Abdullah Almarzouq
5/22/2020

Blink LED0 Kernel Module

Kicking around…

a) Creating '/dev/ece_led' and blink_rate:

To create the character device file (without mknod), I had to use 'class_create()' & 'device_create()' functions in the __init(), and 'class_destroy()' & 'device_destroy()' in the __exit(). I've also been using the same kernel module source code from *Simple PCI Dev Driver* folder. With these functions we just mentioned, a device node can be created with the name '/dev/ece_led'. I've also added a module parameter called 'blink_rate' and assigned it to a value of 2 as a default. As shown in the figure below, when the driver loads, the node gets created and the 'blink_rate' parameter is set to 2.

Also, we the driver get unloaded; the character device file gets destroyed just as shown in the figure below:

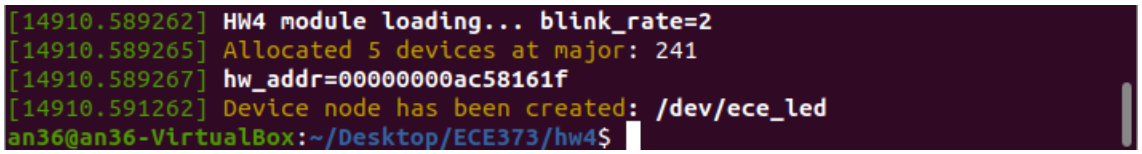
b) Blinking LED0 using a timer:

I've modified the driver module source code to have a timer which controls the blinking of LED0. The timer starts if the device file (/dev/ece_led) has been opened. Also, I've created a userspace program called 'testing.c' which opens the device file to let the timer start. The userspace program has only one parameter which control the led blinking rate. When the userspace program is being executed with a parameter, first it prints the current blink_rate value, then it assigns the blink_rate value with the parameter value. The userspace program can be executed in the following way:

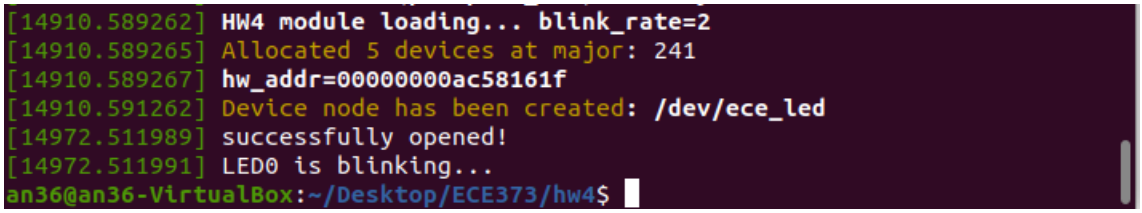
sudo ./<program_name> <parameter value (int)>

If the user didn't enter a value then the parameter will have a default value of 2, and if the user entered a zero or a negative number then the program will return an error and exit.

Also, as shown in the figure(s) below, the blinking starts when the userspace program open the device file:



The figure above shows the kernel log when the driver module has been loaded. While the figure below shows a message "LED0 is blinking…" after executing the userspace program.



c) Modifying blink_rate:

For this step, I've modified the userspace program 'testing.c' to have the ability to open '/sys/module/hw4/parameters/blink_rate', read its current value and modify it to be the same as the value which the user entered.

As shown in the figure below, the userspace program can indeed modify the 'blink_rate' parameter.

```
an36@an36-VirtualBox:~/Desktop/ECE373/hw4$ sudo ./test 2
Current blink_rate: 2
New value of blink_rate: 2
file(s) closed
an36@an36-VirtualBox:~/Desktop/ECE373/hw4$ sudo cat /sys/module/hw4/parameter
s/blink_rate
2
an36@an36-VirtualBox:~/Desktop/ECE373/hw4$ sudo ./test 36
Current blink_rate: 2
New value of blink_rate: 36
file(s) closed
an36@an36-VirtualBox:~/Desktop/ECE373/hw4$ sudo cat /sys/module/hw4/parameter
s/blink_rate
36
an36@an36-VirtualBox:~/Desktop/ECE373/hw4$
```

f) Error on negative and zero:

As shown in the figure below, the userspace program won't let the user write a negative value or a zero to 'blink_rate'. Even if the user echo-ed zero or a negative integer to '/dev/ece_led' (which can change the blink_rate value) the driver will print an error and reset 'blink_rate' to the default value.

```
an36@an36-VirtualBox:~/Desktop/ECE373/hw4$ sudo ./test -1
ERROR: Invalid Argument
an36@an36-VirtualBox:~/Desktop/ECE373/hw4$ sudo ./test -500
ERROR: Invalid Argument
an36@an36-VirtualBox:~/Desktop/ECE373/hw4$ sudo ./test 0
ERROR: Invalid Argument
an36@an36-VirtualBox:~/Desktop/ECE373/hw4$
```