Examples of exercises

Use case: Elevator driver

ARCOS

Operating Systems Design

Degree in Computer Engineering

University Carlos III of Madrid

Exercise

Statement (1/2)

- We start from an embedded system with one CPU core and preemptive OS scheduler. The aim is to develop a driver for the keyboards of an elevator in a five floor building.
- The keyboard inside the elevator emits a hardware interrupt (HW1) each time a key is pressed. Keyboard data register contains the key that was pressed (floor number).
- Each floor has its own key, all of them part of an external keyboard. Each time a key from that keyboard is pressed, a hardware interrupt (HW2) is emitted. The keyboard data register stores floor number related pressed key.
- The driver has to:
  - Store the keys if no processes read them.
  - Manage the keyboard interrupts when users press a key.
  - Upon a process requires to read a key, the driver writesthe first key pressed to the process.
  - If there are no keys pressed, a process reading from the driver remains blocked

Exercise
Statement (2/2)

You are asked to:

a) Design the driver interface, following the UNIX standard for system calls.
b) Define all data structures necessary to provide the required functionality.
c) Implement the requested functionality using pseudo-code to allow the process get the keys using software interrupts.

Exercise
solution

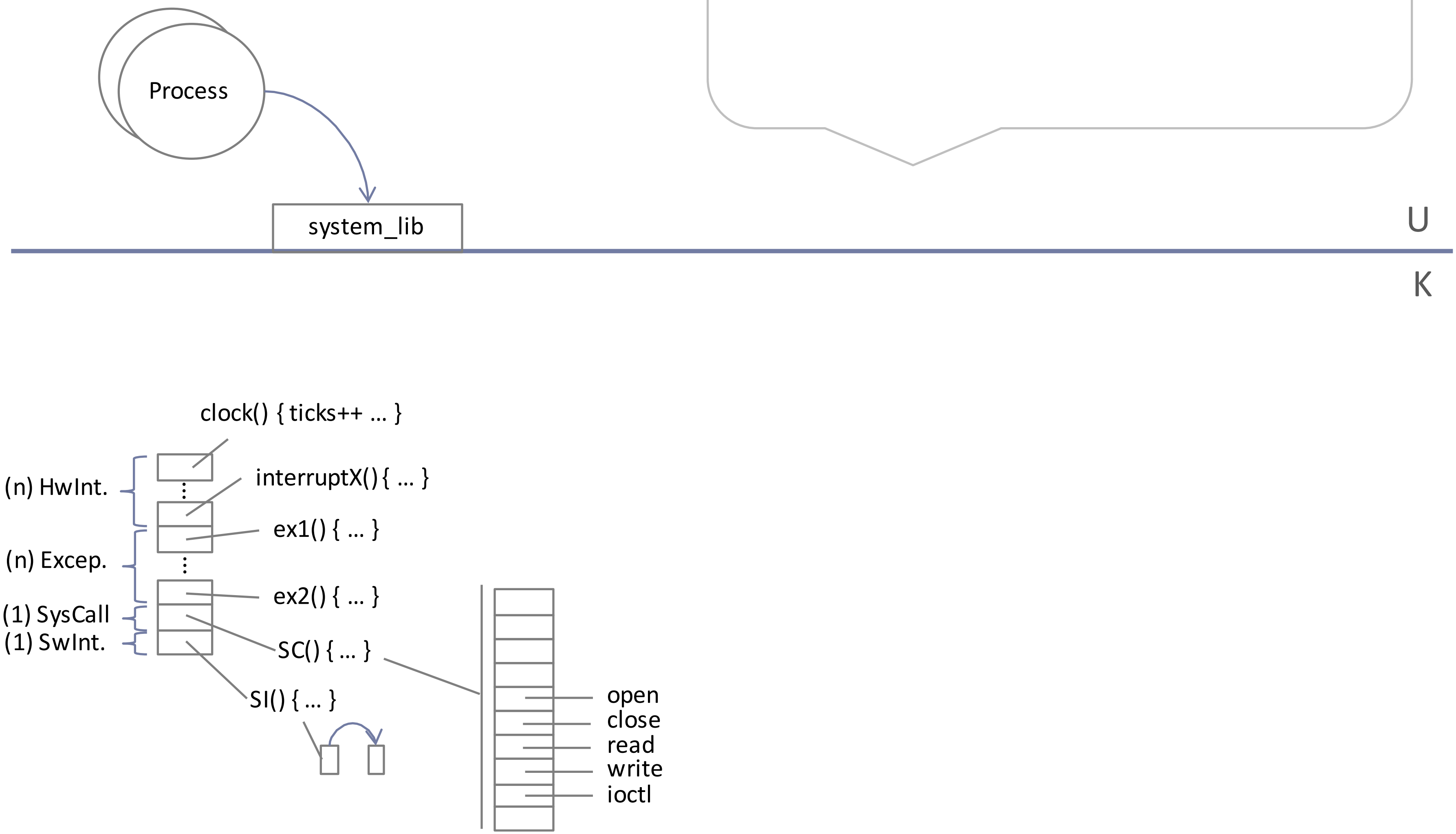
1. Initial approach:
    1. Draw a diagram of initial system state
    2. Modify the diagram to incorporate the exercise requirements
2. Answer the proposed questions
3. Review the answers

