Operating System write up HW6

Ding Qin

Vinay Krishnan

For this homework assignment we had to implement a kernel module that acted as a driver for an actual external device. The driver allowed the user to access the device, control its operation, and read/write data to it. We chose to utilize the GPIO pins on the raspberry pi to blink the led in Morse code of the letter the user chooses. For example the user echos letters “abc” to the device driver. Then when the user cats the device driver, the LED will blink the letters in Morse code with a pause in between each letter.

To do this we took elements from both homework 2 and homework 5. In homework 2 we toggled an external led by writing a 1 or 0 to the GPIO 17 pin. In homework 5 we created a char device driver that the user can echo and cat from.

First we wanted to get the LED to turn on and off. To do so we utilized the gpio\_request, gpio\_direction\_output, gpio\_export, and gpio\_set\_value calls. First to turn on the led we would have to request the specified GPIO pin number, then we set the output of the specified pin and turn it on. Then we have to export it. To turn off the Led we set the value of the GPIO pin to 0.

To implement the Morse code portion of the assignment, we first created a new function called led\_morsecode(), which had a char \* parameter. In this function, it checks the parameter it was passed to see whether it was a “.” Or a “-“. If it was a dot, then it will have a shorter blinking time. To make sure the LED was on for a certain amount of time we used the mdelay call. This would make sure the LED stayed on for a certain amount of time before it was turned off. Also to account for spaces in between multiple letter transmissions I used a space counter. This flag would be set after the led has finished blinking the Morse code for one letter. Once the flag is set, it keeps the led off for a longer duration.

In order to make sure the LED will blink upon the Cat command and not the echo command we put the letter to Morse code conversion in the device\_read function. In here the elements in the file will be converted to dots and dashes. Also since cat reads the file twice we had to set a state so the LED only blinks the command on the first read.

I think our work met the design really well. Everything seemed to work as expected. But initially we wanted to implement the LED staying on for a specified number of milliseconds, but we could not figure out how to properly convert chars to ints. We tried kstrtoint, but we could not get that to work the way we intended.

To connect the GPIO to the led, we had the GPIO 17 pin connected to a 3kohm resistor, then the resistor was connected in series to the LED, and the LED was connected to ground.

To test our code the commands we used were as follows:

**make**

**insmod banner.ko**

**mknod /dev/chardev c 243 0**

**echo “ABC” > /dev/chardev**

**cat /dev/chardev**