

Ejercicios

Procesos e hilos, y planificación

Grupo ARCOS

Diseño de Sistemas Operativos
Grado en Ingeniería Informática
Universidad Carlos III de Madrid



Ejercicio

enunciado (1/3)

Se desea implementar un planificador basado en prioridades. Los procesos pueden tener 2 tipos distintos de prioridades:

- Prioridad alta.
- Prioridad baja.

Cada prioridad tiene su propia política de planificación:

- Los procesos de **prioridad alta** tendrán **FIFO**.
- Los procesos de **prioridad baja** tendrán **Round-Robin**, empleando en este caso una rodaja de tiempo de **100 milisegundos**.

Ejercicio

enunciado (2/3)

Los procesos en la cola de **prioridad alta** se ejecutan en orden estricto de llegada (FIFO). Un proceso de **prioridad alta** se ejecuta hasta que:

- Finaliza el proceso completamente.
- Se duerme.
- Se bloquea.

Un proceso de **prioridad baja** abandona el estado de ejecución cuando:

- Finaliza su rodaja de tiempo.
- Finaliza el proceso completamente.
- Se duerme.
- Se bloquea.

Ejercicio

enunciado (3/3)

Se pide:

a) Diseñar e indicar

qué funciones y estructuras de datos son necesarias para implementar el planificador indicado

usando como base el planificador inicial del sistema operativo base (que está siendo visto en clase en pseudocódigo).

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

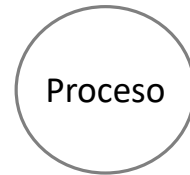
Realicemos un diagrama con el estado inicial del sistema, con los elementos más relevantes para el problema

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)

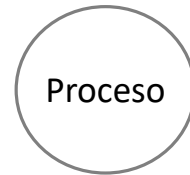
system_lib

U

K

Ejercicio solución

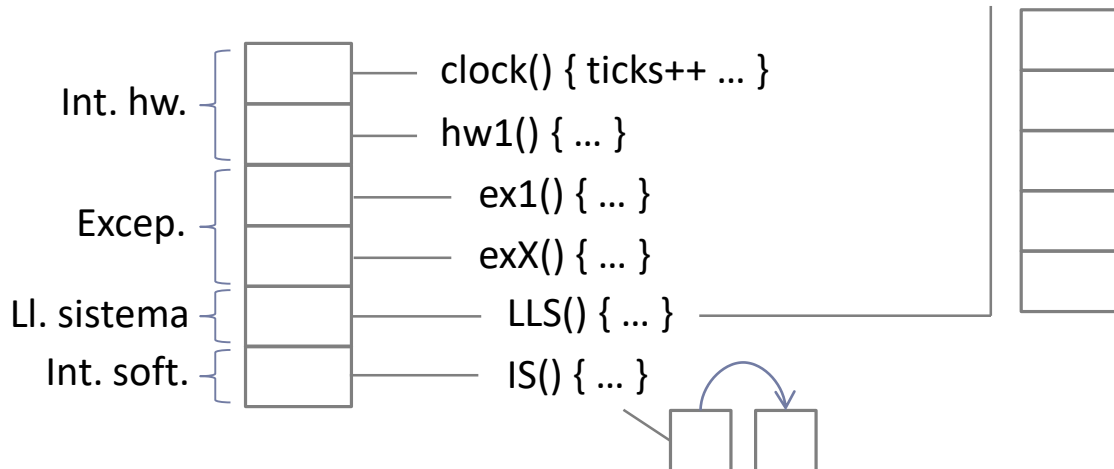
En el tema 2 se introducía el funcionamiento interno del núcleo del sistema operativo: interrupciones software, llamadas al sistema, excepciones e interrupciones hardware