Exercise
solution

1. Initial approach:
   1. Draw a diagram of initial system state
   2. Modify the diagram to incorporate the exercise requirements
2. Answer the proposed questions
3. Review the answers

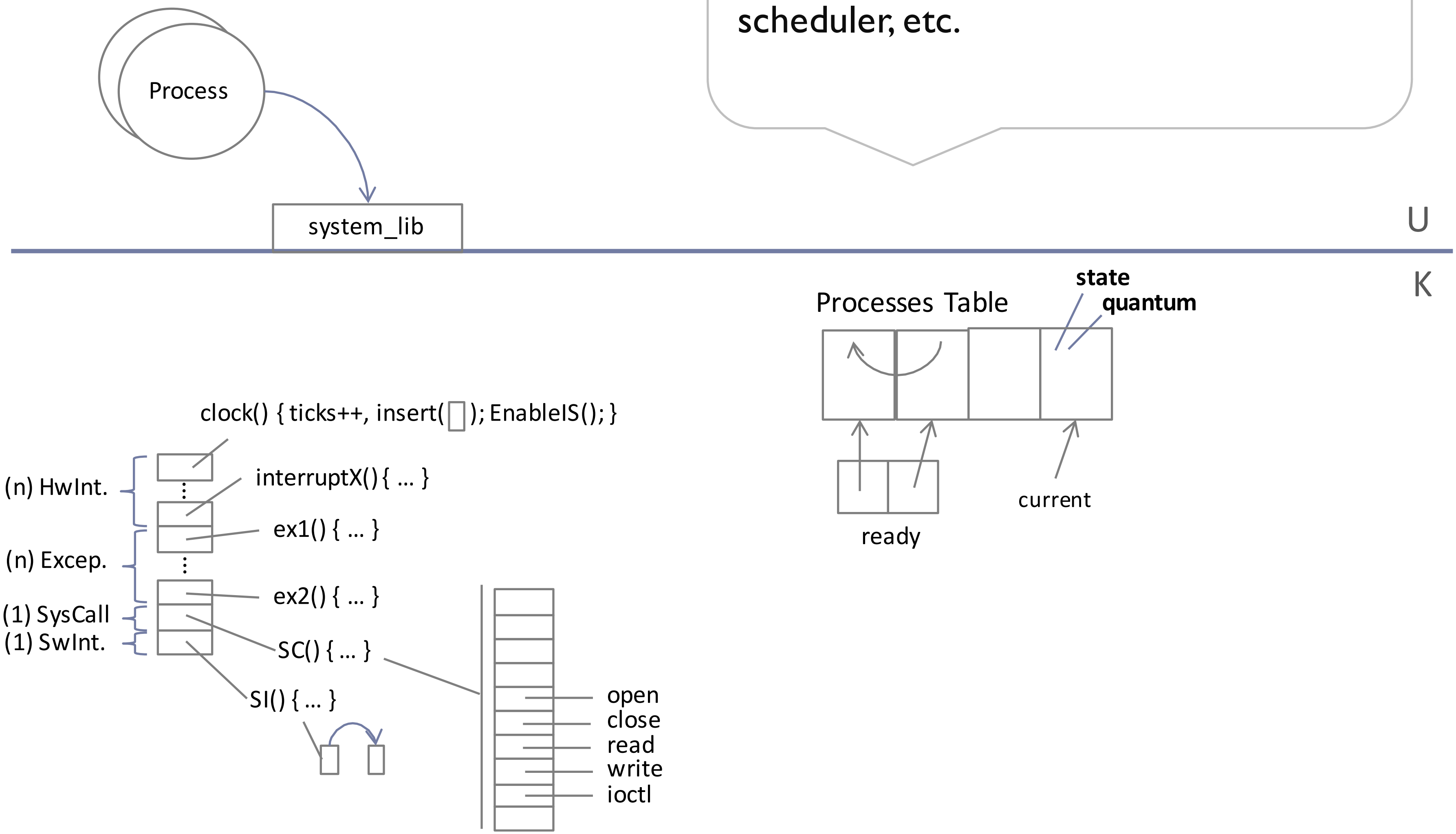
Exercise
solution

