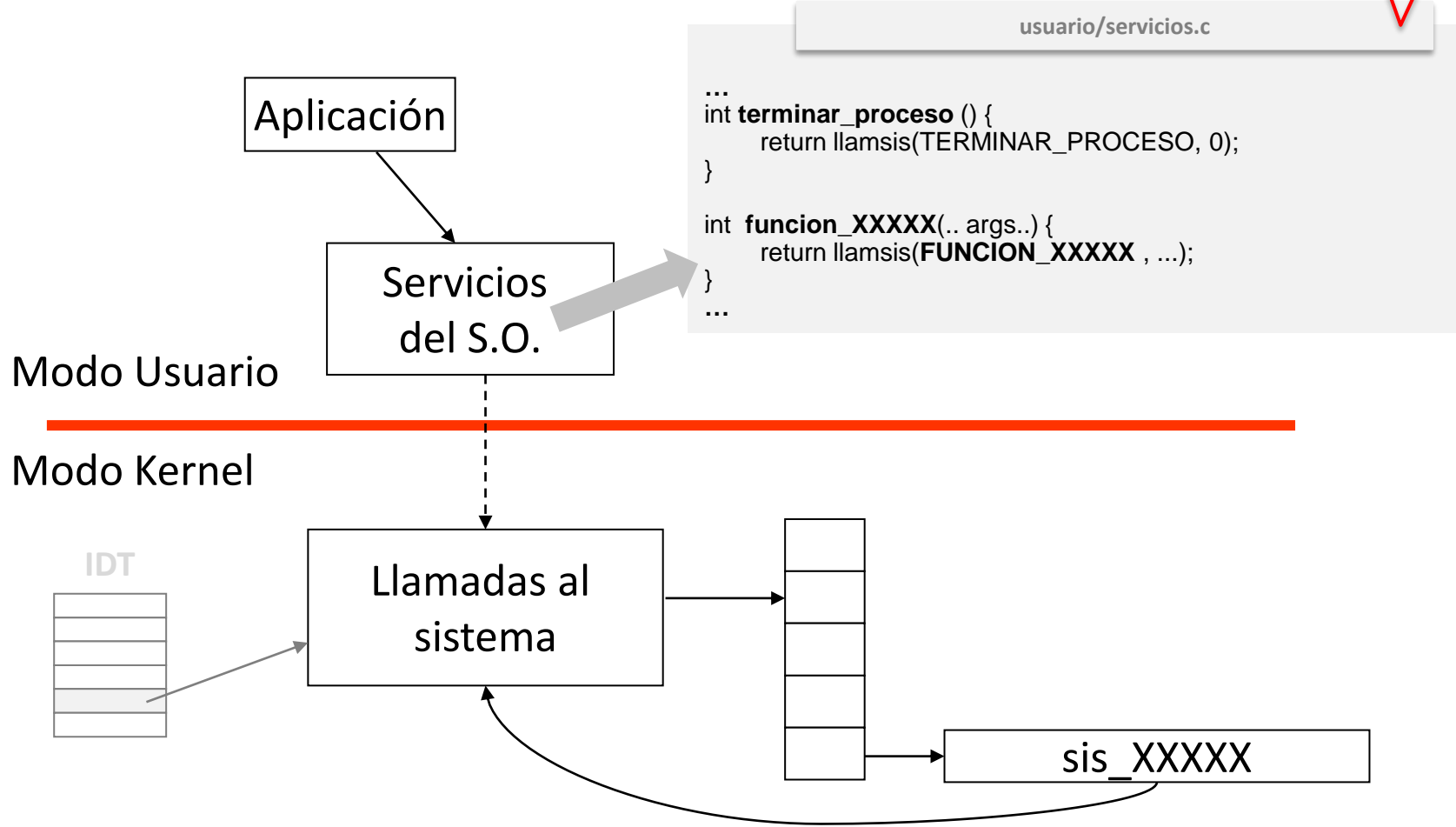
Llamadas al sistema tratamiento (1/9)



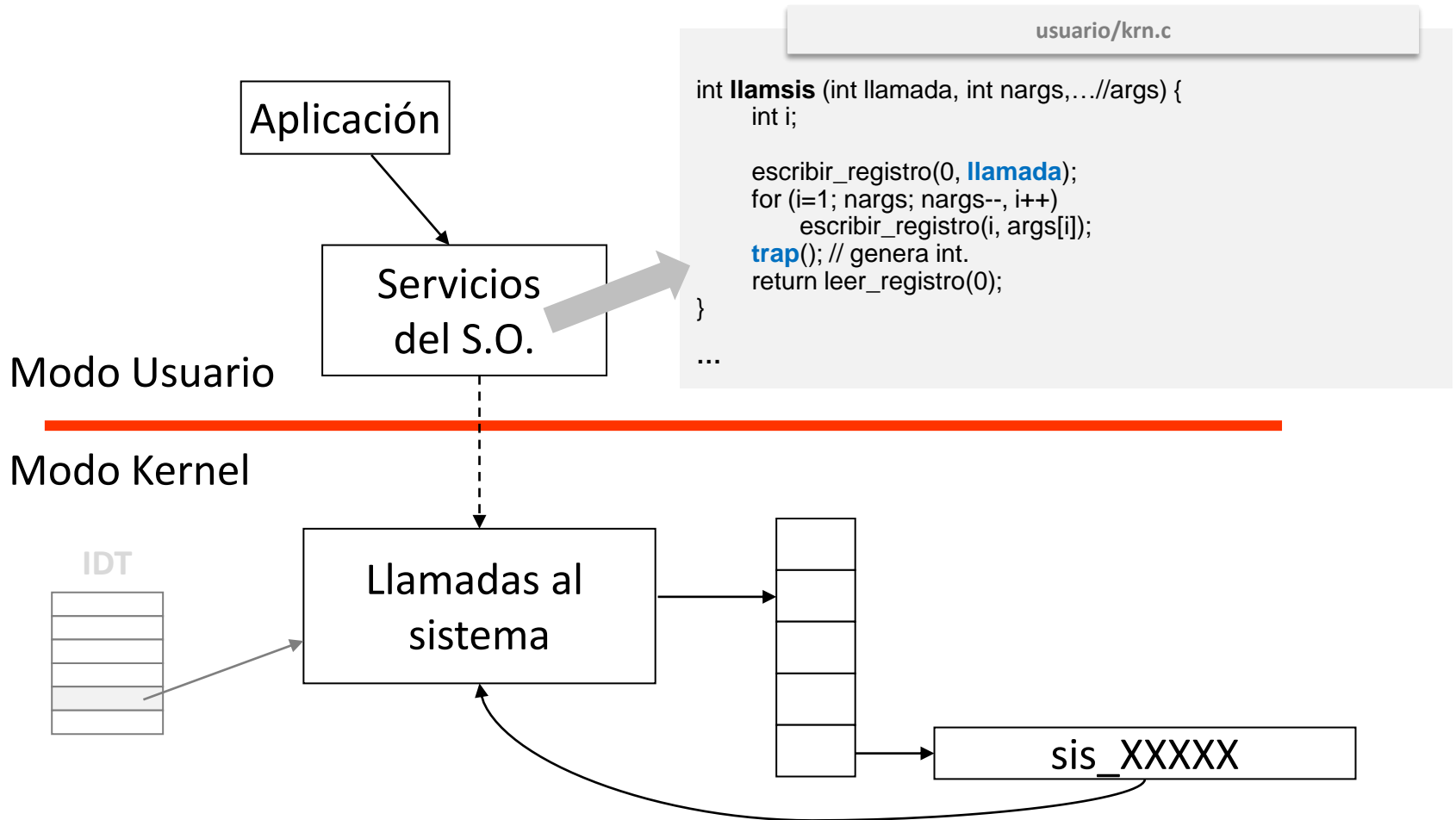
Llamadas al sistema tratamiento (2/9)



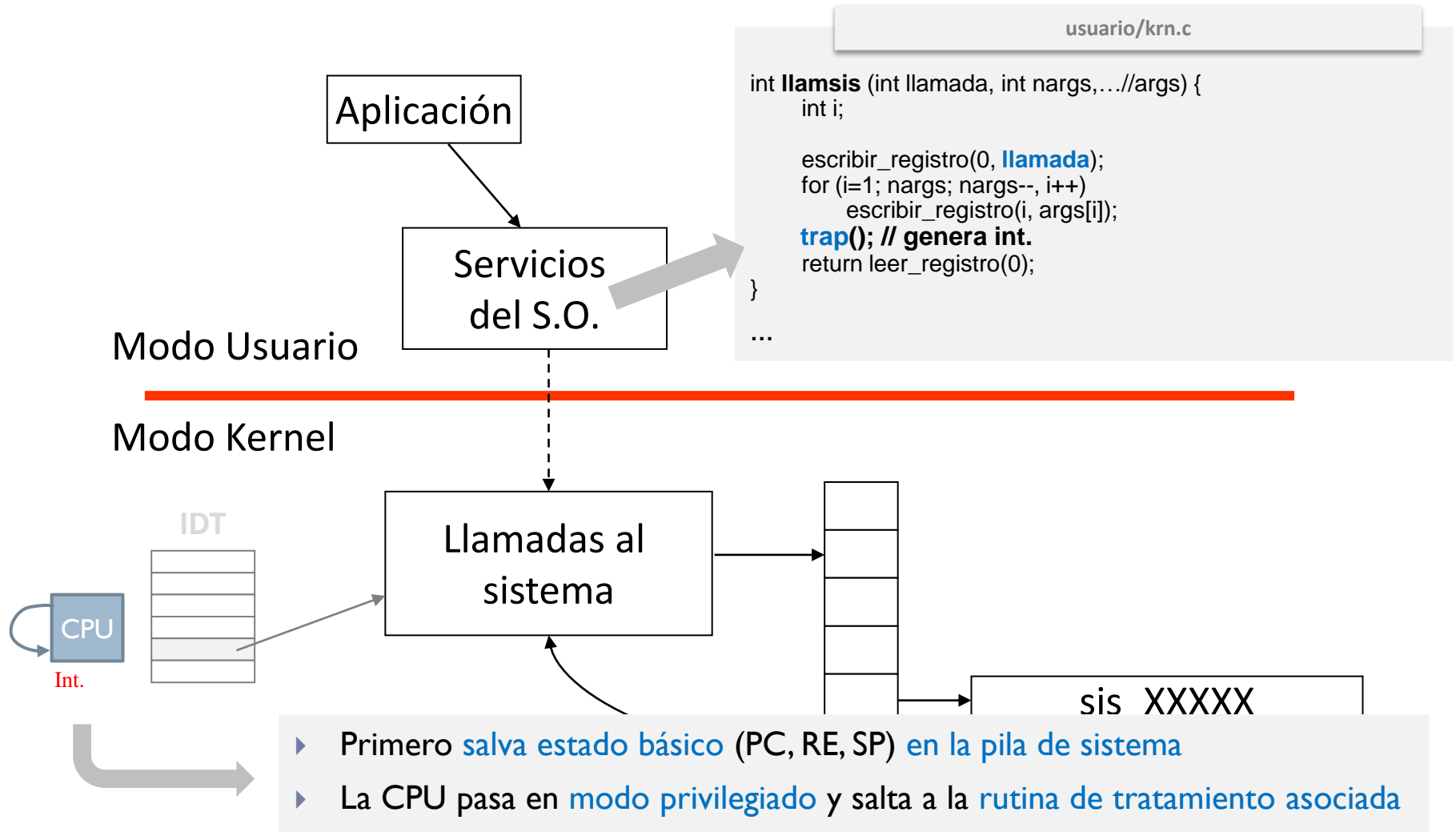
Llamadas al sistema tratamiento (3/9)



