Grupo ARCOS
Departamento de Informática
Universidad Carlos III de Madrid

Ejercicios

drivers y servicios ampliados

Diseño de Sistemas Operativos

Grado en Ingeniería Informática y
Doble Grado I.I. y A.D.E.

cc BY NC SA

Ejercicio
enunciado (1/2)

Una compañía nos contrata para hacer el driver de una tarjeta digitalizadora de sonido. Nos pide para un sistema operativo básico (como el que está siendo visto en clase) que usa un kernel monolítico con planificación Round-Robin que cumpla la siguiente interfaz de usuario:

- Funciones “open(nombre, flags) -> descriptor” y “close(descriptor)” para establecer acceso exclusivo a la tarjeta o liberarla.
- Función “read(descriptor, buffer, tamaño) -> bytes leídos” para leer un buffer con música digitalizada en formato mp3.

Ejercicio
enunciado (2/2)

Se pide:

a) Indicar la interfaz completa que tendrá el driver, así como las estructuras de datos que se necesiten.

b) Indicar los eventos involucrados así como implementar en pseudocódigo dichos eventos, minimizando el número de copias en memoria de los datos hasta llegar al proceso de usuario que los solicitó.

Ejercicio
solución

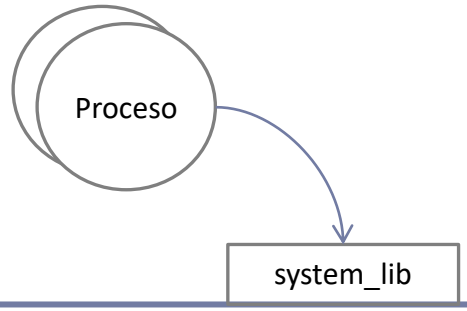
1. Planteamiento inicial
    1. Estado inicial del sistema
    2. Estudio de qué hay que modificar
2. Responder a las preguntas
3. Revisar las respuestas

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

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)