Lesson 2: operating system working introduces the four types of events…



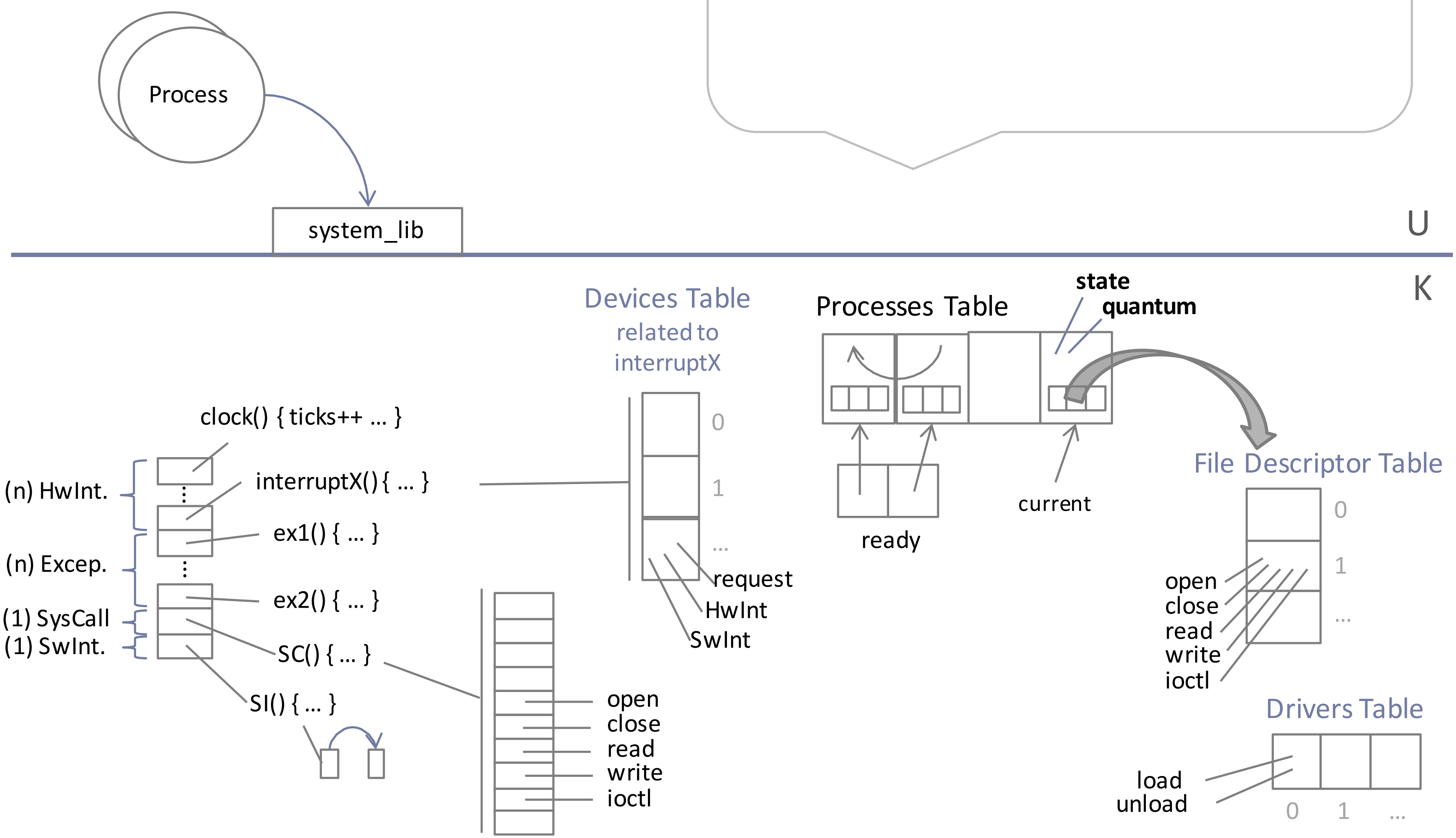
Exercise
solution

Lesson 3b introduces data structures and internal funtions for process management, e.g., the ready queue, the scheduler, etc.