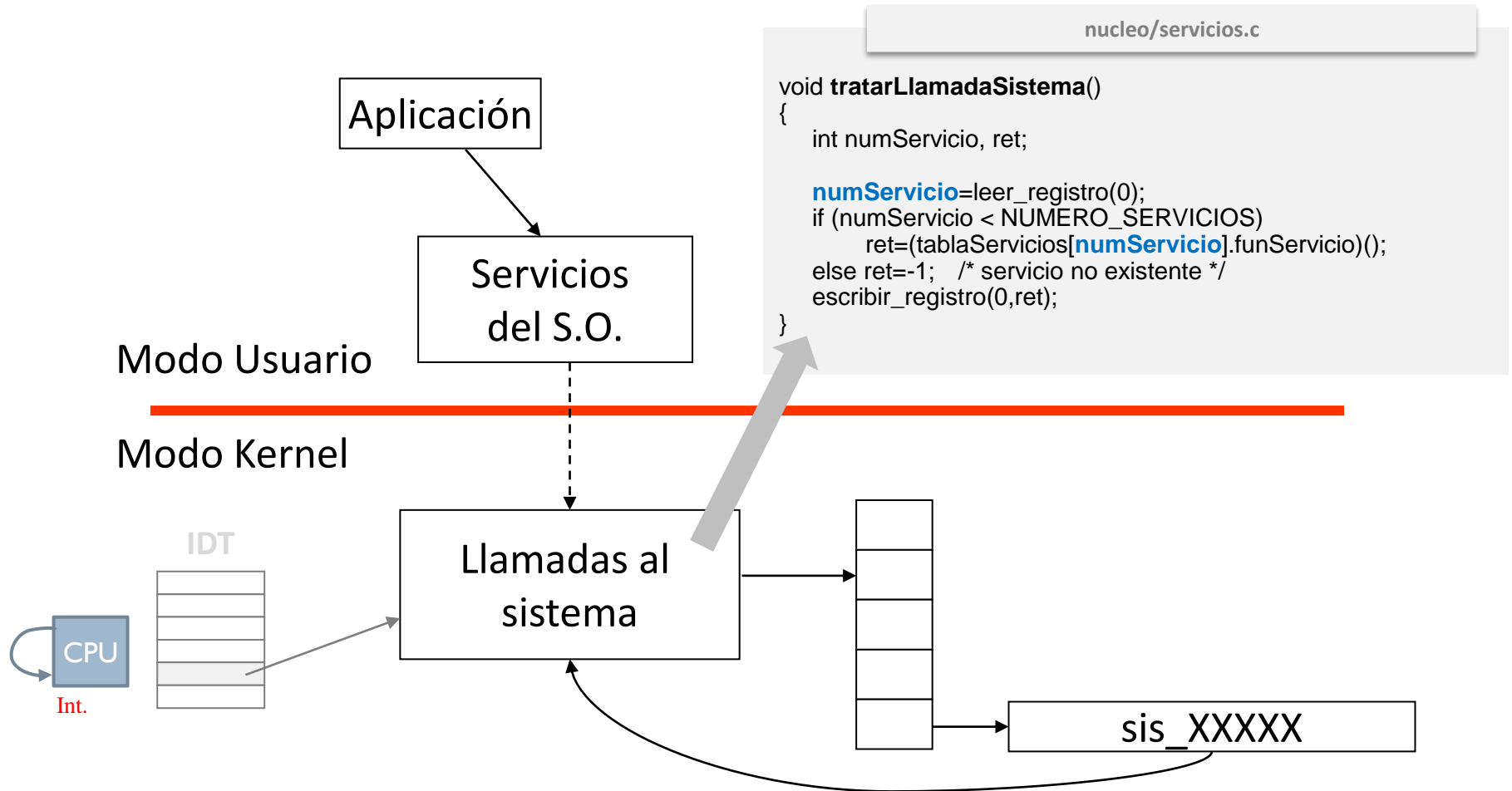
Llamadas al sistema tratamiento (4/9)



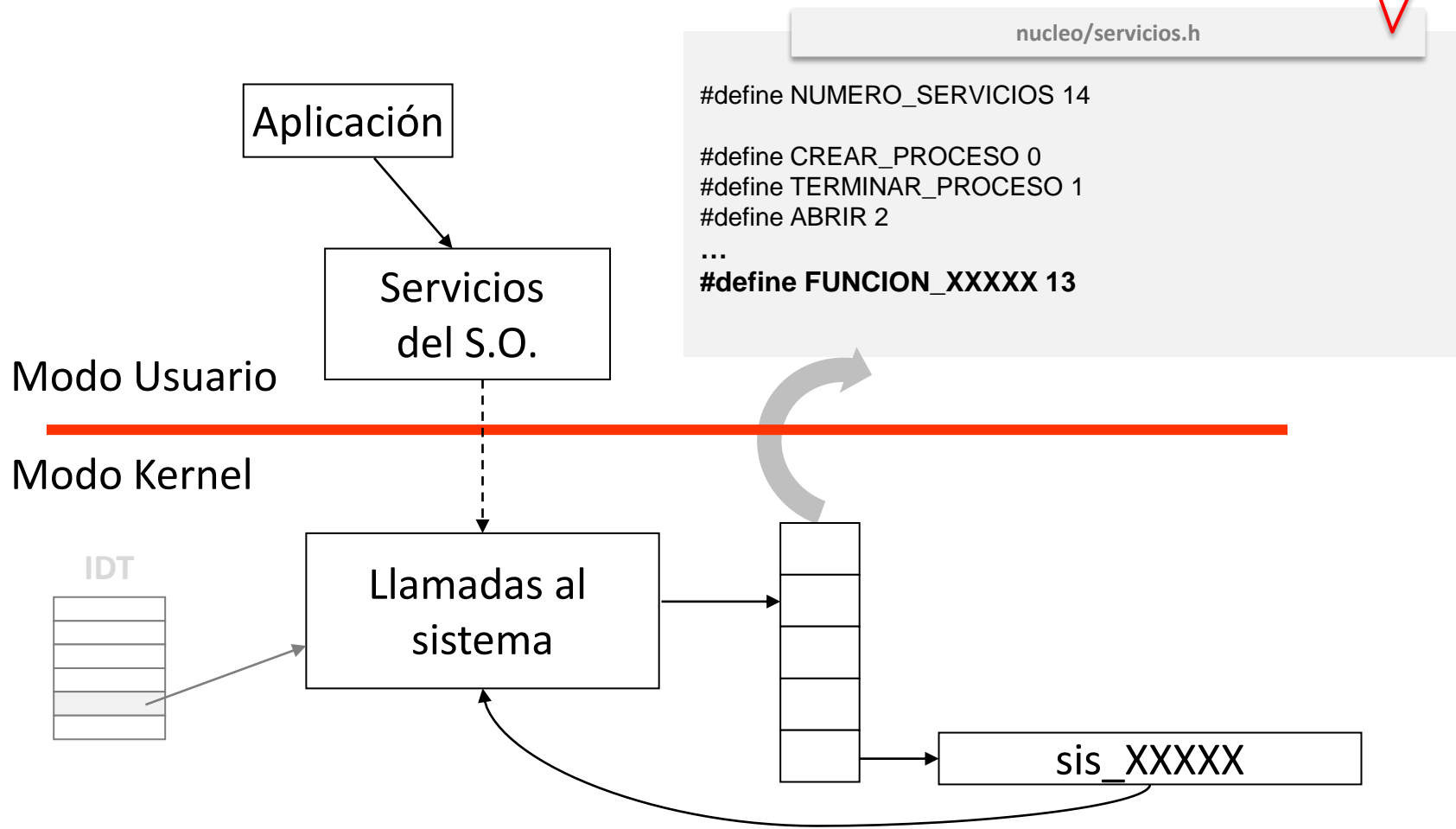
Llamadas al sistema tratamiento (5/9)