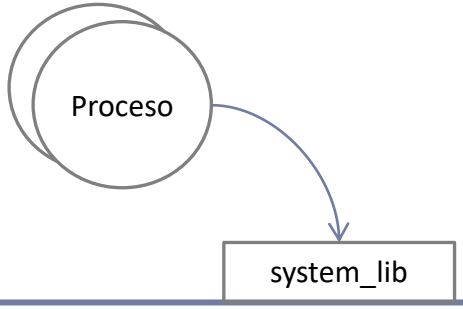


U

K

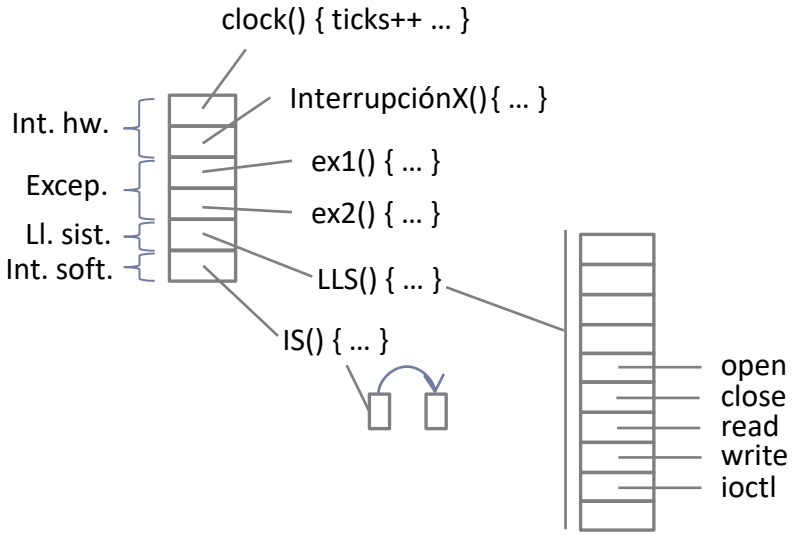
Ejercicio
solución

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



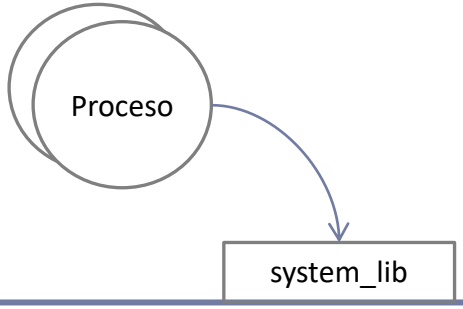
U

K



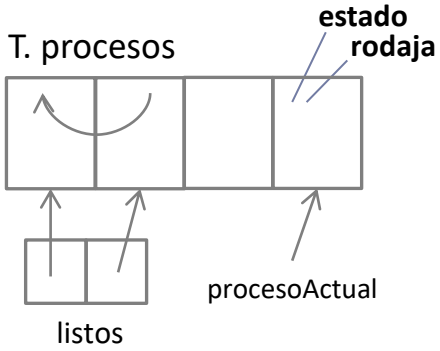
Ejercicio
solución

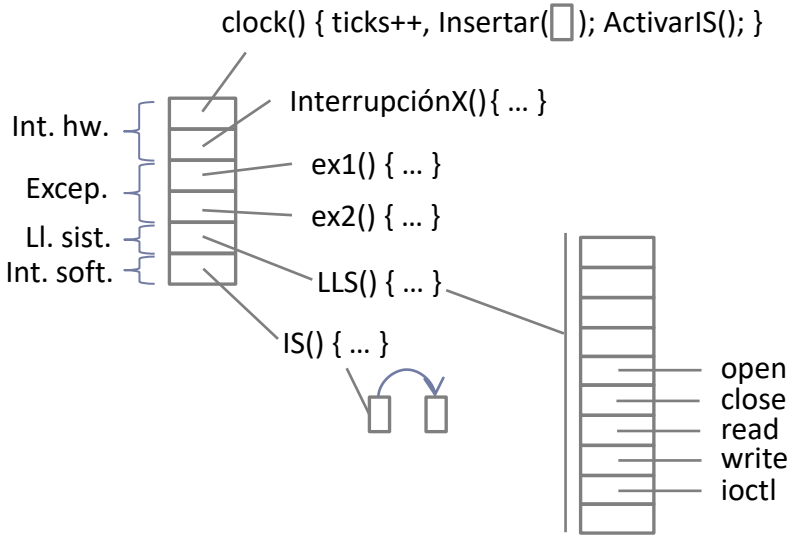
En el tema 3 se introducía las estructuras y funciones internas para la gestión de procesos, como la tabla de procesos, la cola de listos para ejecutar, el planificador, etc.