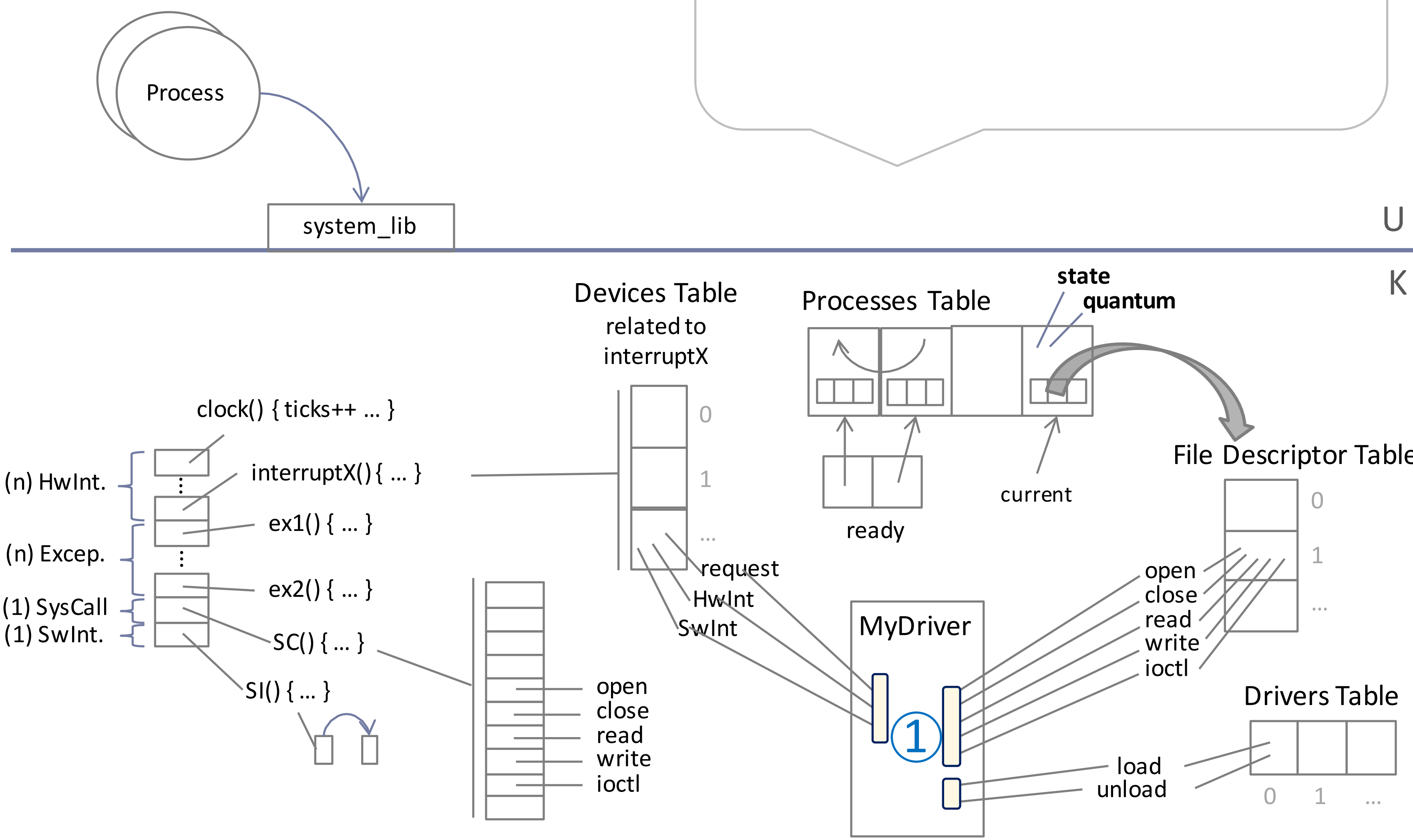
Exercise
solution

Lesson 3c introduces the driver framework (device table, file descriptor and driver table)...