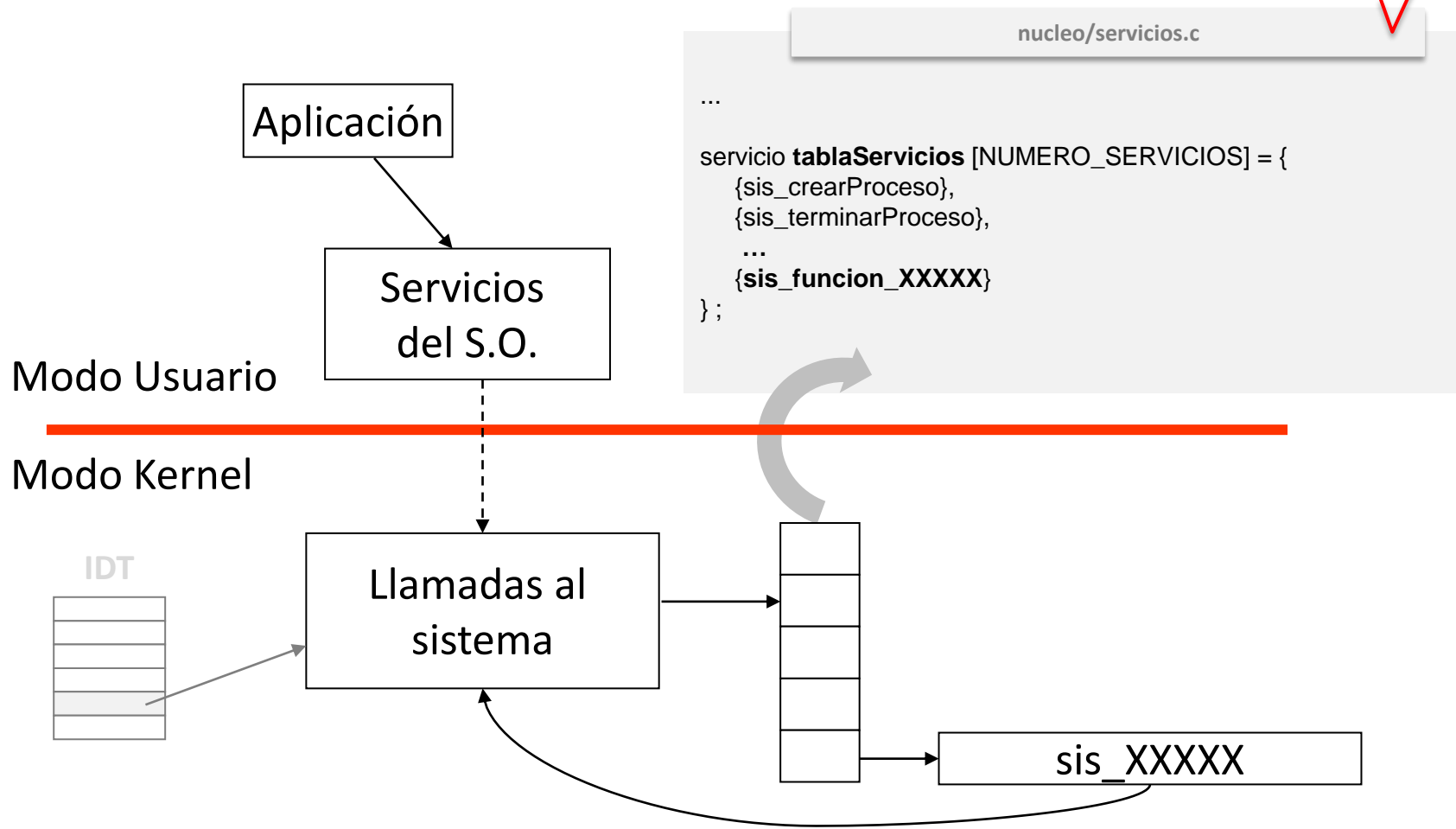
Llamadas al sistema tratamiento (6/9)



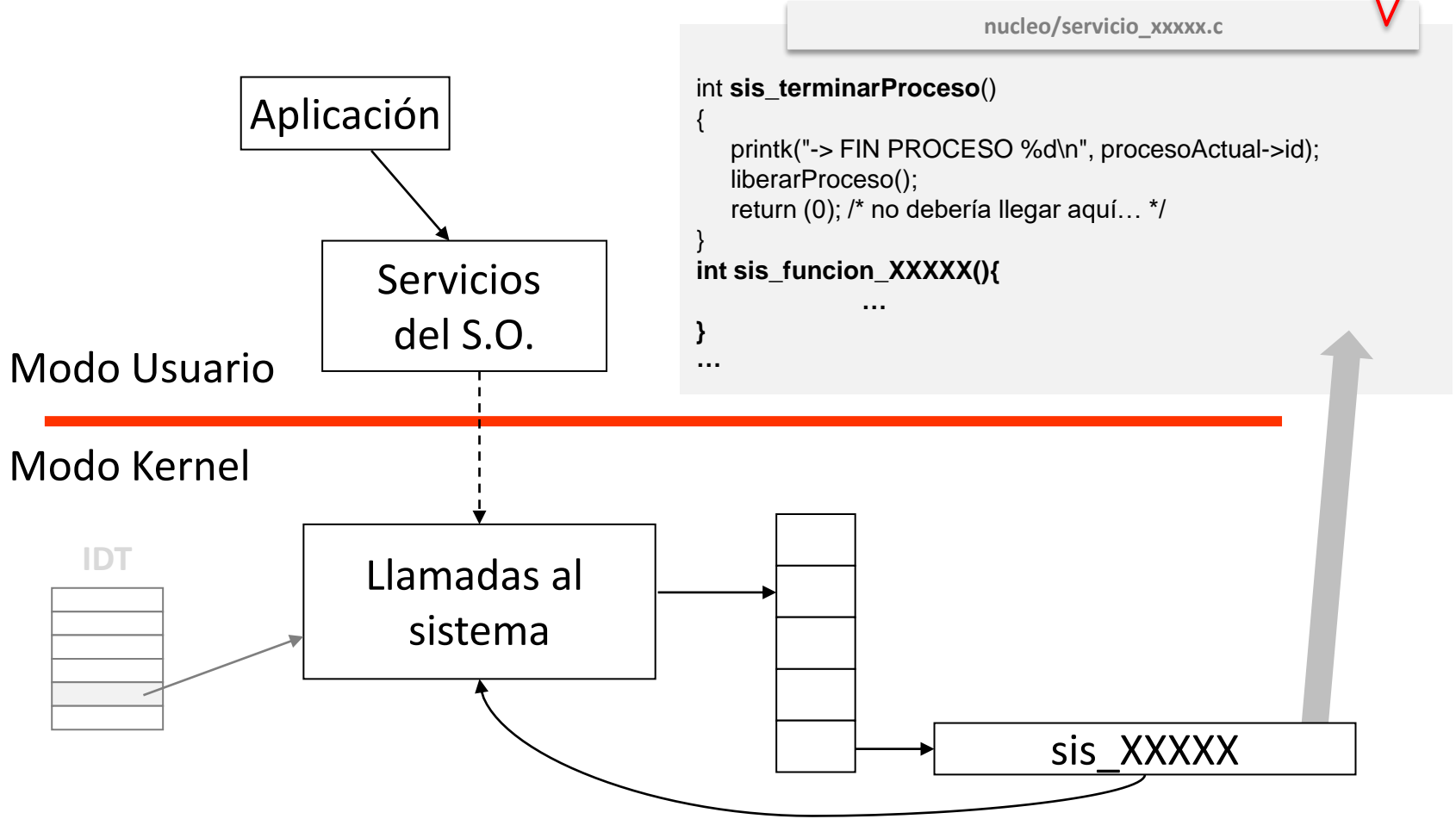
Llamadas al sistema tratamiento (7/9)



Llamadas al sistema tratamiento (8/9)



Llamadas al sistema tratamiento (9/9)



Llamadas al sistema

tratamiento en Linux (1/7)

