

Grupo ARCOS
Departamento de Informática
Universidad Carlos III de Madrid

Ejercicios

Introducción

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



Ejercicio

enunciado (1/2)

Se dispone de un sistema hardware que incluye un dispositivo de reloj, el cual genera una interrupción con cada *tick* del reloj.

Un sistema operativo multiproceso tiene prevista (a falta de la implementación) la llamada al sistema:

```
void ObtenerFecha (struct Fecha * r);
```

Que debe devolver la fecha y hora actuales. Para ello, se dispone además del código de la función:

```
struct Fecha * Convertir_Ticks_en_Fechas (int ticks);
```

Dicha función permite obtener la fecha y hora actuales a partir del número de *ticks* que han transcurrido.

Ejercicio

enunciado (2/2)

Se pide:

- ▶ Implementar en pseudocódigo la funcionalidad necesaria del kernel del sistema operativo para gestionar el reloj y ofrecer a los usuarios la funcionalidad de obtener la fecha y hora actuales cuando lo soliciten. Poner especial interés en indicar cuales son las estructuras de datos requeridas o modificadas, la interfaz de las funciones implementadas y los eventos utilizados.

Ejercicio

solución

1. Planteamiento inicial
 1. Estado inicial del sistema
 2. Estudio de qué hay que modificar
2. Responder a las preguntas
3. Revisar las respuestas

Ejercicio

solución

1. Planteamiento inicial

1. Estado inicial del sistema
2. Estudio de qué hay que modificar

2. Responder a las preguntas

3. Revisar las respuestas

Ejercicio

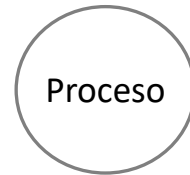
solución

Realicemos un diagrama con el estado inicial del sistema, con los elementos más relevantes para el problema

U

K

Ejercicio solución



En espacio de usuario (U) tenemos los procesos que hacen llamadas al sistema a través de `system_lib` o provocan excepciones, lo que provoca la ejecución del núcleo (K)

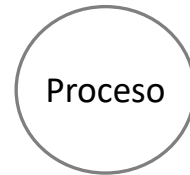
system_lib

U

K

Ejercicio solución

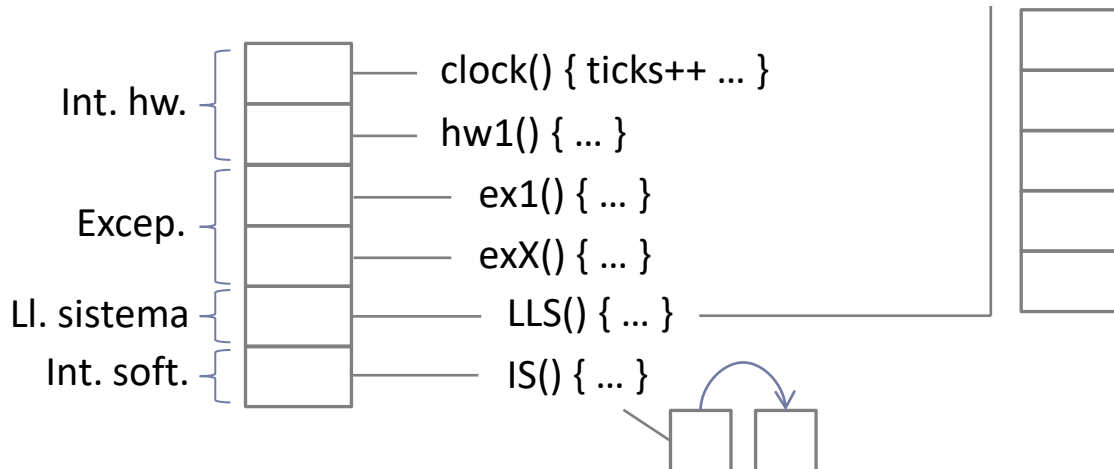
En el tema 2 se introducía el funcionamiento interno del núcleo del sistema operativo: interrupciones software, llamadas al sistema, excepciones e interrupciones hardware



