Interrupts and Handlers

Objective: In this lab you will build a LKM called "interkey" will exercise some of the Linux kernel interrupt handling routines. Specifically, your module will create a handler for the keyboard and detect the presence of the ESC key which will be logged.

File(s) for this lab:

1. The following output demonstrates what your module should print to the debug log.

```
kernel: init_module() called
kernel: interkey: registering keyboard interrupt handler
kernel: interkey: ESC pressed
last message repeated 22 times
last message repeated 5 times
kernel: interkey: unregistering keyboard interrupt handler
kernel: Unloading interkey ...
```

Hints:

- Create a new project Lab8 and import the files from "LKI/Lab8" to get started.
- Use the following code sequence to get started with the LKM. The irq_handler is provided.

```
#include <linux/interrupt.h>
#include < linux/io.h >

#define DRIVER_AUTHOR "FocalPoint "
#define DRIVER_DESC   "Lab8"

MODULE_LICENSE("GPL");           // Get rid of taint message by declaring code as GPL.

/*  Or with defines, like this: */
MODULE_AUTHOR(DRIVER_AUTHOR);    // Who wrote this module?
MODULE_DESCRIPTION(DRIVER_DESC); // What does this module do?

int init(void) ; void
cleanup(void) ;

/* service keyboard interrupts handler */
irq_handler_t irq_handler(int irq, void *dev_id, struct pt_regs *regs)
{ static unsigned char scancode;

        /* read keyboard */
        scancode = inb( 0x60 );
        if((scancode == 0x01) || (scancode ==0x81)) printk("interkey: ESC pressed\n") ;
 return (irq_handler_t) IRQ_HANDLED;
}
```

- Complete the code to register and unregister the irq handler.

```
/* register the irq handler */
static int keybrd_int_register(void)
{
}

/* remove the handler */
static void keybrd_int_unregister(void)
{
}
```