U

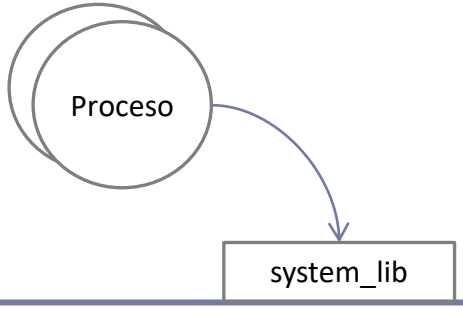
K





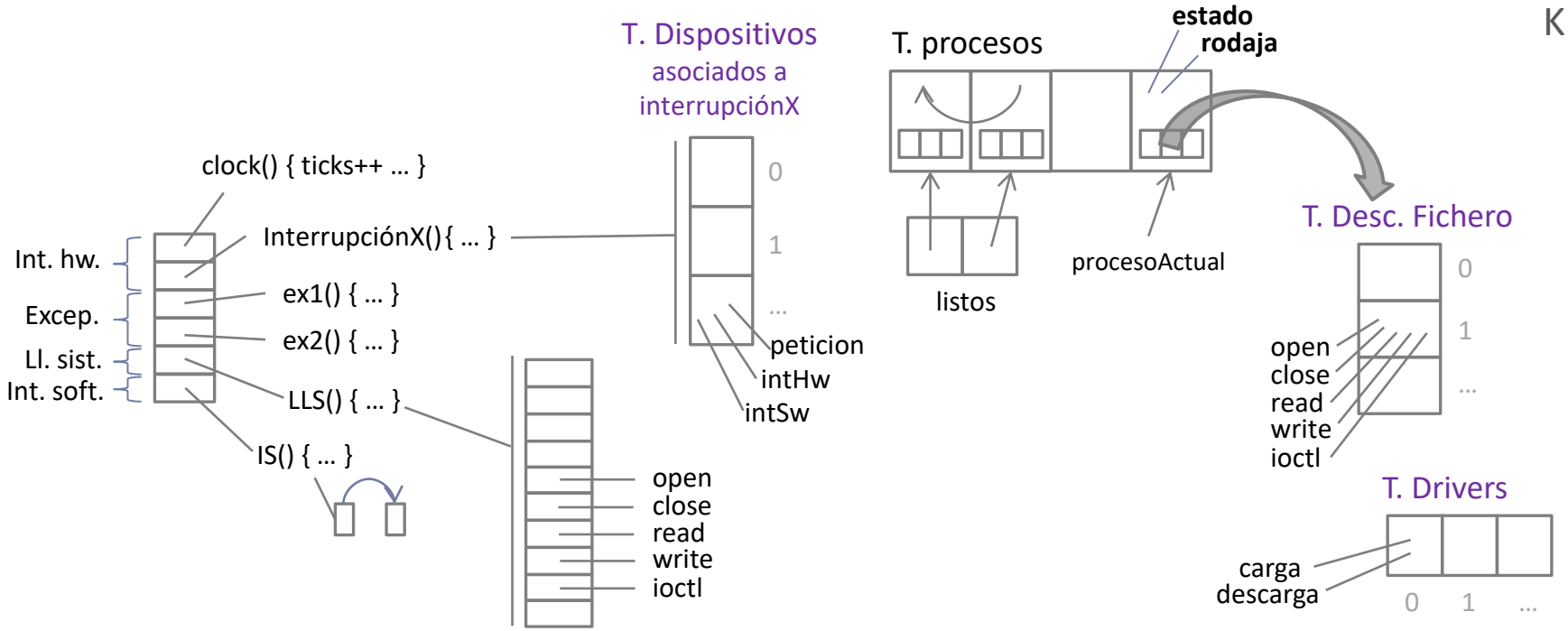
Ejercicio
solución

En el tema 4 añadimos tres tablas:
- Dispositivos: asociados a interrupciónX.
- Desc. de fichero: interfaz del dispositivo (1 tabla por proceso, en cada BCP).
- Drivers: carga y descarga.