system_lib

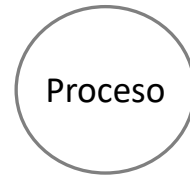
U

K



Ejercicio solución

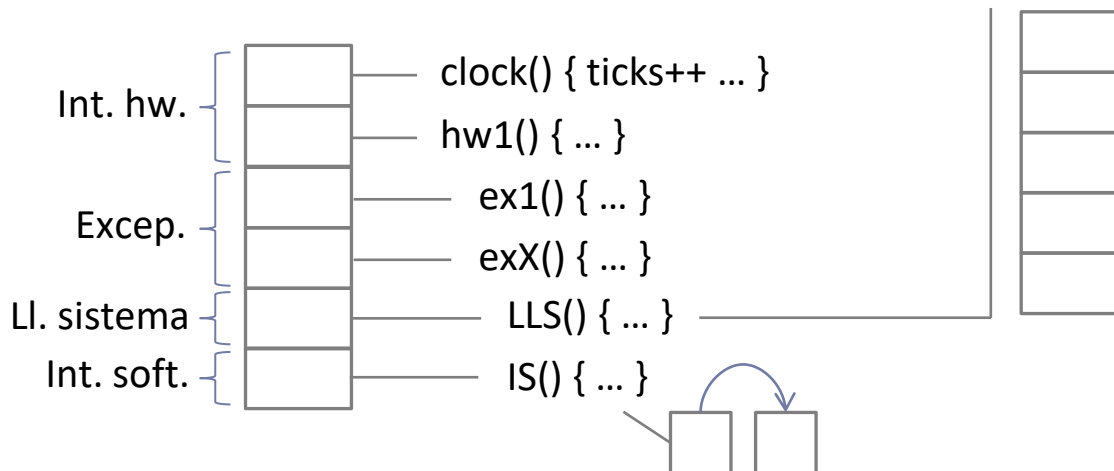
Tenemos en este diagrama el estado inicial del sistema, con los elementos más relevantes para el problema



system_lib

U

K



Ejercicio

solución

1. Planteamiento inicial

1. Estado inicial del sistema

2. Estudio de qué hay que modificar

2. Responder a las preguntas

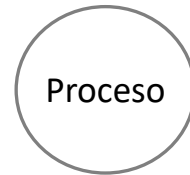
3. Revisar las respuestas



Ejercicio solución

Partimos del sistema existente,
y en el enunciado nos piden:

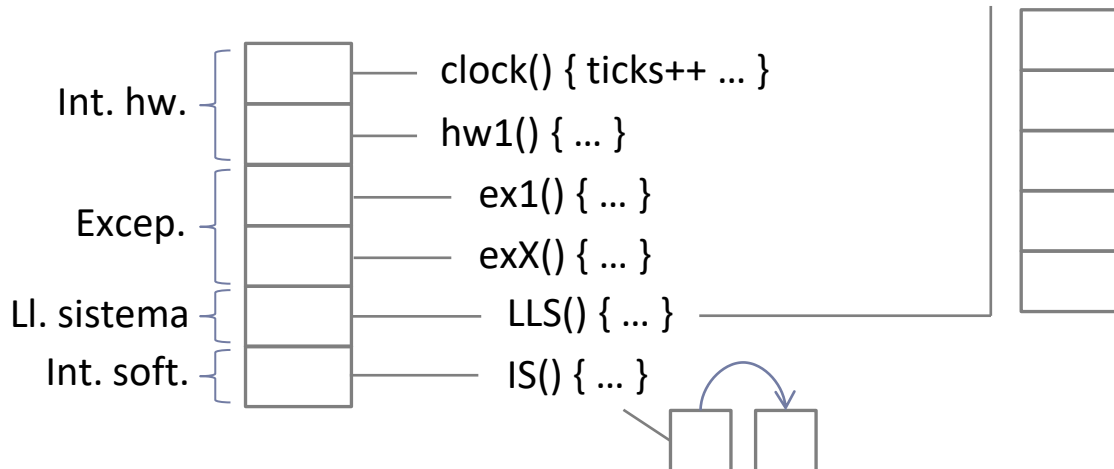
- Añadir una llamada al sistema.
- Usar para dicha llamada una función existente en el *kernel*.