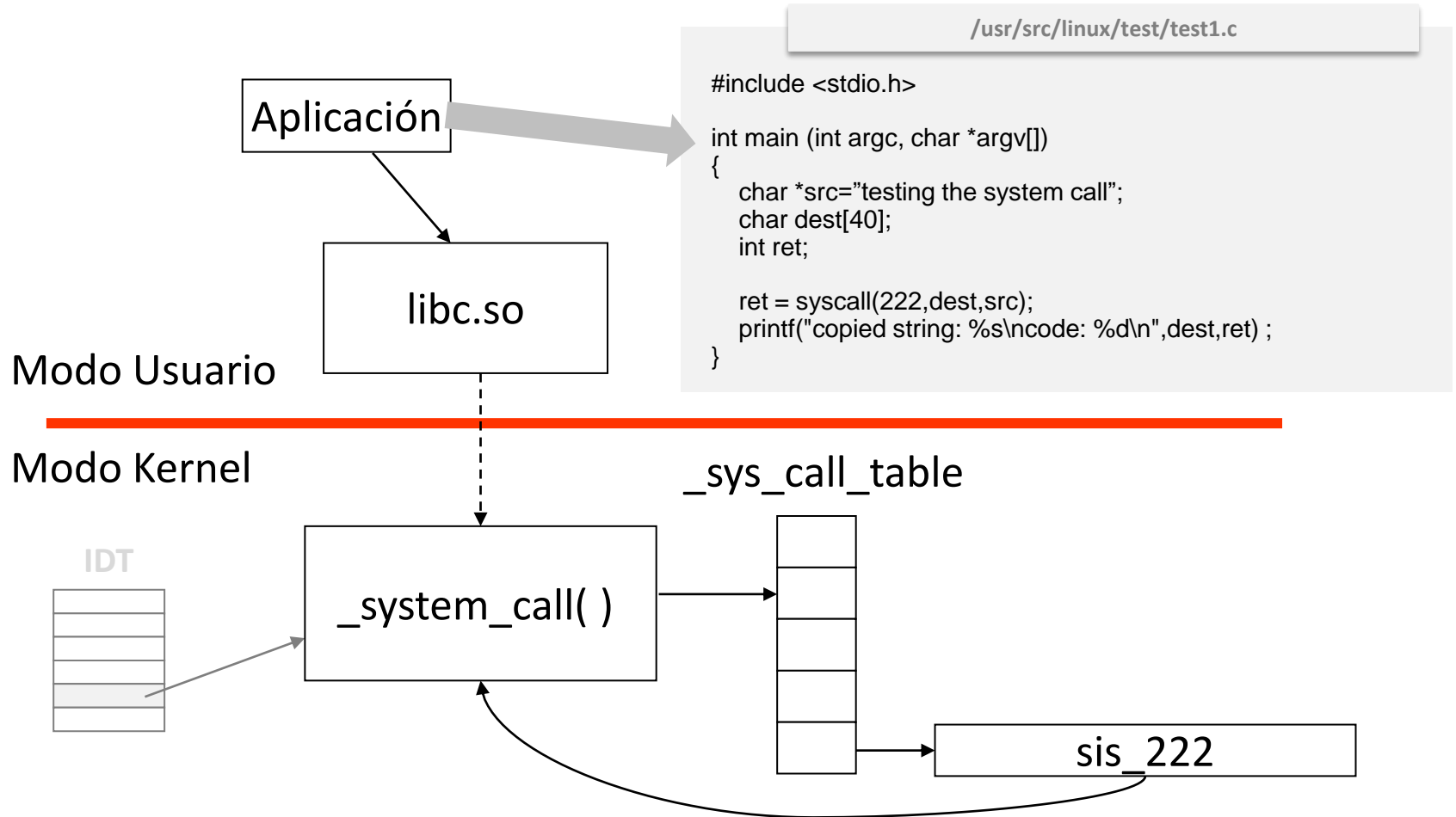
/usr/src/linux/arch/x86/kernel/traps.c

```
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);

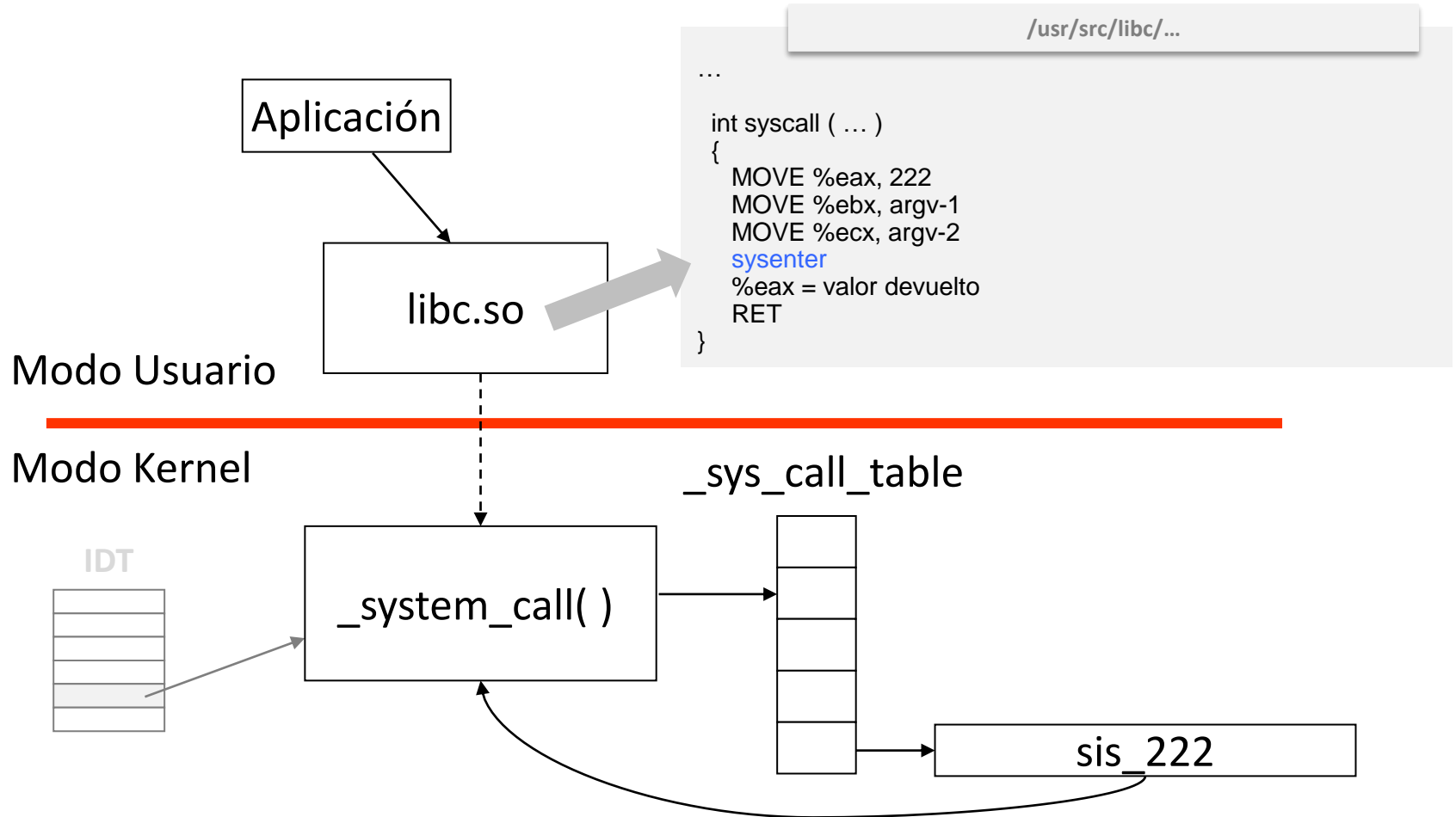
#ifdef CONFIG_IA32_EMULATION
    set_system_intr_gate(IA32_SYSCALL_VECTOR, ia32_syscall);
    set_bit(IA32_SYSCALL_VECTOR, used_vectors);
#endif

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

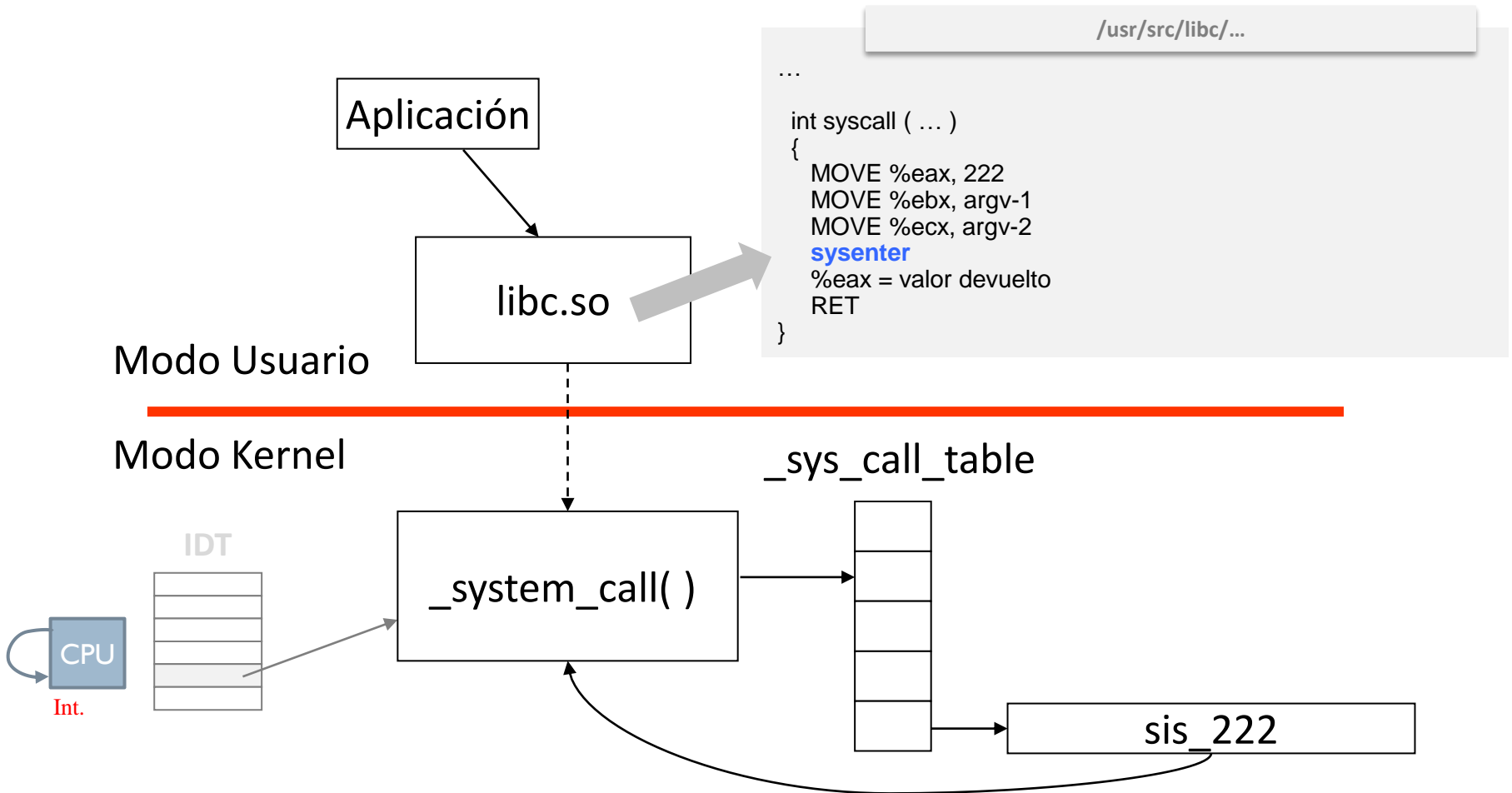
Llamadas al sistema tratamiento en Linux (2/7)



Llamadas al sistema tratamiento en Linux (3/7)

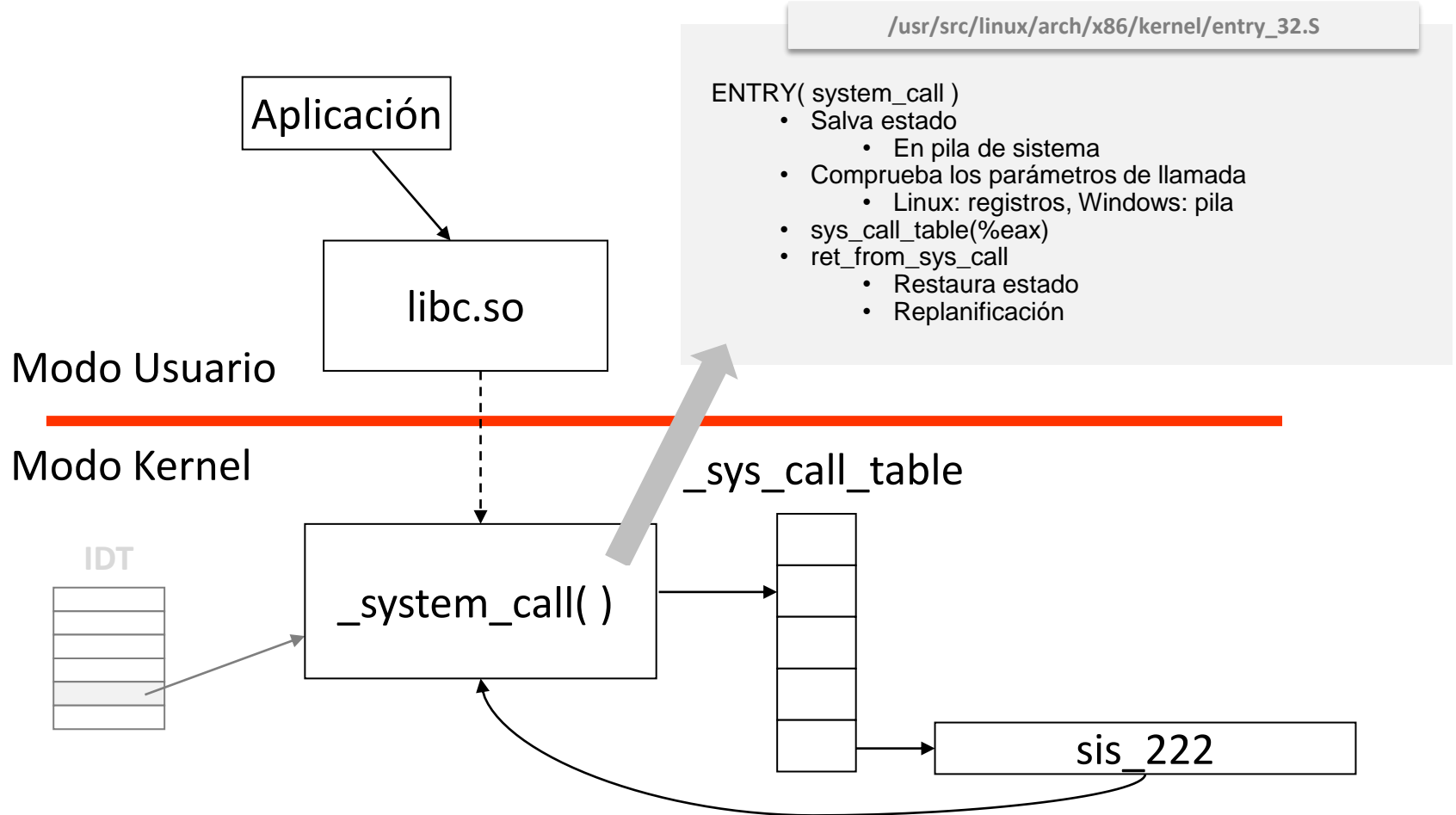