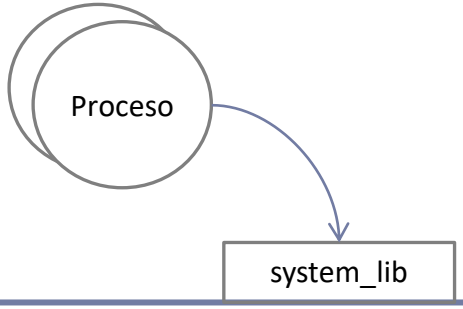
U

K



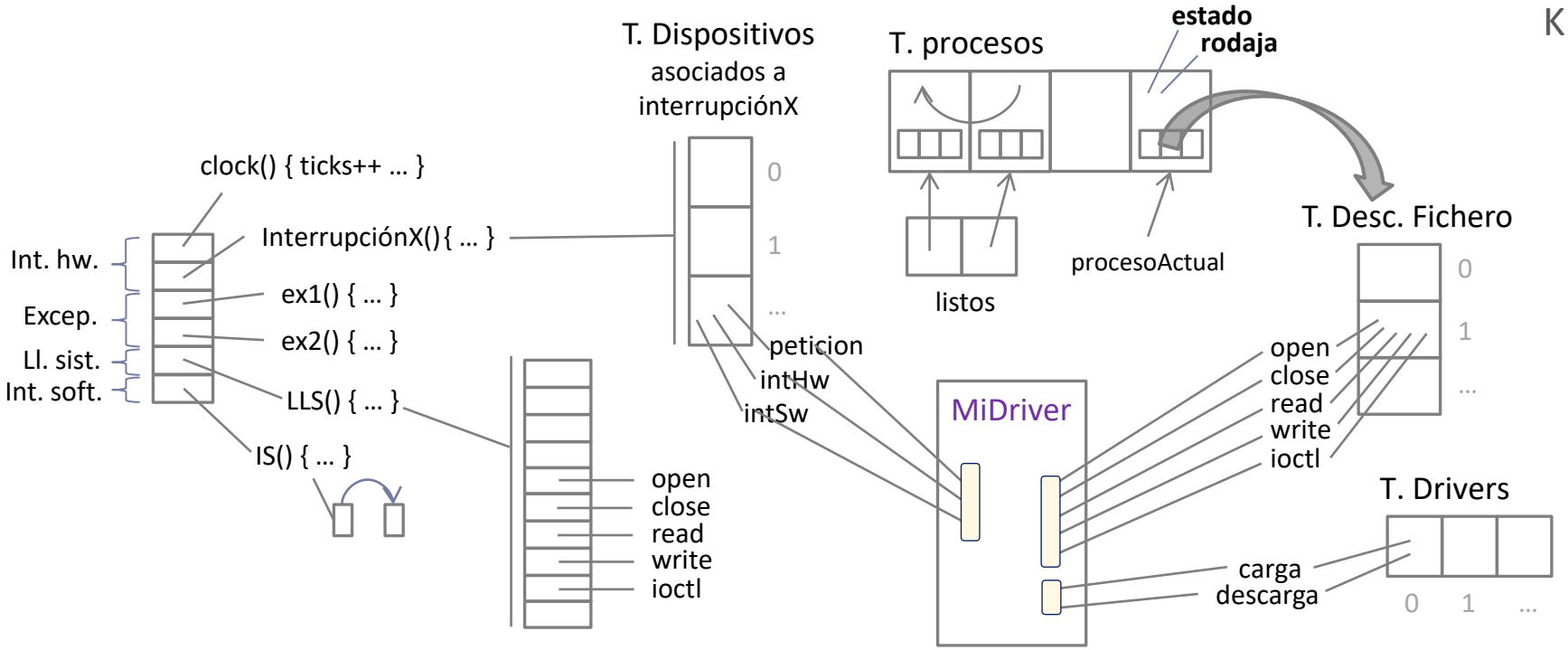
Ejercicio
solución

En el tema 4 creamos un driver como un fichero con, al menos, tres conjuntos de funciones.
Cada conjunto está asociado a una de las tres tablas comentadas antes.