system_lib

U

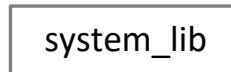
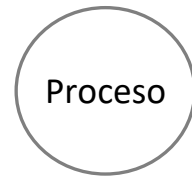
K



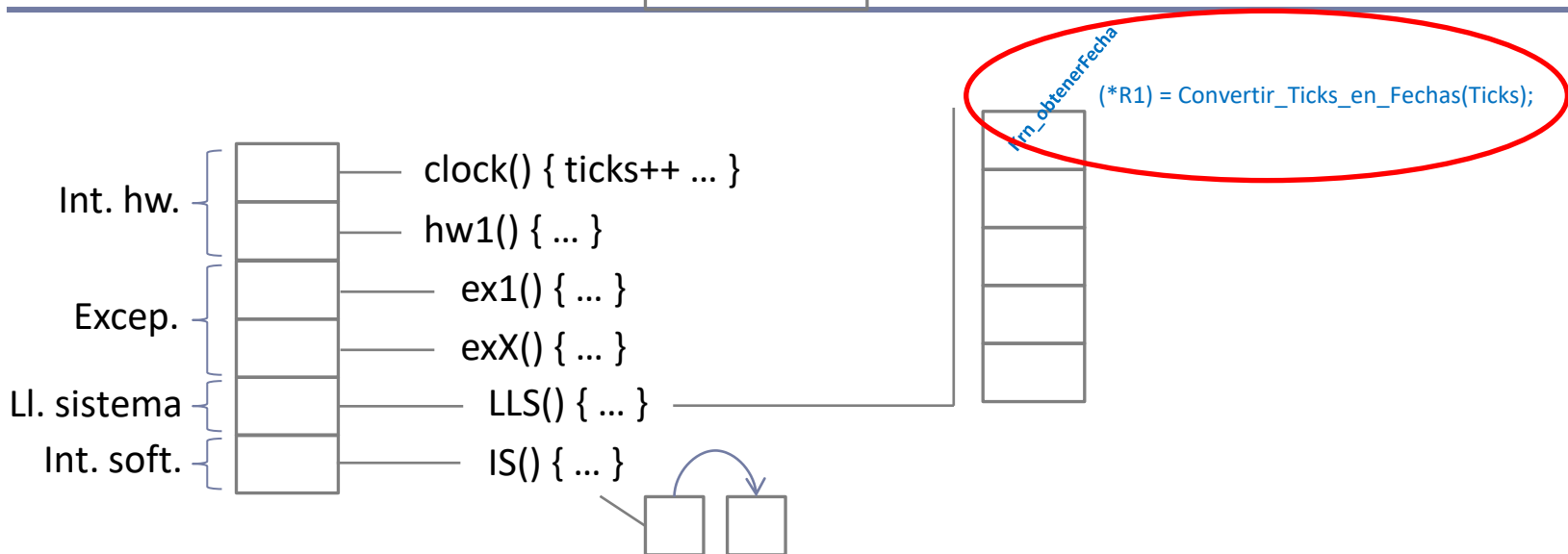
Ejercicio solución

Una llamada al sistema tiene una parte dentro del núcleo y otra parte en el espacio de usuario.

La parte en el núcleo (krn_XX) guarda el resultado en el registro R0.

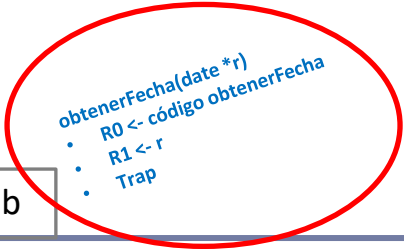
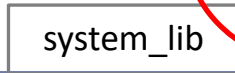
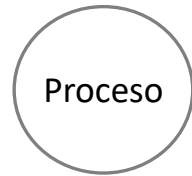