system_lib

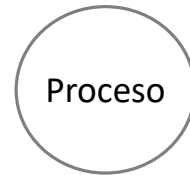
U

K



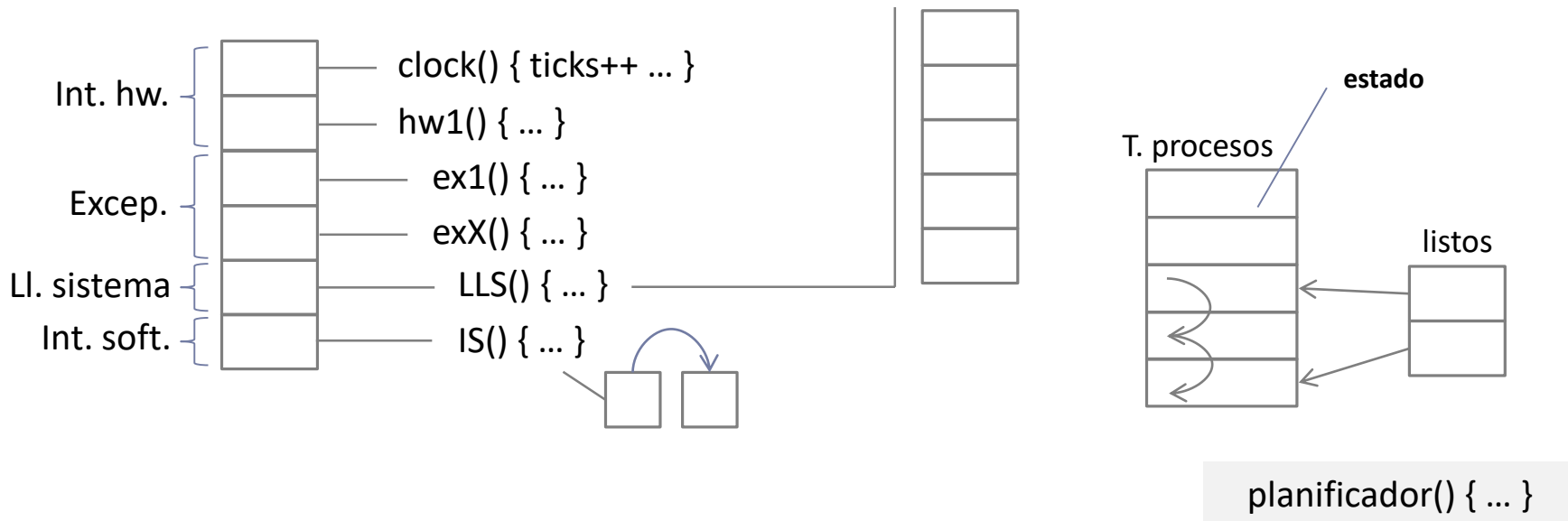
Ejercicio solución

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.



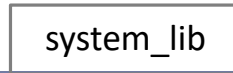
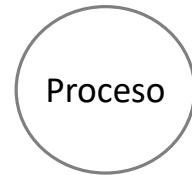
system_lib