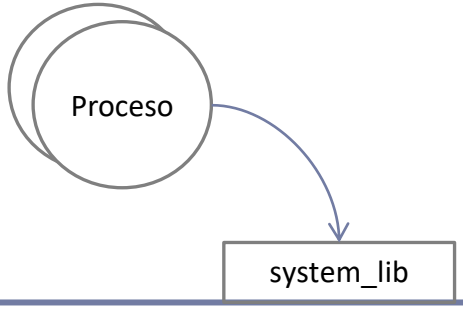
U

K



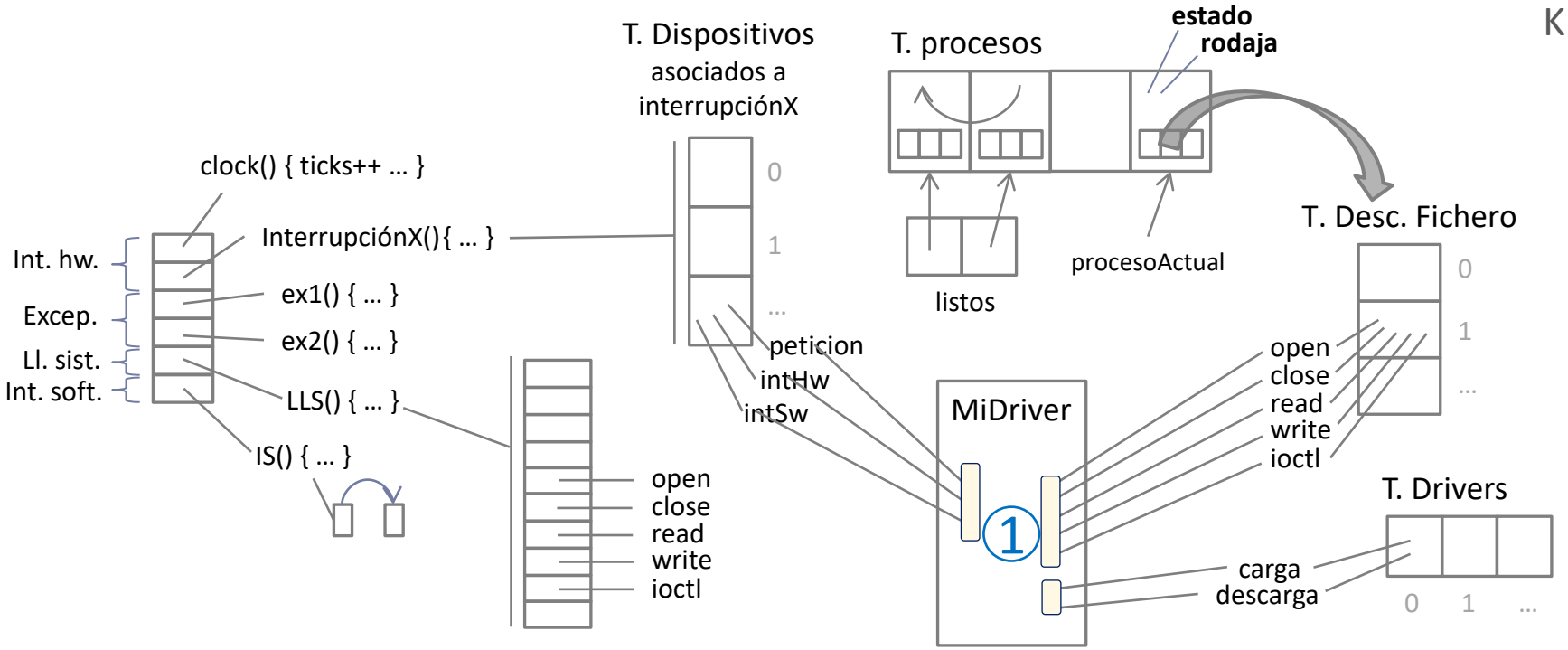
Ejercicio
solución

Apartado a)
Hay que detallar los tres conjuntos de funciones a implementar en el driver de teclado.