U
K



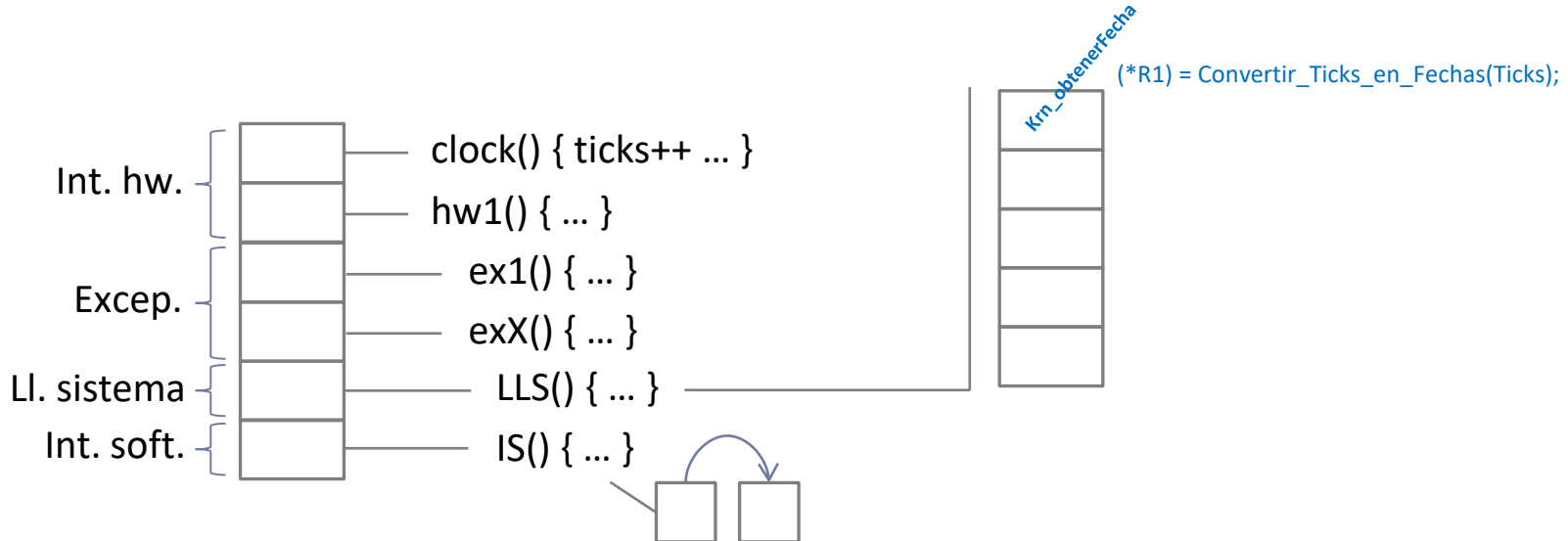
Ejercicio solución

La parte en espacio de usuario (XX) indica el código en R0, pasa a núcleo y devuelve el valor del registro R0.



U

K



Ejercicio solución

Finalmente repasamos el enunciado para comprobar que todos y cada uno de los requisitos los cumplimos en el planteamiento.