U
K



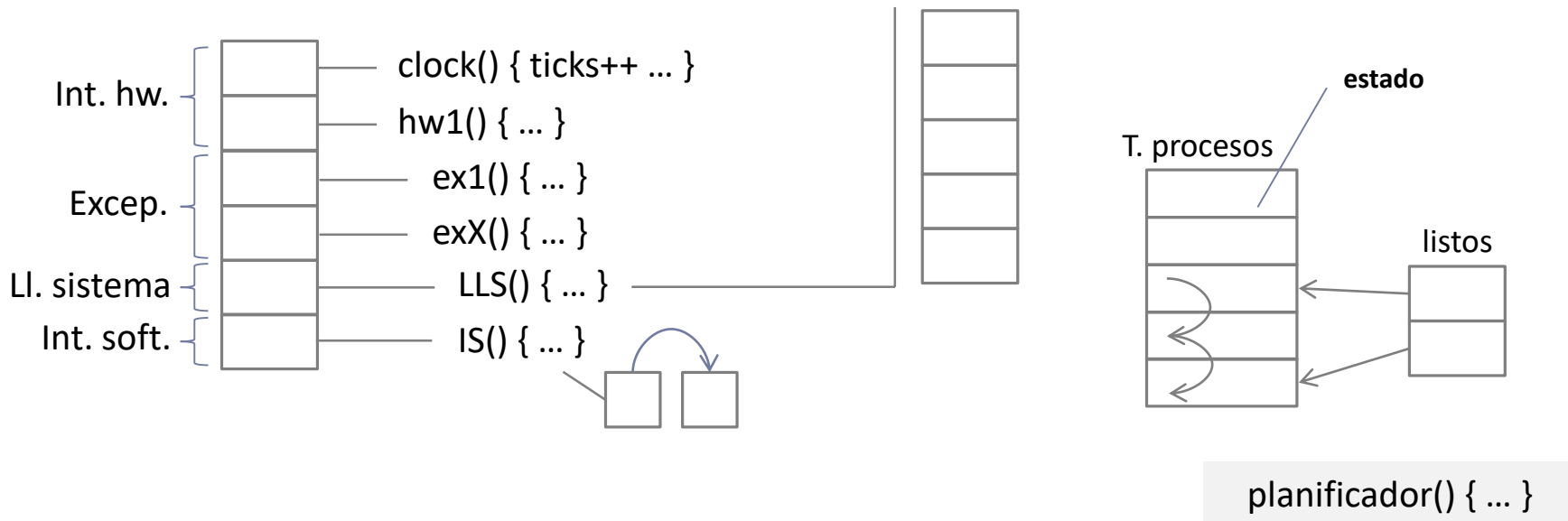
Ejercicio solución

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



U

K



Ejercicio

solución

1. Planteamiento inicial

1. Estado inicial del sistema

2. Estudio de qué hay que modificar

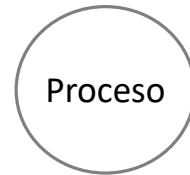
2. Responder a las preguntas

3. Revisar las respuestas

Ejercicio solución

Partimos de planificador FIFO,
y en el enunciado nos piden:

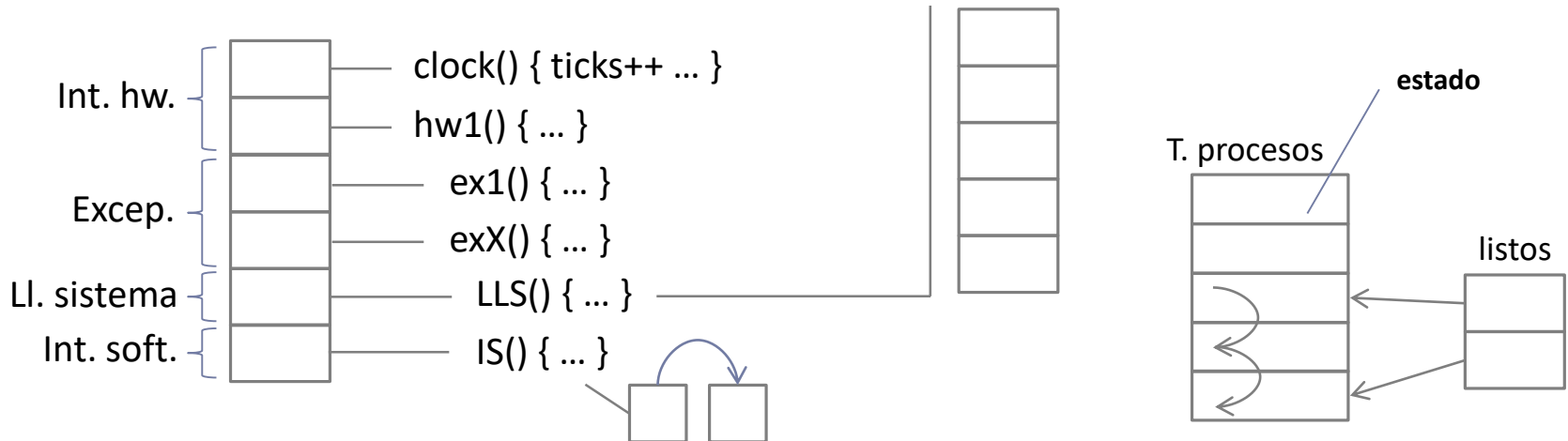
- Añadir prioridades.
- Añadir RR de 100 ms de rodaja para prioridad baja



system_lib

U

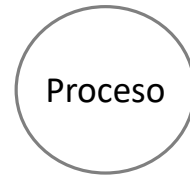
K



Ejercicio solución

Para añadir prioridades hay que:

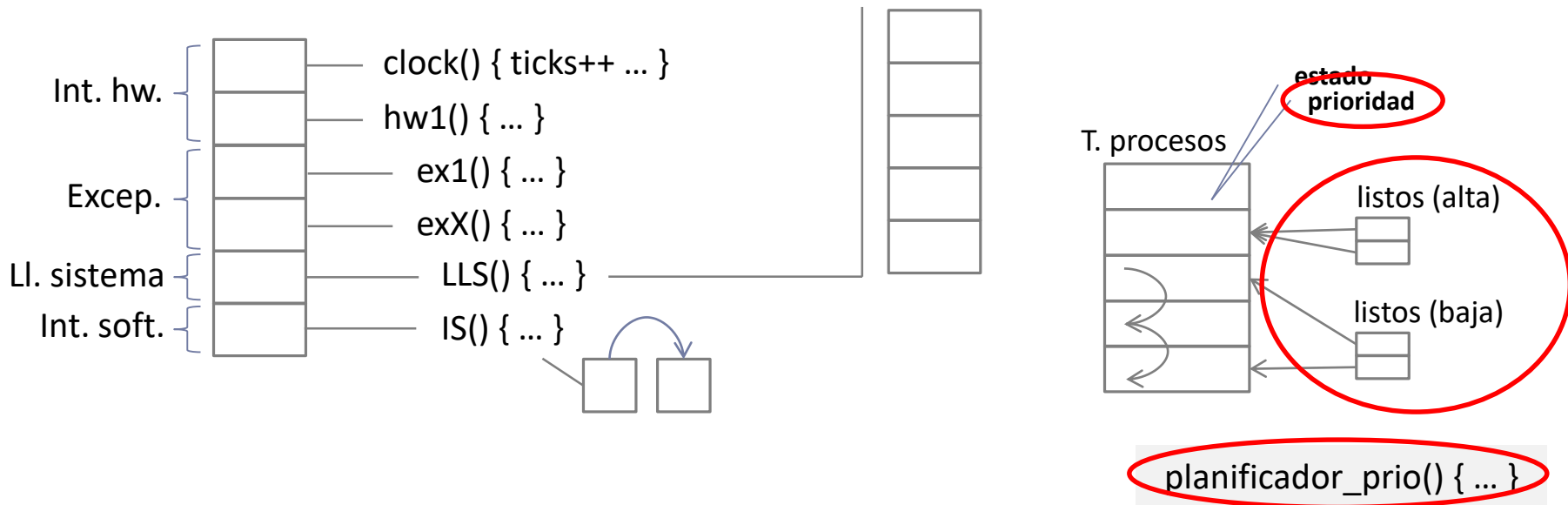
- 1) Añadir el campo prioridad en el BCP
- 2) Disponer de dos listas de listos
- 3) Modificar el algoritmo de planificación



system_lib

U

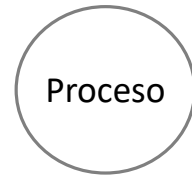
K



Ejercicio solución

Para añadir *Round-Robin* hay que:

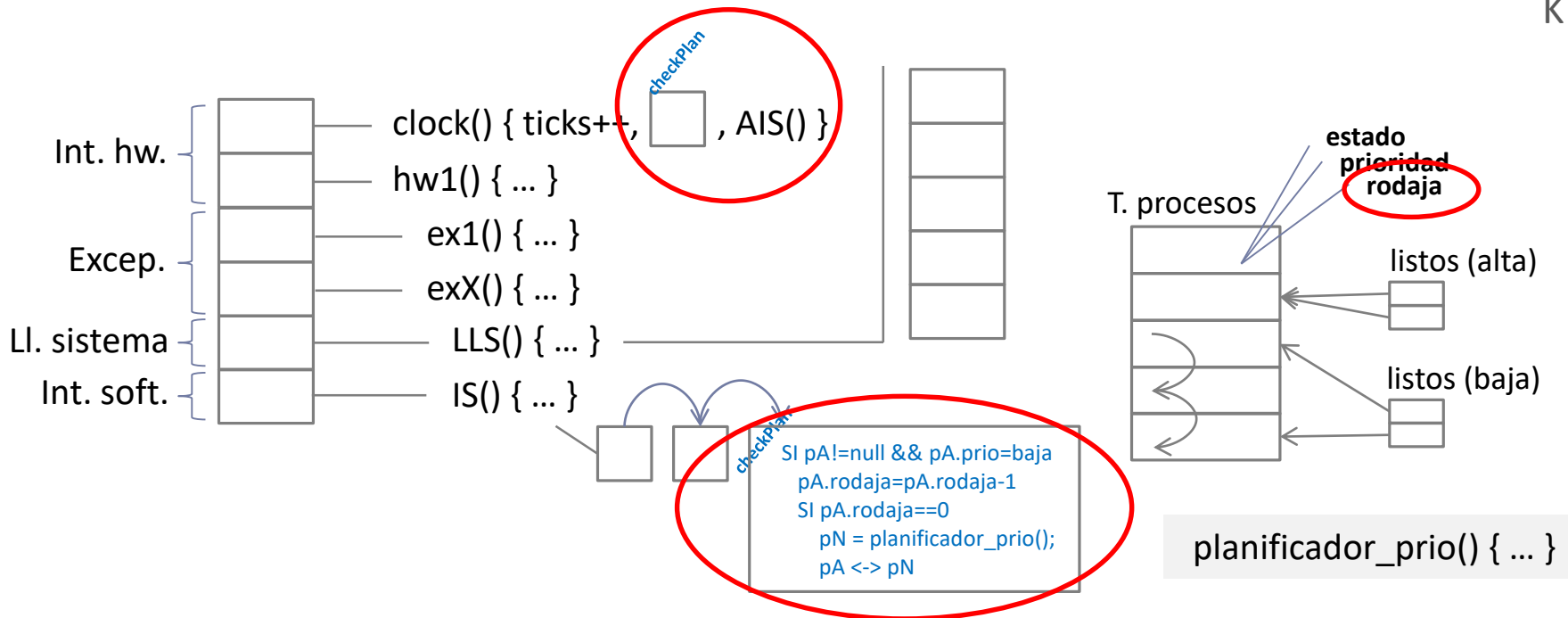
- 1) Añadir el campo rodaja en el BCP
- 2) Modificar la Interrupción de reloj
- 3) Tener una tarea pendiente a ejecutar en la interrupción software