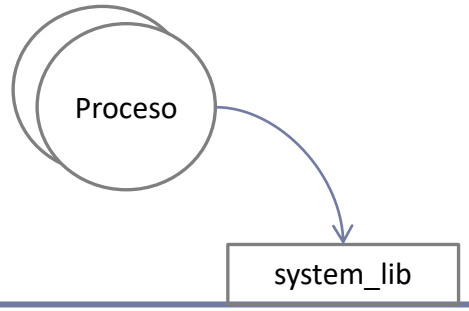
U

K



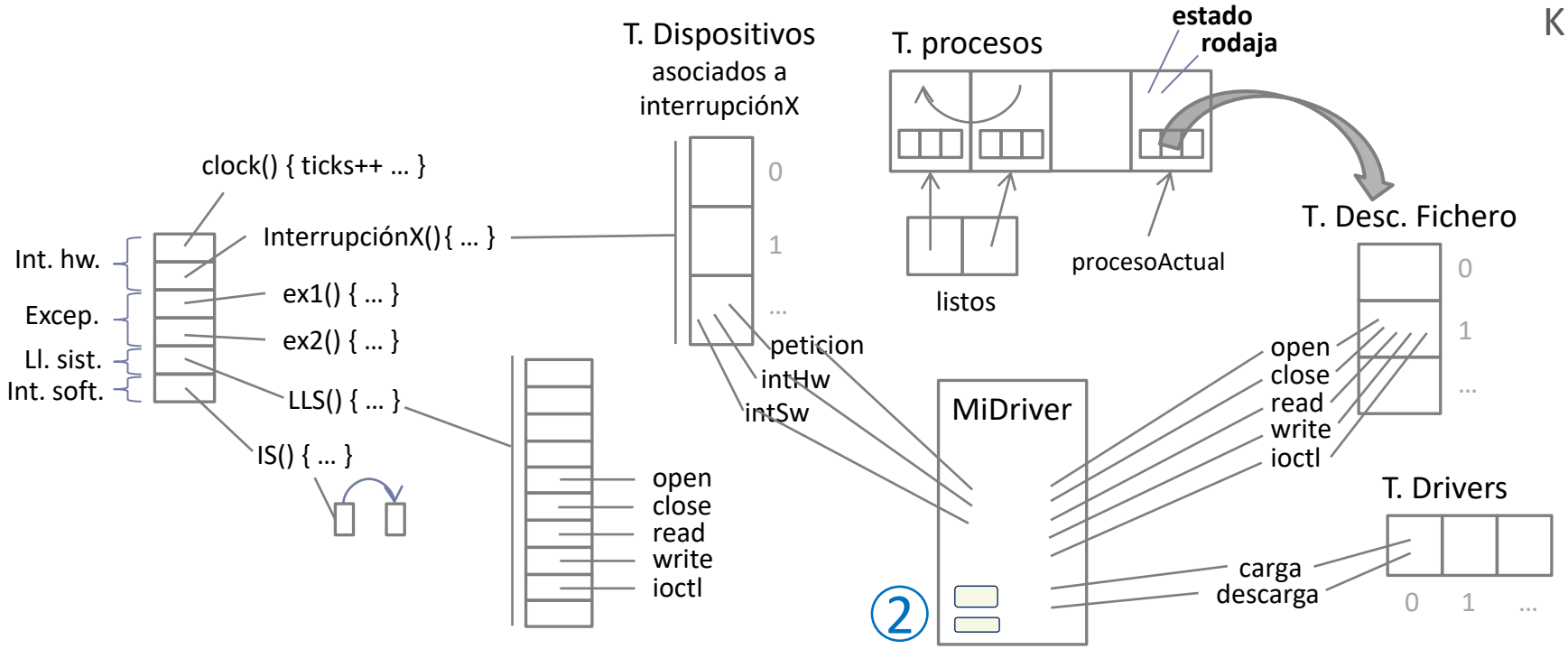
Ejercicio
solución

Apartado a)
Y también las estructuras de datos:
- Lista de peticiones, donde
- Petición = dir. datos grabados + puntero al BCP bloqueado