Llamadas al sistema tratamiento en Linux (3/7)



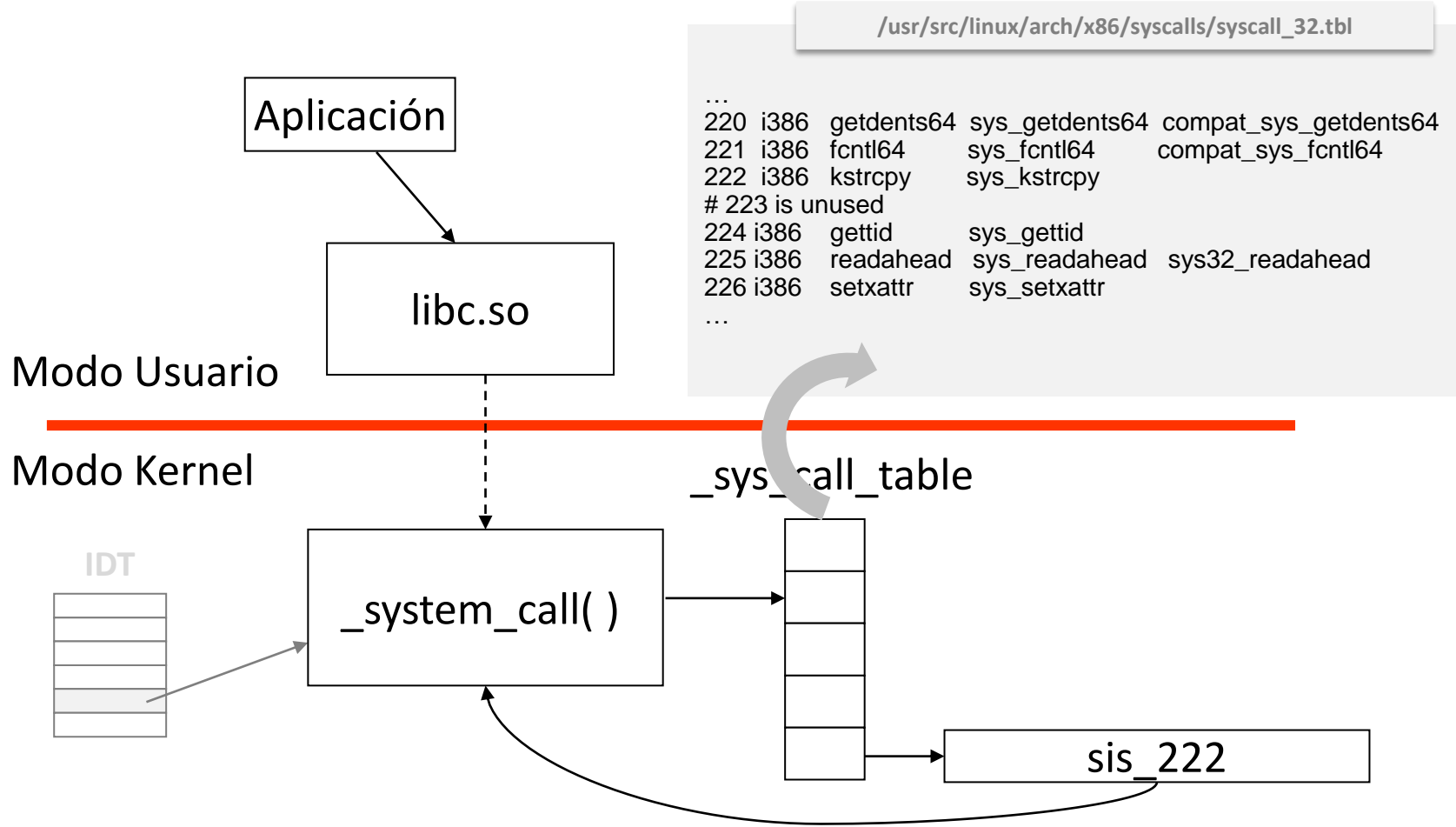
Llamadas al sistema

tratamiento en Linux (4/7)



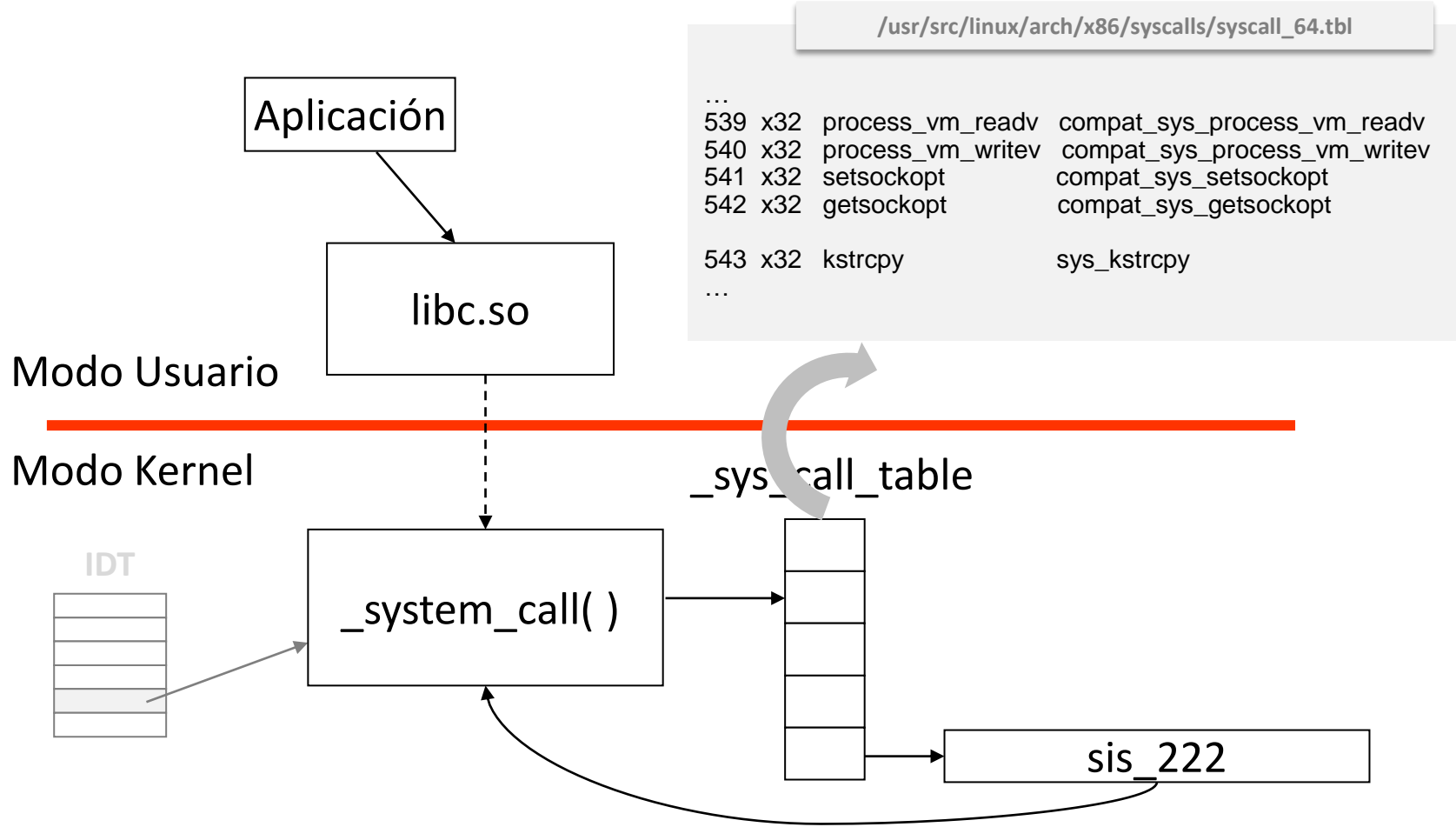
Llamadas al sistema

tratamiento en Linux (5/7)

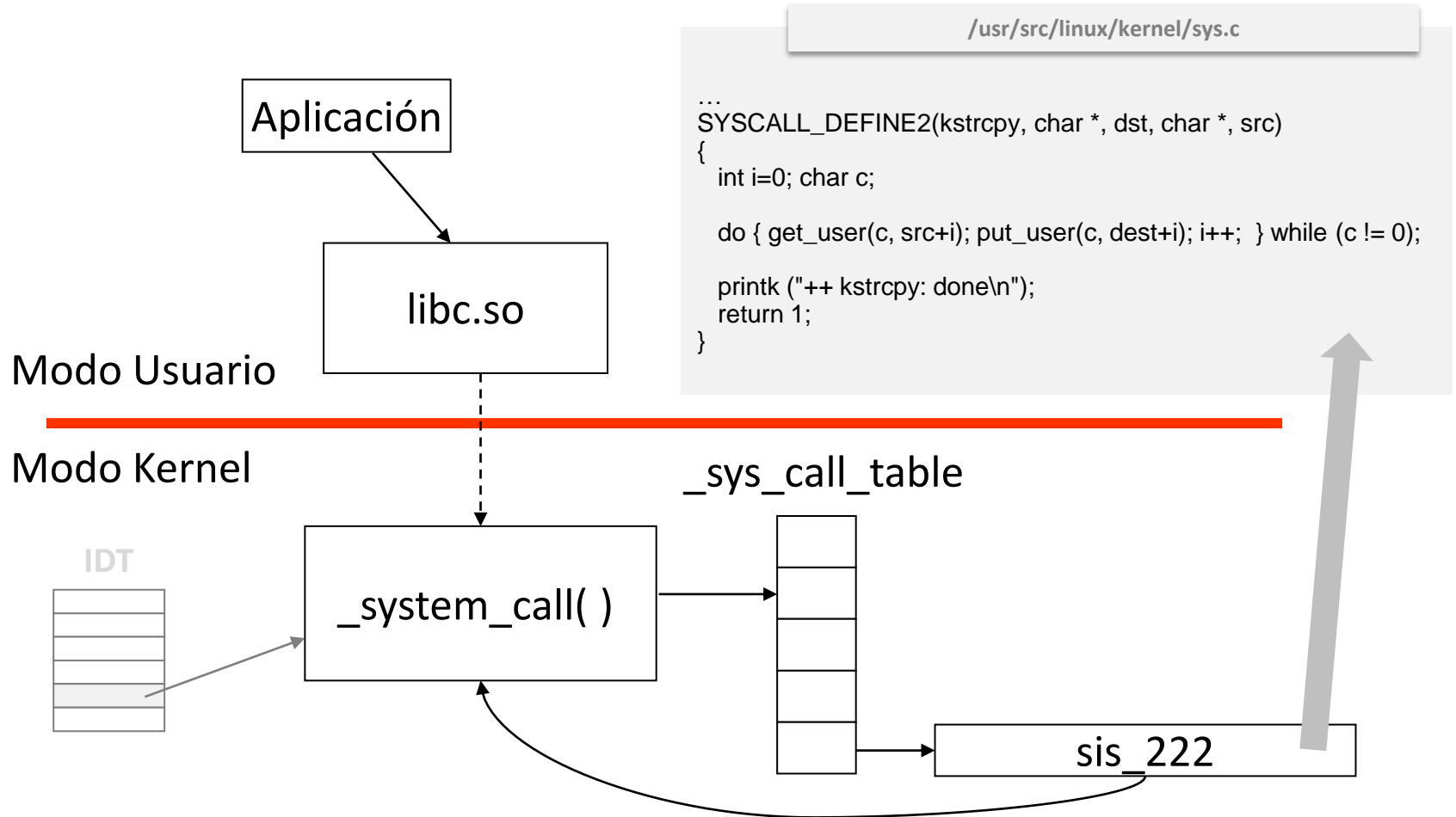


Llamadas al sistema

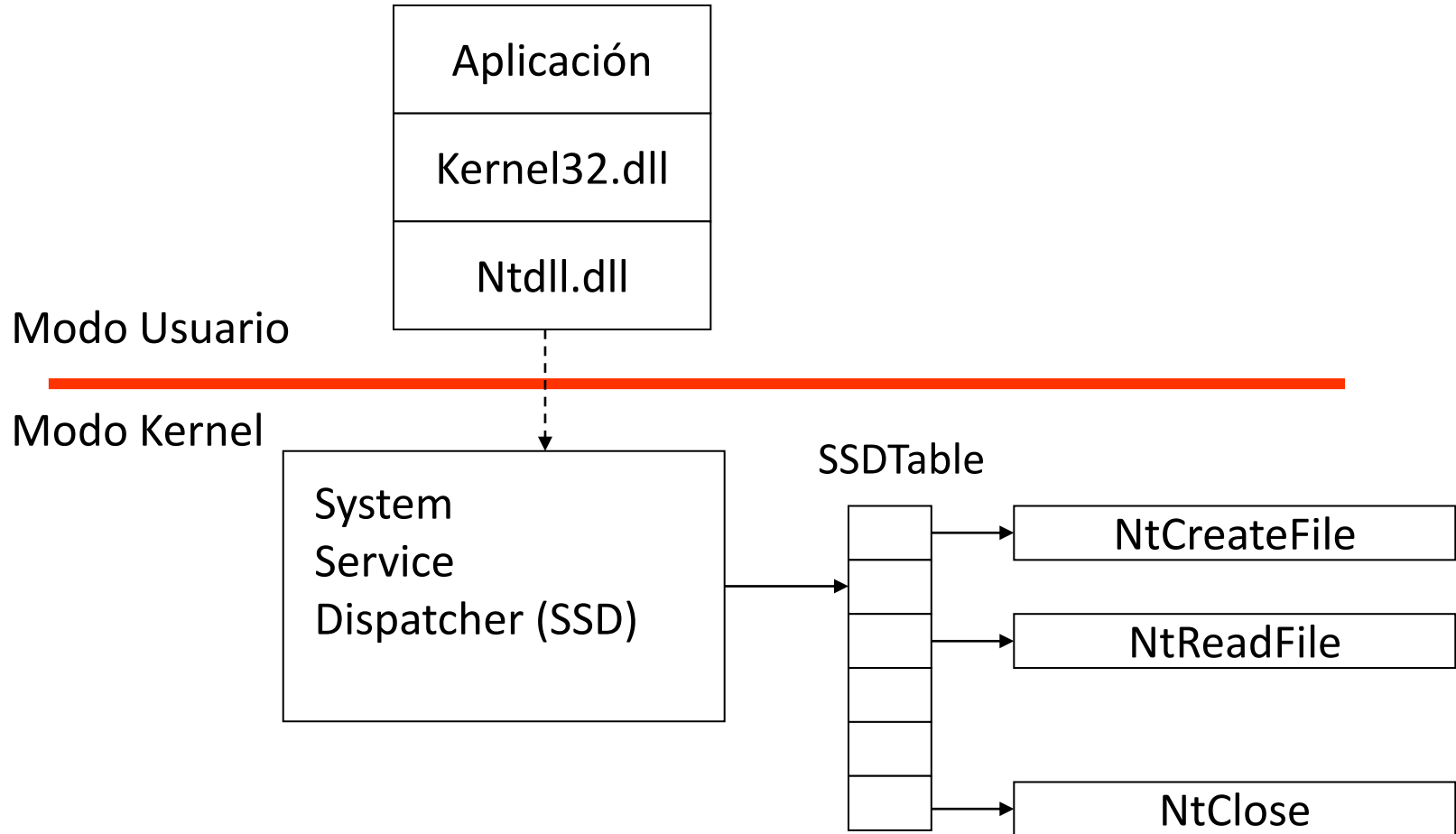
tratamiento en Linux (6/7)



Llamadas al sistema tratamiento en Linux (7/7)

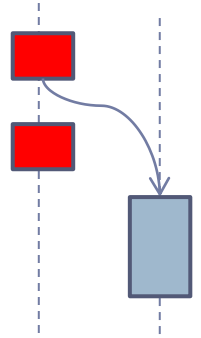


Llamadas al sistema tratamiento en Windows



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);

    ...
}
```