system_lib

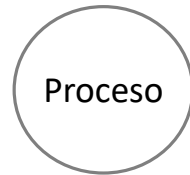
U

K



Ejercicio solución

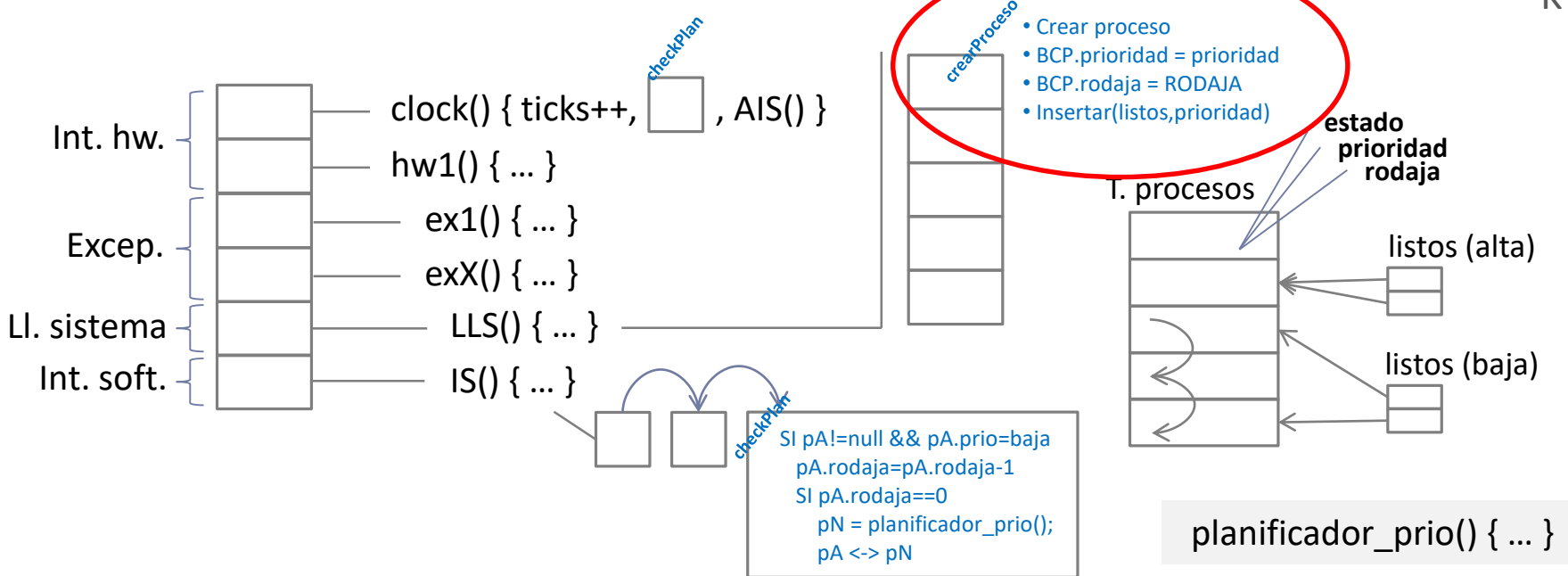
Para ambas cosas, los valores iniciales de prioridad y rodaja (e insertar en la lista correspondiente) se ha de hacer en la creación del proceso (llamada al sistema que hay que modificar)