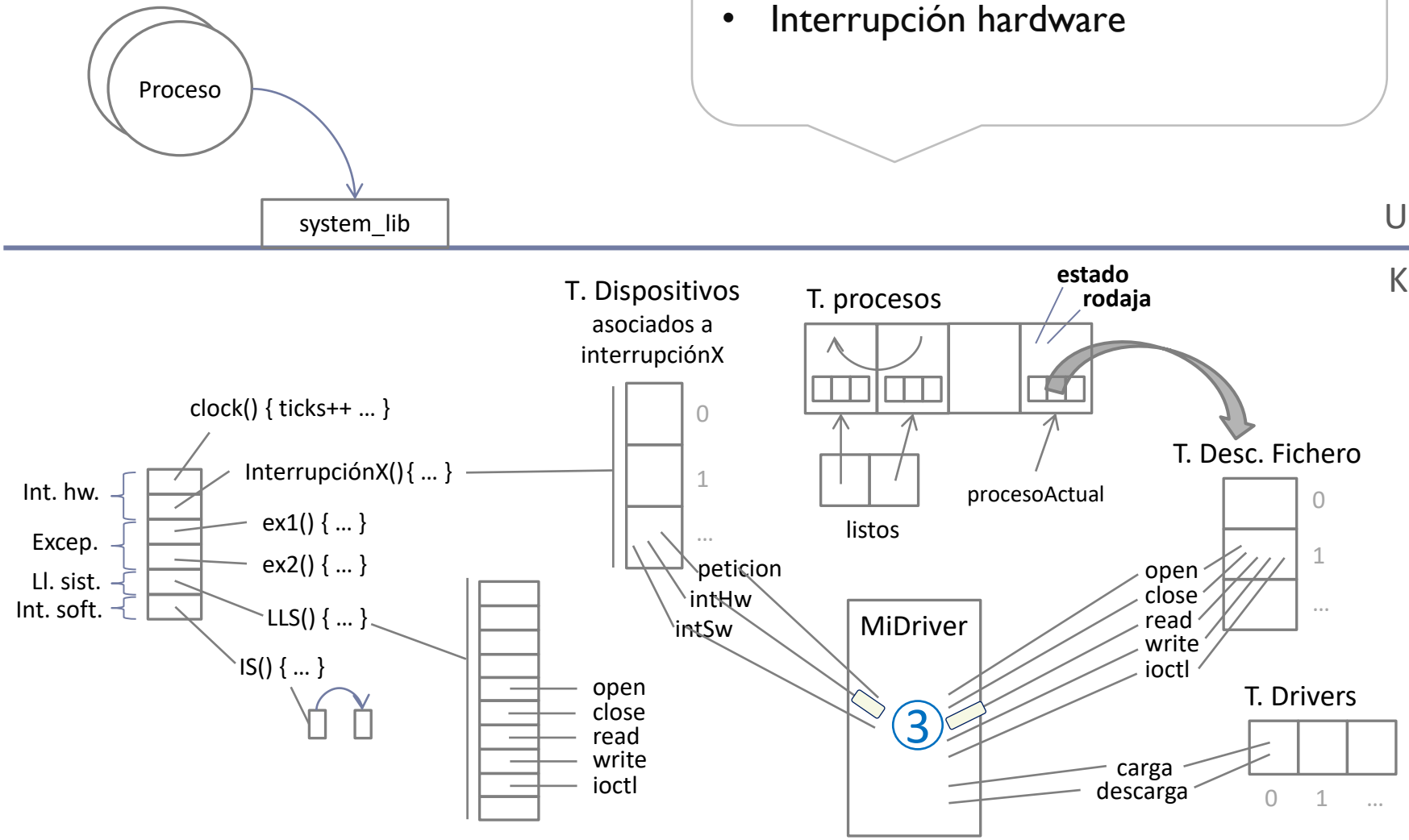
U

K



Ejercicio
solución

Apartado b)
Eventos involucrados son:
- Llamada al sistema open/close/read
- Interrupción hardware



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

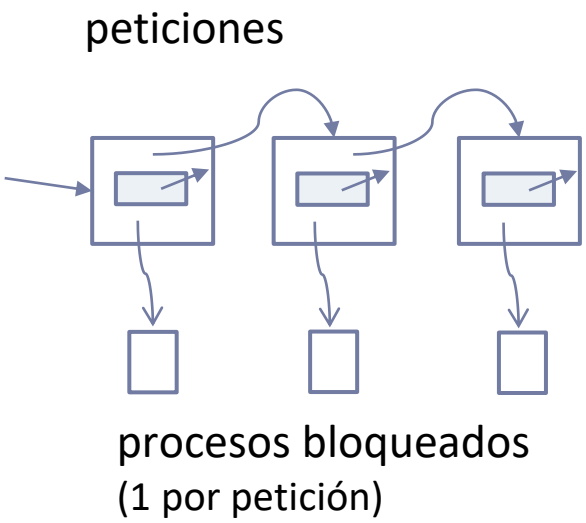
Ejercicio
solución a)