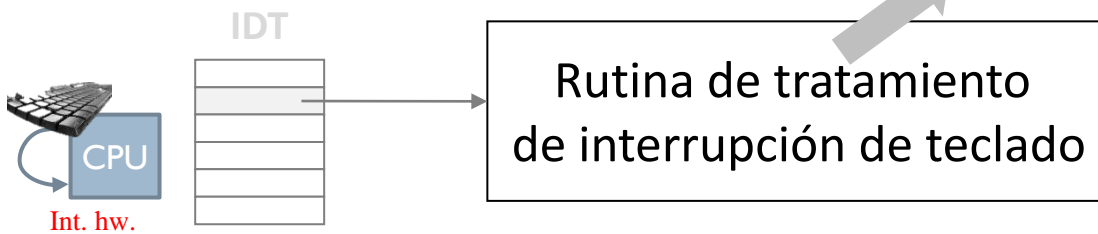
Interrupción hardware

tratamiento (1/2)

Modo Usuario

Modo Kernel

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



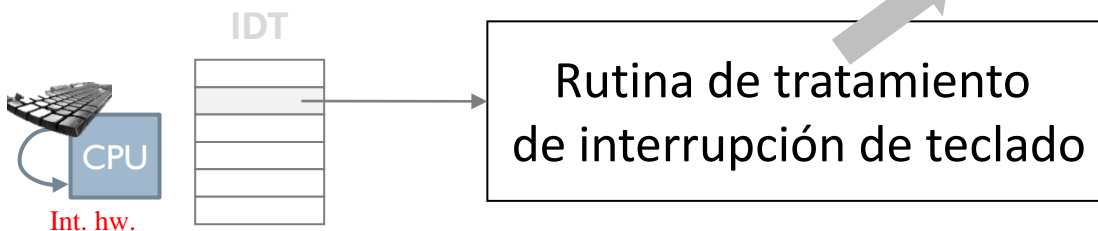
Interrupción hardware

tratamiento (1/2)

Modo Usuario

Modo Kernel

```
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 (1/2)

Modo Usuario

Modo Kernel

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



Rutina de tratamiento
de interrupción software

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

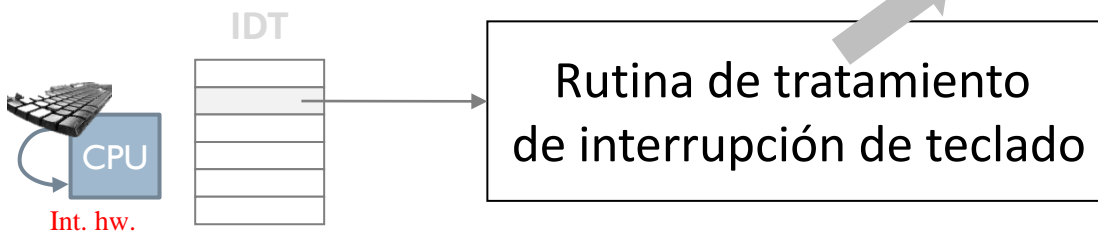
Interrupción hardware

tratamiento (2/2)

Modo Usuario

Modo Kernel

```
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)

Modo Usuario

Modo Kernel

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

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

IDT



Rutina de tratamiento
de interrupción software

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 1ª 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.

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

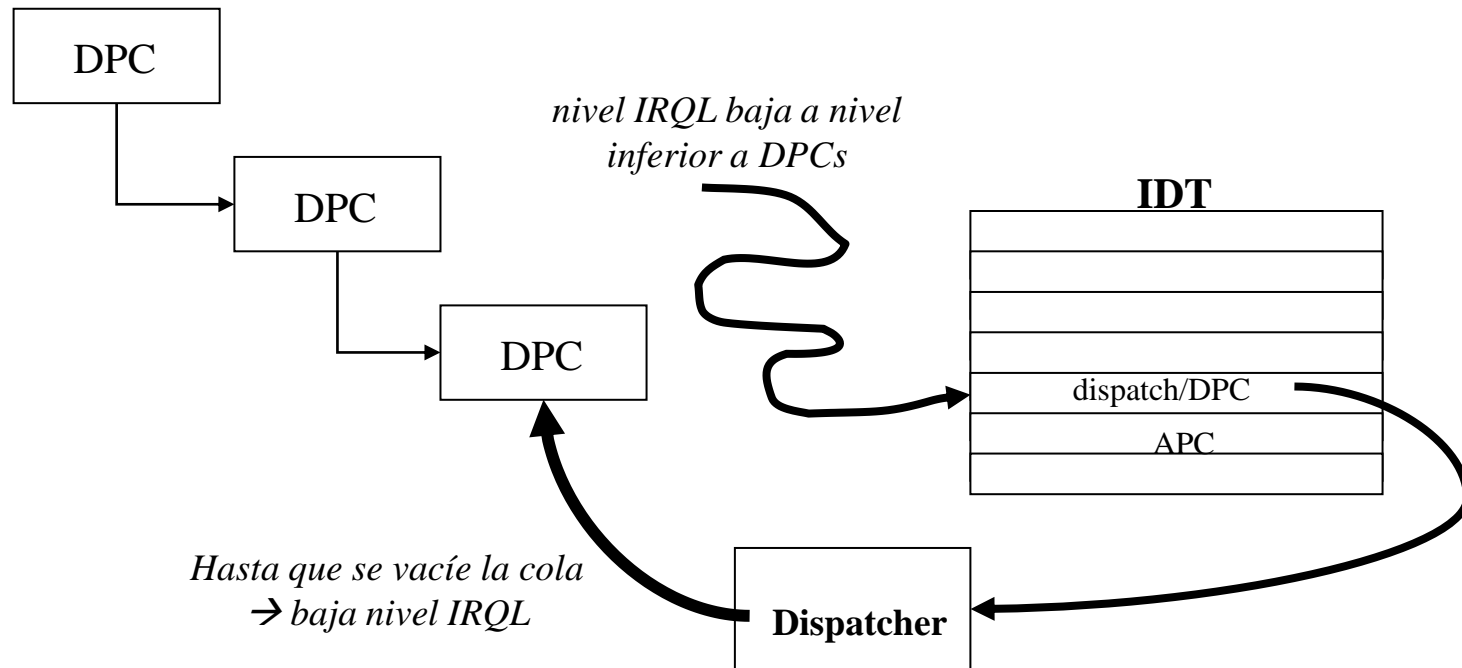
Interrupción software

tipos de tratamiento en Windows: las DPC

Usuario

Kernel

Cola de DPCs *objects* (p.ej., código a ejecutar): única por procesador:



Contenidos

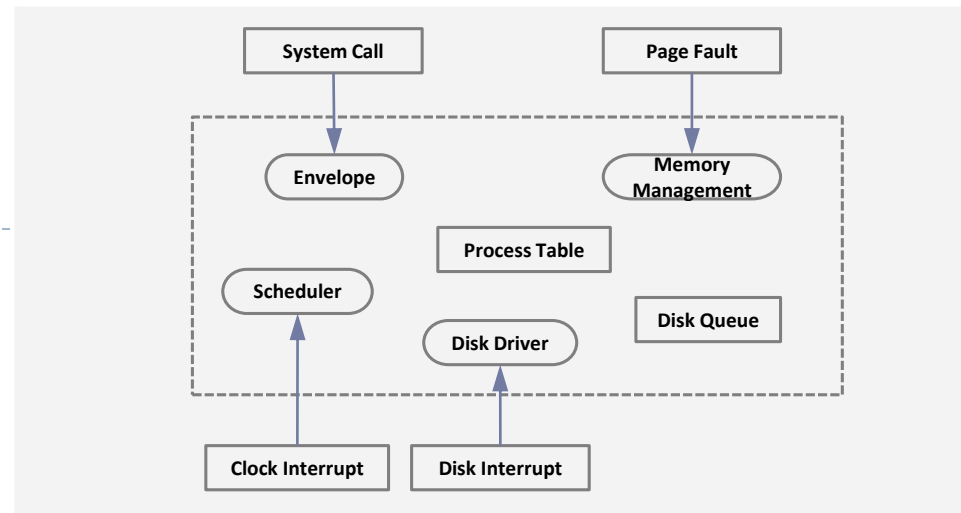
- ▶ **Introducción**

- ▶ **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**

Procesos de núcleo

Ejemplo en Linux

- ▶ kworker, ksoftirqd, irq, rcuob, rcuos, watchdog, ...

PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
1	root	20	0	34100	3484	1500	S	0,0	0,0	0:00.98	init
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:00.12	ksoftirqd/0
5	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworker/0:0H
7	root	20	0	0	0	0	S	0,0	0,0	0:14.27	rcu_sched
8	root	20	0	0	0	0	S	0,0	0,0	0:08.35	rcuos/0
9	root	20	0	0	0	0	S	0,0	0,0	0:05.92	rcuos/1
10	root	20	0	0	0	0	S	0,0	0,0	0:06.10	rcuos/2
11	root	20	0	0	0	0	S	0,0	0,0	0:06.28	rcuos/3
12	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcu_bh
13	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcuob/0
14	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcuob/1
15	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcuob/2
16	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcuob/3
17	root	rt	0	0	0	0	S	0,0	0,0	0:00.29	migration/0
18	root	rt	0	0	0	0	S	0,0	0,0	0:00.10	watchdog/0
19	root	rt	0	0	0	0	S	0,0	0,0	0:00.10	watchdog/1
20	root	rt	0	0	0	0	S	0,0	0,0	0:00.19	migration/1
21	root	20	0	0	0	0	S	0,0	0,0	0:00.32	ksoftirqd/1
22	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kworker/1:0
23	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworker/1:0H
24	root	rt	0	0	0	0	S	0,0	0,0	0:00.09	watchdog/2
25	root	rt	0	0	0	0	S	0,0	0,0	0:00.25	migration/2
...											

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	<ul style="list-style-type: none">• Una CPU solo	<pre>unsigned long flags; local_irq_save(flags); /* ... SC: sección crítica ... */ local_irq_restore(flags);</pre>
Spin Locks	<ul style="list-style-type: none">• Todas las CPU• Espera activa:<ul style="list-style-type: none">• NO se puede dormir, planificar, etc. en SC	<pre>#include <linux/spinlock.h> spinlock_t l1 = SPIN_LOCK_UNLOCKED; spin_lock(&l1); /* ... SC: sección crítica ... */ spin_unlock(&l1);</pre>
Mutex	<ul style="list-style-type: none">• Todas las CPU• Espera bloqueante:<ul style="list-style-type: none">• NO usar en int. HW	<pre>#include <linux/mutex.h> static DEFINE_MUTEX(m1); mutex_lock(&m1); /* ... SC: sección crítica ... */ mutex_unlock(&m1);</pre>
Operaciones Atómicas	<ul style="list-style-type: none">• Todas las CPU	<pre>atomic_t a1 = ATOMIC_INIT(0); atomic_inc(&a1); printk("%d\n", atomic_read(&a1));</pre>



Ejemplo de mecanismos compuestos...

Linux



Técnica	Ámbito	Ejemplo de esqueleto
RW locks	<ul style="list-style-type: none">• Todas las CPU• Espera activa:<ul style="list-style-type: none">• NO se puede dormir, planificar, etc. en SC	<pre>rwlock_t x1 = RW_LOCK_UNLOCKED; read_lock(&x1); /* ... SC: sección crítica ... */ read_unlock(&x1); write_lock(&x1); /* ... SC: sección crítica ... */ write_unlock(&x1);</pre>
Spin Locks + irq	<ul style="list-style-type: none">• Todas las CPU• Espera activa y no interrup.:<ul style="list-style-type: none">• NO se puede dormir, planificar, etc. en SC	<pre>spinlock_t l1 = SPIN_LOCK_UNLOCKED; unsigned long flags; spin_lock_irqsave(&l1, flags); /* ... SC: sección crítica ... */ spin_unlock_irqrestore(&l1, flags);</pre>
RW locks + irq	<ul style="list-style-type: none">• Todas las CPU• Espera activa y no interrup.:<ul style="list-style-type: none">• NO se puede dormir, planificar, etc. en SC	<pre>read_lock_irqsave(); read_lock_irqrestore(); write_lock_irqsave(); write_lock_irqrestore();</pre>



Ejecución anidada de tratamiento de evento

Evento en ejecución	Evento que llega	Tratamiento habitual
Int. Hw. / Excepción	Int. Hw. / Excepción	<ul style="list-style-type: none">• Se permite todas, ninguna o solo de más prioridad (si S.C., deshabilitadas).
Ll. sist. / Int. Sw.	Int. Hw. / Excepción	<ul style="list-style-type: none">• Interrumpe siempre (si S.C., deshabilitadas).
Int. Hw. / Excepción	Ll. sist. / Int. Sw.	<ul style="list-style-type: none">• No pueden interrumpirlas.
Ll. sist. / Int. Sw.	Ll. sist. / Int. Sw.	<ul style="list-style-type: none">• Kernel no expulsivo<ul style="list-style-type: none">• No pueden interrumpir (se encolan).• Muchos UNIX y Linux antes.• Kernel expulsivo.<ul style="list-style-type: none">• 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

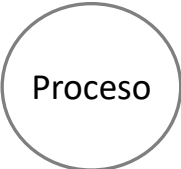


Contenidos

- ▶ **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...

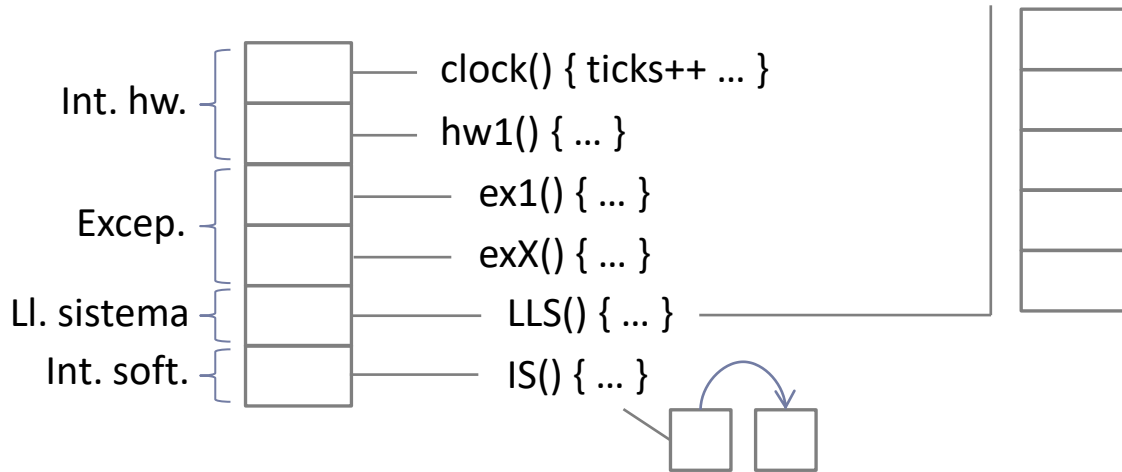
Funcionamiento interno del núcleo
repartido entre: **interrupciones software, llamadas al sistema, excepciones e interrupciones hardware**



system_lib

U

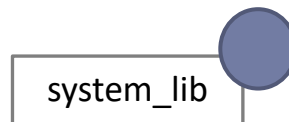
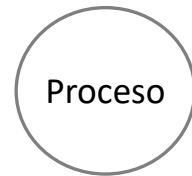
K