system_lib

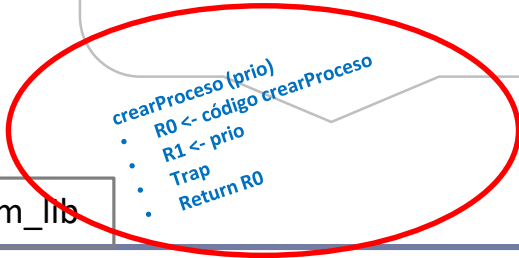
U

K



Ejercicio solución

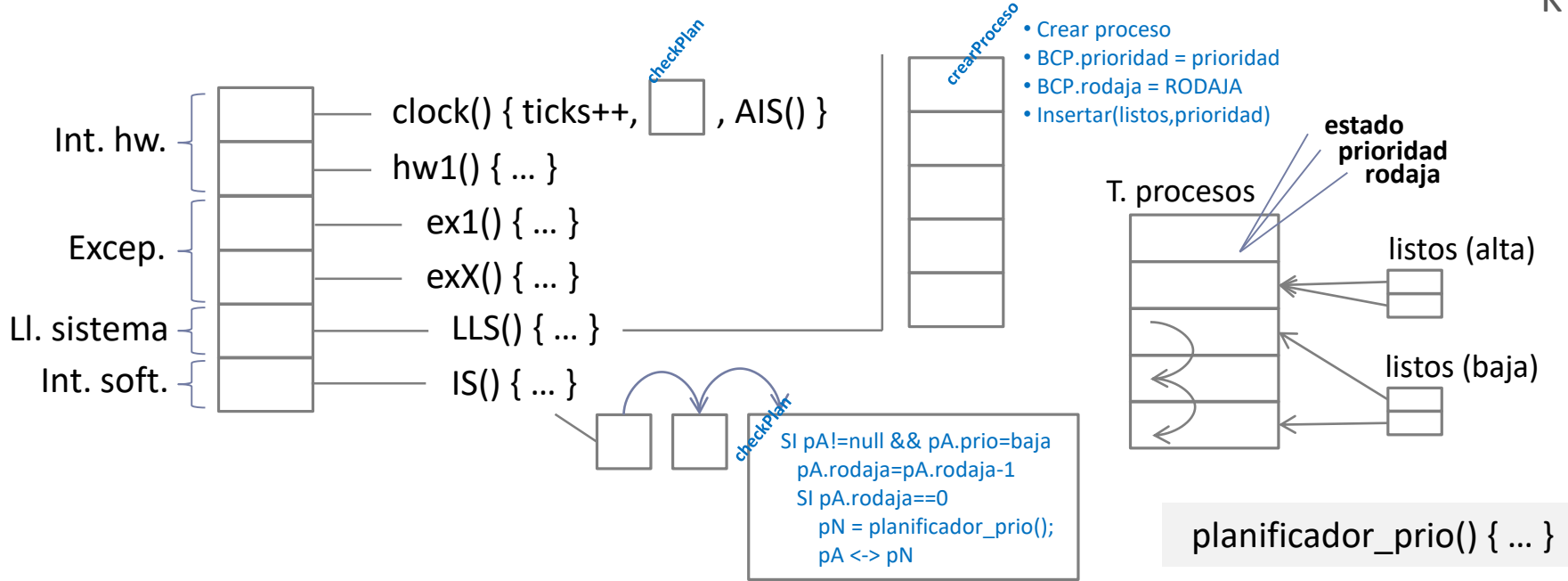
Al tener que pasar la prioridad como nuevo parámetro a crearProceso, hay que modificar su invocador en system_lib para pasar ese parámetro.