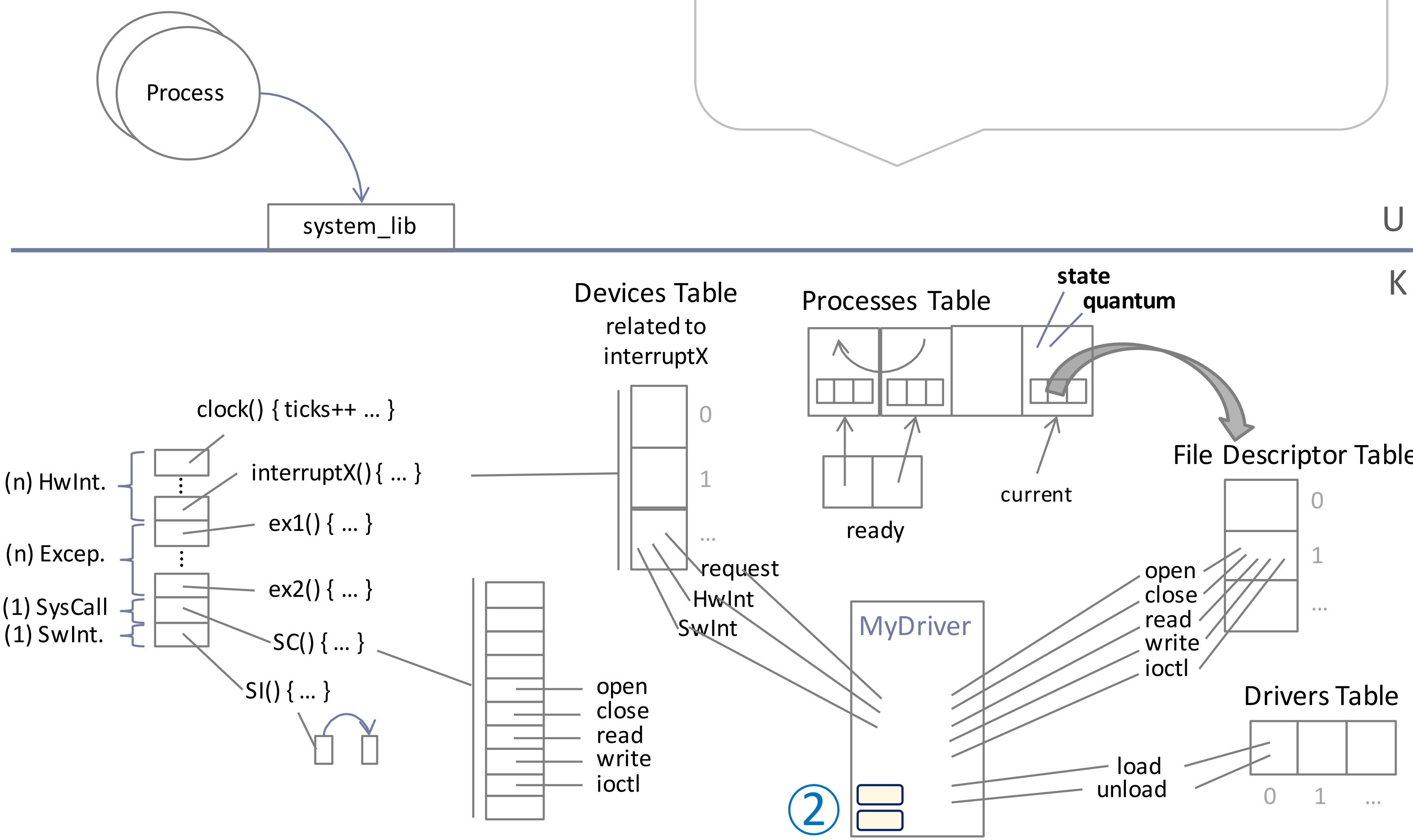
Exercise
solution

Lesson 3c also introduces the three set of functions in which a driver consists of…