Mirando el planteamiento realizado, contestamos a las preguntas

- Gestionar la interrupción de la tarjeta de sonido:
  - **TarjetaSonido_ManejadorInterrupcionHW();**
  - **TarjetaSonido_ManejadorInterrupcionSW();**
- Gestionar las llamadas al sistemas del driver:
  - Uso de la opción de utilizar el estándar POSIX/UNIX:
    - **Desc = Open (nombre_teclado, flags)**
    - **Res = Close (Desc)**
    - **Res = Read (Desc, buffer, size)**
      - Indica que grabe sonido y devuelve los datos en buffer y el número de bytes como función.
- Gestionar la carga y descarga del driver en tiempo de ejecución:
  - **TarjetaSonido_Cargar();**
  - **TarjetaSonido_Descargar();**

Ejercicio
solución a)

- Los datos del driver de tarjeta de sonido en una estructura, con los siguientes campos:
  - Punteros a funciones de la interfaz.
  - Lista enlazada de peticiones (datos grabados, proceso bloqueado)



Ejercicio

~~solución b)~~ primer intento de solución b)

TarjetaSonido_ManejadorInterrupcionHW():

- Obtener resultado de los registros de E/S
- peticionEnCurso = primera(SONIDO.ListaPeticiones)
- Copiar(registro_de_dato, peticionEnCurso->buffer)
- Si quedan_más_datos
  - Return;
- Borrar(SONIDO.ListaPeticiones, peticionEnCurso)
- peticionEnCurso->BCP->estado = LISTO
- Insertar(ListaListos, peticionEnCurso->BCP)
- peticionEnCurso = primera(SONIDO.ListaPeticiones)
- Si peticionEnCurso != NULL
  - Enviar la orden de grabación a los registros de control y de datos del dispositivo

Ejercicio
solución b)

TarjetaSonido_ManejadorInterrupcionHW():

- Obtener resultado de los registros de E/S
- peticionEnCurso = primera(SONIDO.ListaPeticiones)
- Copiar(registro_de_dato, peticionEnCurso->buffer)
- Si quedan_más_datos
  - Return;

- InsertarTareaPendiente(TarjetaSonido_ManejadorInterrupciónSW) ;
- ActivarInterrupcionSoftware();

TarjetaSonido_ManejadorInterrupcionSW():

- Borrar(SONIDO.ListaPeticiones, peticionEnCurso)
- peticionEnCurso->BCP->estado = LISTO
- Insertar(ListaListos, peticionEnCurso->BCP)

- peticionEnCurso = primera(SONIDO.ListaPeticiones)
- Si peticionEnCurso != NULL
  - Enviar la orden de grabación a los registros de control y de datos del dispositivo

Ejercicio
solución b)

Llamada al sistema read(fd, buffer, size):

- Crear una nueva petición de grabación (petición)
  - petición->BCP = procesoActual
  - petición->Buffer = buffer
- Insertar(SONIDO.ListaPeticiones, petición)
- Si el dispositivo está libre
  - Enviar la orden de grabación a los registros de control y de datos del dispositivo
- procesoActual->estado = BLOQUEADO
- procesoAnterior = procesoActual
- procesoActual = Planificador()
- procesoActual->estado = EJECUTANDO
- CambioContexto(procesoAnterior, procesoActual)
  /* Aquí se bloquea el proceso, hasta un cambio de contexto de vuelta */
- (liberar memoria) y return petición->leídos

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 a evitar

1) Contestar a la primera pregunta de un apartado únicamente (y no contestar al resto de preguntas/peticiones)
2) Contestar a otra pregunta de la pedida.
3) Respuestas largas:
    1) Quitan tiempo para realizar el resto del examen.
    2) Contestar más de lo pedido puede suponer fallos extra.
    3) Importante que las partes claves del ejercicio estén correctas.
4) Usar el planteamiento del problema como respuesta.

Grupo ARCOS
Departamento de Informática
Universidad Carlos III de Madrid

Ejercicios

drivers y servicios ampliados

Diseño de Sistemas Operativos

Grado en Ingeniería Informática y
Doble Grado I.I. y A.D.E.

cc BY NC SA