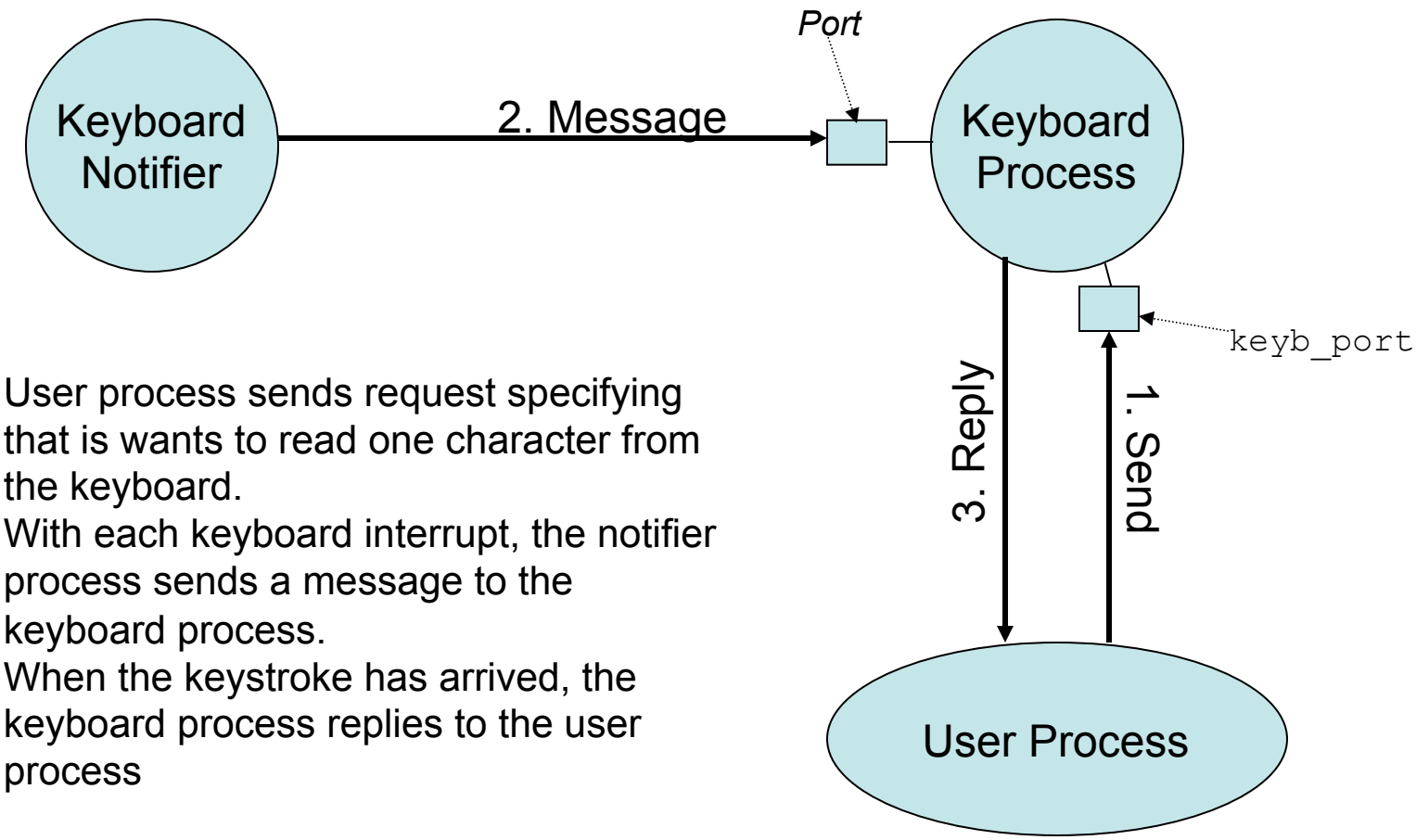
TOS Shell

Activating the Keyboard

- In order to use the keyboard, the following steps have to be done:
  - Register the ISR for the keyboard interrupt. This can be accomplished by calling `init_idt_entry (KEYB_IRQ, isr_keyb)` in `init_interrupts()`
  - Make sure that `isr_keyb()` works with your implementation of `wait_for_interrupt()`
  - Call `init_keyb()` from the boot process.

Design of Keyboard Service



1. User process sends request specifying that is wants to read one character from the keyboard.
2. With each keyboard interrupt, the notifier process sends a message to the keyboard process.
3. When the keystroke has arrived, the keyboard process replies to the user process

Keyboard Process Message

```
typedef struct _Keyb_Message {
    char* key_buffer;
} Keyb_Message;
```

- Defined in `~/tos/include/kernel.h`
- Instances of `Keyb_Message` are sent to the Keyboard Process by the user process
- Member:
  - `key_buffer`: pointer to a buffer that can hold a single character.

Using the Keyboard

```
void user_process(PROCESS self, PARAM param)
{
    char ch;
    Keyb_Message msg;

    while (1) {
        msg.key_buffer = &ch;
        send(keyb_port, &msg);
        kprintf("%c", ch);
    }
}
```

- The program above will print out whatever the user types on the keyboard.
- `keyb_port` is owned by the keyboard process and will be initialized in `init_keyb()`
- Note: keyboard process only returns one character at a time.

TOS Shell

```
WINDOW shell_wnd = {0, 9, 61, 16, 0, 0, 0xDC};

void shell_process(PROCESS self, PARAM param)
{
    while (1) {
        - read command from keyboard
        - when user hits <enter> execute command
    }
}
```

- The purpose of the TOS shell is to allow a user to type commands.
- The TOS shell is implemented as a TOS process. This process gets created in `init_shell()`.
- The shell should generate all its output in the window defined by `shell_wnd`.

TOS Shell Commands

- The TOS Shell should understand the following commands:
  - Help function (print a list of all commands)
  - Clear the shell window (`clear_window()`)
  - Print the process table (`print_all_processes()`)
  - Make the train stop and go
  - Change the position of the switches
  - Start the PacMan game
- You can define the syntax of each of those commands.

Assignment 10

- Implement a TOS shell in `~/tos/kernel/shell.c`
- The shell should understand the commands mentioned on an earlier slide.
- Make sure you initialize the keyboard.
- No test cases are available for this assignment.