Exercise
solution

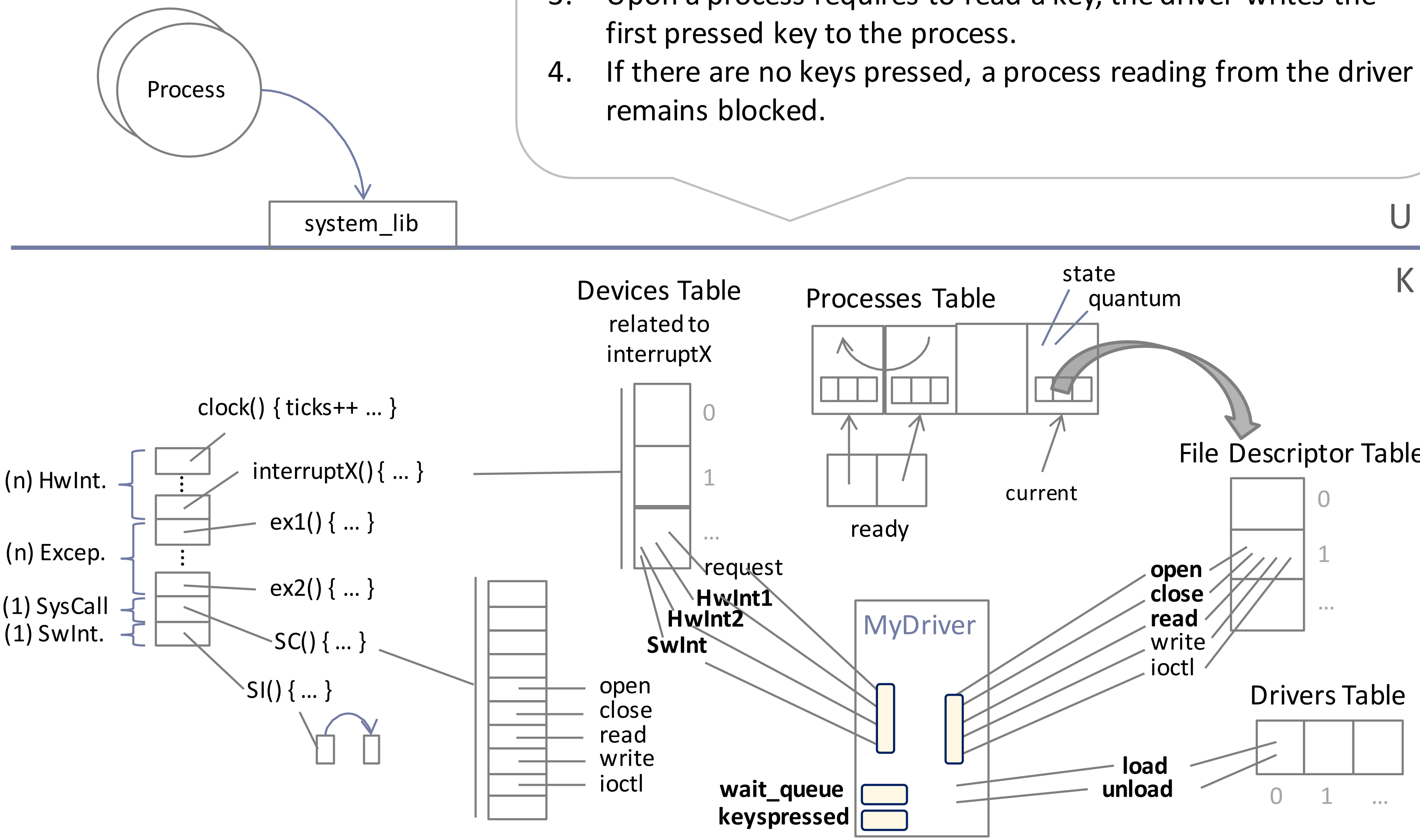
Lesson 3c also introduces the data structures…