system_lib

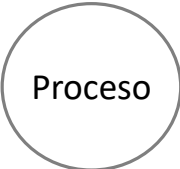
U

K



Ejercicio solución

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

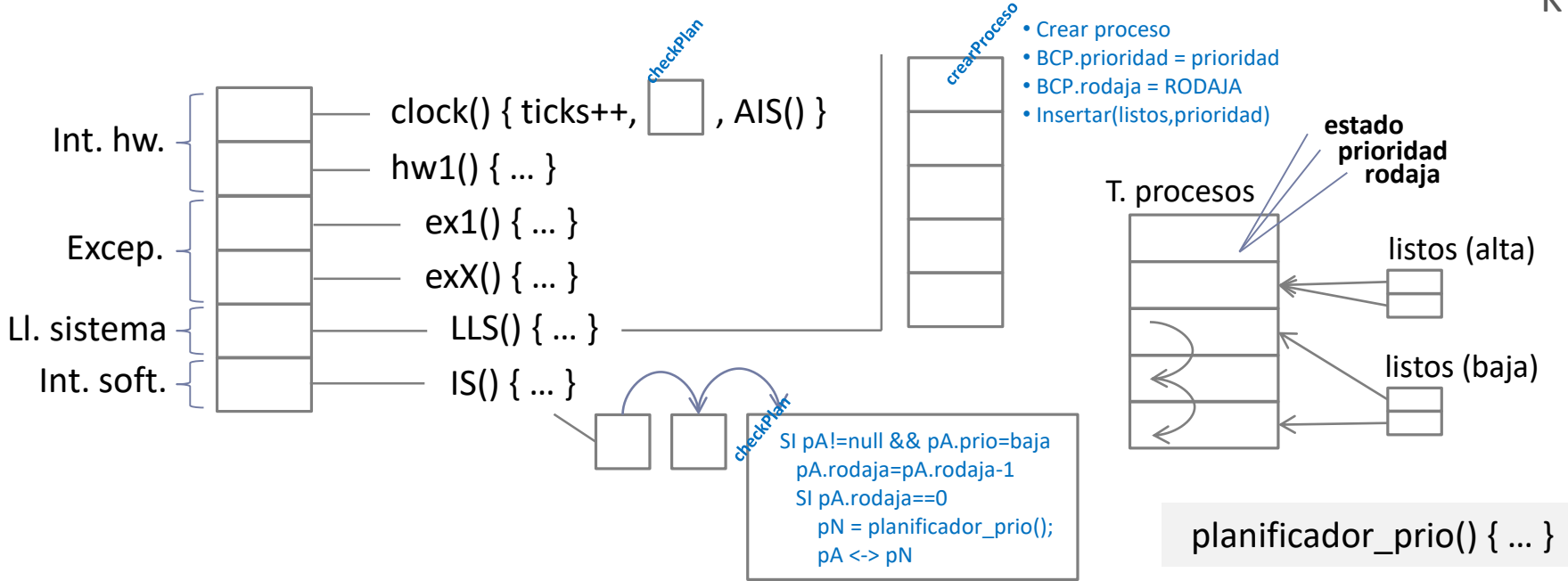


system_lib

- crearProceso (prio)
- R0 <- código crearProceso
- R1 <- prio
- Trap
- Return R0

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:

- En el BCP:
 - Prioridad
 - Rodaja
- Cambiar la lista de listos por dos listas:
 - Listos con prioridad baja
 - Listos con prioridad alta

Ejercicio

solución

Funciones:

Planificador_prioridades()

- Si No_vacia (lista_procesos_listo_alta_prioridad)
 - Proc=Obtener_primer_proceso (lista_procesos_listo_alta_prioridad)
 - Borrar (lista_procesos_listo_alta_prioridad,Proc)
- En caso contrario // lista vacía
 - Proc=Obtener_primer_proceso (lista_procesos_listo_baja_prioridad)
 - Borrar (lista_procesos_listo_baja_prioridad, Proc)
- Devolver Proc

Ejercicio

solución

Manejador_interruccionHW_reloj()

- Ticks++;
- Insertar_Interruccion_Software(Manejador_interruccionSW_reloj);
- Generar_Interruccion_Software();

Ejercicio

solución

Manejador_interrupcionSW_reloj()

- Si ((procesoActual == null) || (procesoActual.prioridad == alta))
 - Terminar función
- procesoActual.rodaja = procesoActual.rodaja - 1
- Si (proceso_en_ejecucion.rodaja == 0)
 - procesoActual.estado = LISTO ;
 - procesoActual.rodaja = TICKS_POR RODAJA // == 100 milisegundos
 - InsertarAlFinal (lista_procesos_listos_baja_prioridad, procesoActual) ;
 - antiguoProcesoActual = procesoActual;
 - procesoActual=Planificador_prioridades() ;
 - procesoActual.estado = EJECUTANDO;
 - CambiarContexto (antiguoProcesoActual.contexto, procesoActual.contexto) ;

Ejercicio

solución

int crear_proceso_prioridad (prioridad)

- Si ((prioridad != alta) && (prioridad != baja))
 - Error
- CPU.Registro0 = código de la llamada crear_proceso_prioridad
- CPU.Registro1 = prioridad
- TRAP
- Return CPU.Registro0

Ejercicio

solución

kernel_crear_proceso_prioridad ()

- BCP = Crear proceso (*proceso*)
- BCP.prioridad = CPU.RegistroI
- BCP.rodaja = RODAJA // tiene sentido para prioridad==baja
- BCP.estado = LISTO
- Si (*prioridad* == alta)
 - Insertar el proceso en la lista de listos(alta)
- Si (*prioridad* == baja)
 - Insertar el proceso en la lista de listos(baja)
- En caso contrario
 - Error (CPU.Registro0 = código de error)

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

Procesos e hilos, y planificación



Grupo ARCOS

Diseño de Sistemas Operativos
Grado en Ingeniería Informática
Universidad Carlos III de Madrid