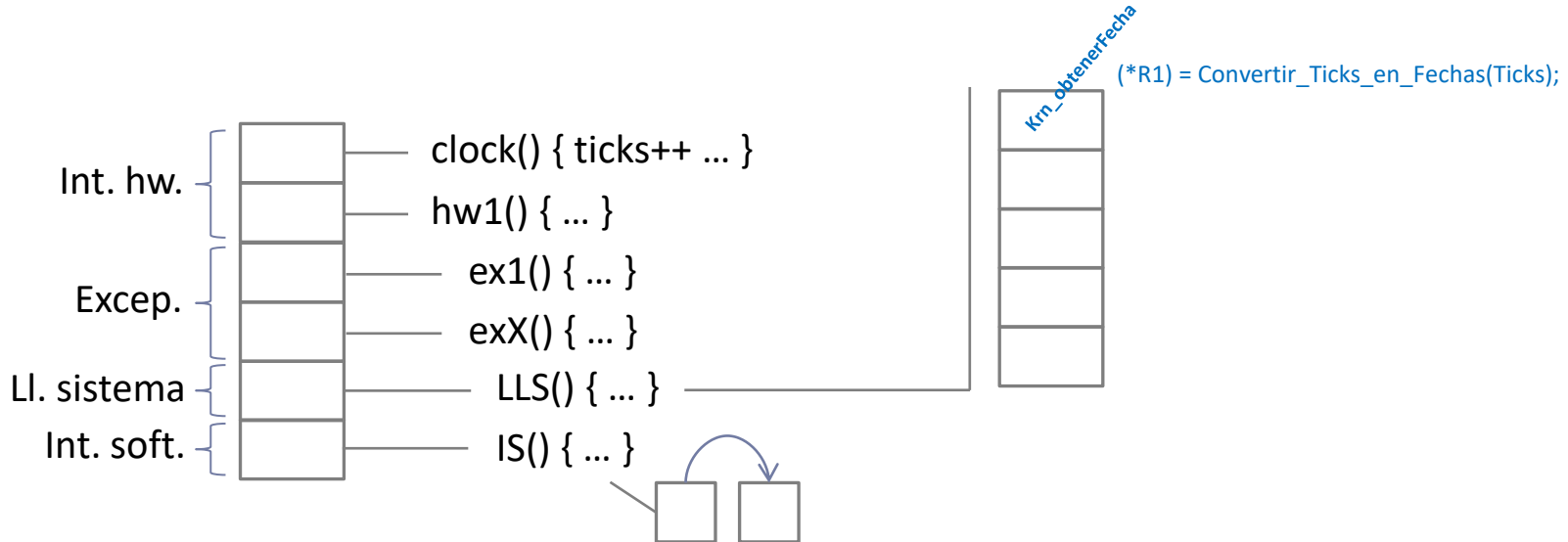


system_lib

```
obtenerFecha(date *r)  
• RO <- código obtenerFecha  
• R1 <- r  
• Trap
```

U

K



Ejercicio

solución

1. Planteamiento inicial

1. Estado inicial del sistema
2. Estudio de qué hay que modificar

2. Responder a las preguntas

3. Revisar las respuestas

Ejercicio

solución

Mirando el planteamiento realizado,
contestamos a las preguntas

Estructuras de datos:

- Ticks: Números de *ticks* de reloj desde el arranque de la máquina.

Ejercicio

solución

Funciones en el núcleo (*kernel*):

Manejador_interrupción_reloj ():

- Ticks = Ticks + 1;

struct Fecha * KERNEL_ObtenerFecha ():

- (*RI) = Convertir_Ticks_en_Fechas (Ticks) ;

Ejercicio

solución

Funciones en espacio de usuario:

void USER_ObtenerFecha (struct Fecha *r):

- R0 = OBTENER_FECHA
- R1 = r
- Trap

Ejercicio

solución

1. Planteamiento inicial

1. Estado inicial del sistema
2. Estudio de qué hay que modificar

2. Responder a las preguntas

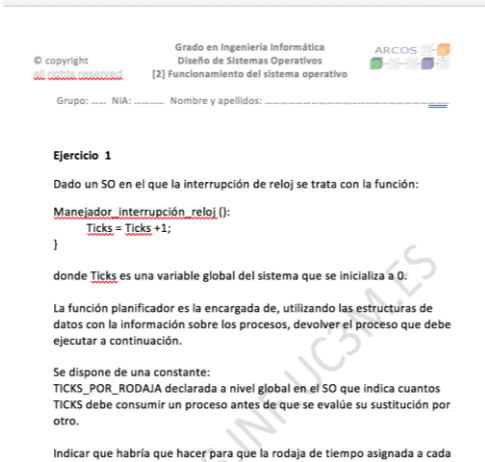
3. Revisar las respuestas

Fallos típicos



- 1) Contestar a la primera pregunta de un apartado únicamente.
- 2) Contestar a otra pregunta de la pedida.
- 3) Contestar a más de lo que se pide:
 - 1) Si está mal la parte extra, puede que se evalúe ...

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: <code>TICKS_POR_RODAJA</code> declarada a nivel global en el SO que indica cuantos <code>TICKS</code> 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>	

Grupo ARCOS
Departamento de Informática
Universidad Carlos III de Madrid

Ejercicios

Introducción

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