Periférico

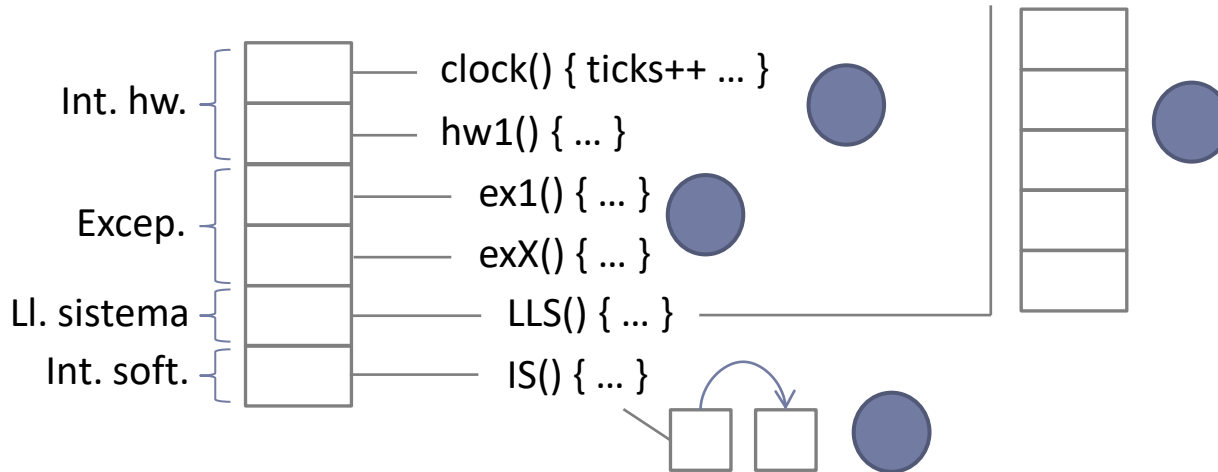
Contexto...

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



U

K



Periférico

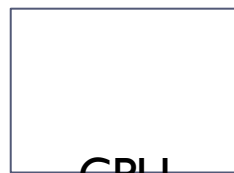
Ejemplo...

App.

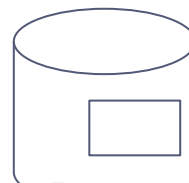
- `char buffer[1024];`
 - ...
 - `read(fd,buffer)`
 - `buffer[2048]='\0';`
-

S.O.
(kernel)

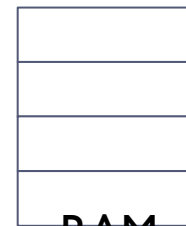
HW.



CPU



Disco



RAM

Ejemplo...

App.

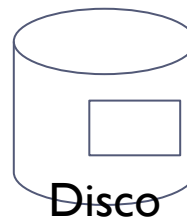
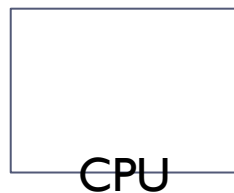
- `char buffer[1024];`
- ...
- `read(fd,buffer)`
- `buffer[2048]='\0';`

Il. sistema

- Pedir bloque
- Ejecutar $P_i + I$

S.O.
(kernel)

HW.



Ejemplo...

App.

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

Il. sistema

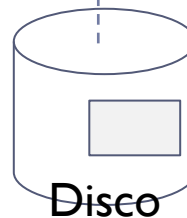
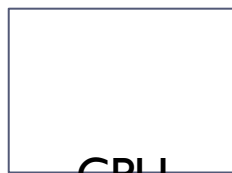
- Pedir bloque
- Ejecutar Pi+I

S.O.
(kernel)

- Copiar a RAM
- Act. int. soft.

int. hw

HW.



Ejemplo...

App.

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

Il. sistema

- Pedir bloque
- Ejecutar Pi+I

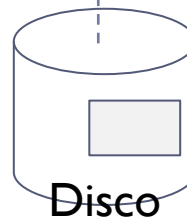
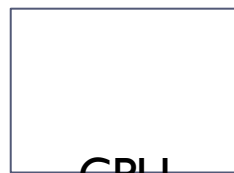
S.O.
(kernel)

- Copiar a RAM
- Act. int. soft. → • Pi listo

int. sw

int. hw

HW.

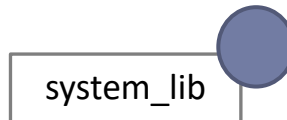
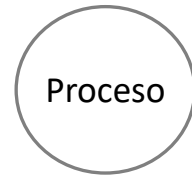




Contexto...

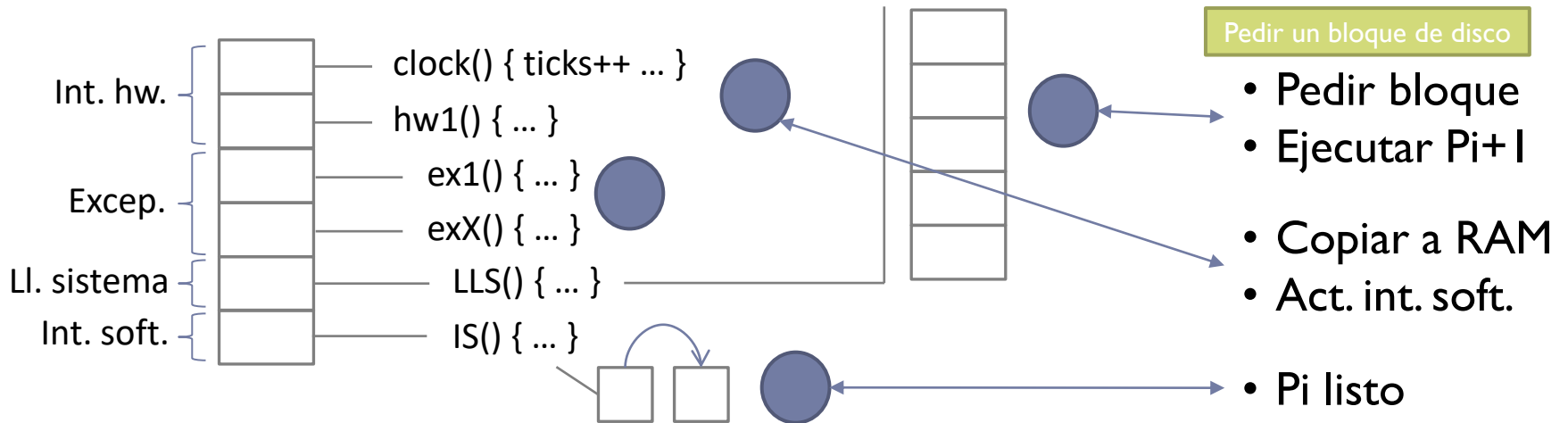
Una funcionalidad es una secuencia de tareas

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



U

K

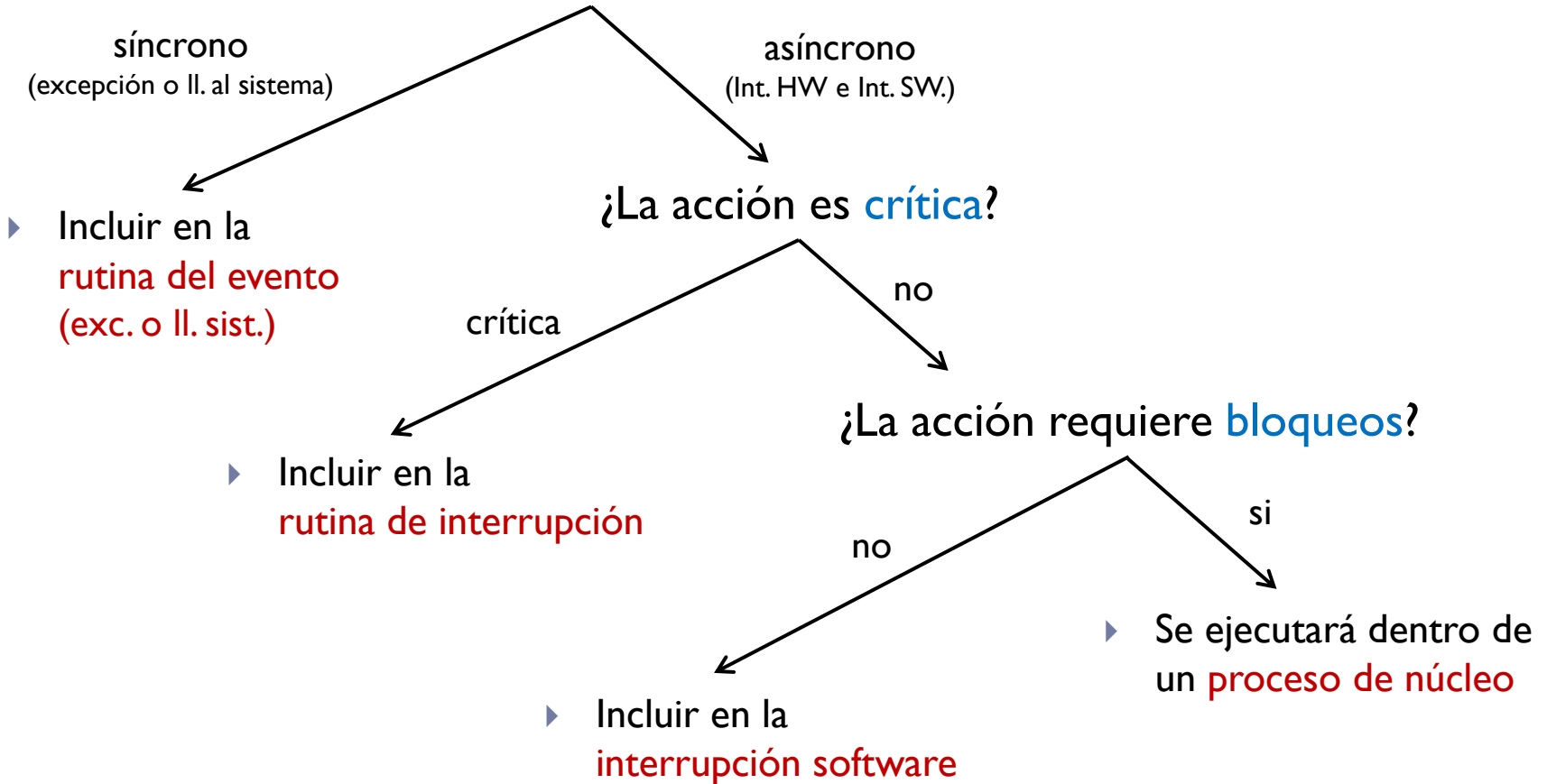


Periférico



Criterios para asignar una tarea a un contexto de ejecución

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



Grupo ARCOS
Departamento de Informática
Universidad Carlos III de Madrid

Lección 2

Funcionamiento del sistema operativo

Diseño de Sistemas Operativos
Grado en Ingeniería Informática y Doble Grado I.I. y A.D.E.