Exercise
solution

Modify the diagram to:

1. Store the keys if no processes read them.
2. Manage the keyboard interrupts when users press a key.
3. Upon a process requires to read a key, the driver writes the first pressed key to the process.
4. If there are no keys pressed, a process reading from the driver remains blocked.



Exercise
solution

1. Initial approach:
   1. Draw a diagram of initial system state
   2. Modify the diagram to incorporate the exercise requirements
2. Answer the proposed questions
3. Review the answers

Exercise
solution

a) The interface consists of three main parts:
- Operating System:
  - **Load and unload the driver**
- Manage de HW device:
  - **HW_interrupt_1_handler();**
  - **HW_interrupt_2_handler();**
  - **SW_interrupt();**
- Driver interface using UNIX/POSIX:
  - **desc = open(keyboard_name, flags);**
  - **res = close(desc);**
  - **res = read(desc, buffer, size);**

b) Data structures are:
- Driver data structure which contains the function pointers.
- Keyboard buffer (list of stored keys)
- List of processes blocked waiting for a key

Exercise
solution

c) The events involved in a key read are the following:
- Keyboard interrupts
- Key request function
- Read system call

- The pseudo-code is the following:

HW_interrupt_1_handler()
- key = inb(data_register);
- Insert_key(key, keyboard.buffer)
- Insert_sofware_interrupt(SW_interrupt)
- Raise_software_interrupt();

HW_interrupt_2_handler()
- key = inb(data_register);
- Insert_key(key, keyboard.buffer)
- Insert_sofware_interrupt(SW_interrupt)
- Raise_software_interrupt();

Exercise
solution

SW_interrupt()
- Proc = ExtractFirstProcess(keyboard.blocked_processes_list);
- Proc.state = ready;
- Enqueue(ready_state_queue, proc);

Read_char();
- If (empty(keyboard.buffer))
  - enqueue (keyboard.blocked_processes_list, current);
  - current.state = BLOCKED;
  - old_current = current;
  - current = Scheduler(); // ExtractFirstProcess (ready_state_queue);
  - current.state = EXECUTION;
  - swap_context (old_current, current); // Activator (dispatcher)
- return (extract_key(keyboard.buffer));

Exercise
solution

_kernek_read_system_call(int fd, char * buffer, int size)

- for (i=0; i<size; i++)
  - Buffer[i] = Read_char();
- return size;

Exercise
solution

1. Initial approach:
    1. Draw a diagram of initial system state
    2. Modify the diagram to incorporate the exercise requirements
2. Answer the proposed questions
3. Review the answers

Examples of exercises
Use case: Elevator driver

ARCOS

Operating Systems Design

Degree in Computer Engineering

University Carlos III of Madrid