Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 9
по операционным системам

Студент Колганов О.С.

Группа ИУ7 — 62Б

Преподаватель Рязанова Н.Ю.

Москва.
2020 г.

Задание:
· Написать загружаемый модуль ядра, в котором зарегистрировать обработчик аппаратного прерывания с флагом IRQF_SHARED.

· Инициализировать тасклет.

· В обработчике прерывания запланировать тасклет на выполнение.

· Вывести информацию о тасклете используя, или printk(), или seq_file interface - <linux/seq_file.h> (Jonathan Corber: http://lwn.net//Articales//driver-porting/).

Текст программы:

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/interrupt.h>
#include <linux/time.h>

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("TASKLET1_MODULE");
MODULE_AUTHOR("Moxxx1e");

static struct timespec64 cur_time;
static char my_tasklet_data[] = "my_tasklet_function was called";
void my_tasklet_handler(unsigned long data);

DECLARE_TASKLET(my_tasklet, my_tasklet_handler,
    (unsigned long) & my_tasklet_data);

void my_tasklet_handler(unsigned long data)
{
    ktime_get_real_ts64(&cur_time);
    int h = cur_time.tv_sec / 3600 % 24;
    int m = cur_time.tv_sec / 60 % 60;
    int s = cur_time.tv_sec % 60;

    printk(KERN_INFO "TASKLET INFO:"
                     "[TIME:%.2d:%.2d:%.2d]\n"
```

```
                    "\nstate:  %ld, count: %d, data: %s\n",
        h, m, s, my_tasklet.state, my_tasklet.count, my_tasklet.data);
    return;
}

static const int def_irq = 1;
static int irq_counter = 0;
static irqreturn_t irq_handler(int irq, void* dev)
{
    if (irq == def_irq) {
        irq_counter++;
        printk(KERN_INFO "INTERRUPT! irq_counter = %d", irq_counter);
        tasklet_schedule(&my_tasklet);
        return IRQ_HANDLED; // прерывание обрабатано
    }
    return IRQ_NONE; // прерывание не обработано
}

static int dev_id;
static int __init tasklet1_module_init(void)
{
    /* Schedule the Bottom Half */
    if (request_irq(def_irq, irq_handler, IRQF_SHARED,
"TASKLET_interrupt", &dev_id))
        return -1;

    printk(KERN_INFO "TASKLET1_MODULE loaded!");
    return 0;
}

static void __exit tasklet1_module_exit(void)
{
```

```
    /* Stop the tasklet before we exit */
    tasklet_kill(&my_tasklet);
    synchronize_irq(def_irq);
    free_irq(def_irq, &dev_id);
     printk(KERN_INFO "TASKLET1_MODULE: result irq_counter = %d\n",
irq_counter);
    printk(KERN_INFO "TASKLET1_MODULE unloaded!\n");
}


module_init(tasklet1_module_init);
module_exit(tasklet1_module_exit);
```

Загрузка модуля, проверка установленного обработчика первого прерывания(линия IRQ “разделяется”):

```
oleg@Moxxx1e:~/Документы/BMSTU/OS/lab_09$ sudo insmod tasklet.ko
oleg@Moxxx1e:~/Документы/BMSTU/OS/lab_09$ cat /proc/interrupts
           CPU0       CPU1       CPU2       CPU3
  0:          5          0          0          0   IO-APIC   2-edge      timer
  1:          0      21738          0          0   IO-APIC   1-edge      i8042, TASKLET_interrupt
  8:          0          0          1          0   IO-APIC   8-edge      rtc0
  9:          0         56          0          0   IO-APIC   9-fasteoi   acpi
 12:     854701          0          0          0   IO-APIC  12-edge      i8042
```

Демонстрация работы программы на следуюшей странице.

Демонстрация работы программы:

```
[24998.946108] TASKLET1_MODULE loaded!
[24999.078336] INTERRUPT! irq_counter = 1
[24999.078371] TASKLET INFO:[TIME:17:35:53]

               state:  2, count: 0, data: my_tasklet_function was called
[24999.117182] INTERRUPT! irq_counter = 2
[24999.117187] TASKLET INFO:[TIME:17:35:53]

               state:  2, count: 0, data: my_tasklet_function was called
[24999.120194] INTERRUPT! irq_counter = 3
[24999.120199] TASKLET INFO:[TIME:17:35:53]

               state:  2, count: 0, data: my_tasklet_function was called
[24999.123212] INTERRUPT! irq_counter = 4
[24999.123217] TASKLET INFO:[TIME:17:35:53]

               state:  2, count: 0, data: my_tasklet_function was called
[24999.126287] INTERRUPT! irq_counter = 5
[24999.126328] TASKLET INFO:[TIME:17:35:53]

               state:  2, count: 0, data: my_tasklet_function was called
[24999.233983] INTERRUPT! irq_counter = 6
[24999.233990] TASKLET INFO:[TIME:17:35:53]
```

```
               state:  2, count: 0, data: my_tasklet_function was called
[24999.329695] INTERRUPT! irq_counter = 10
[24999.329700] TASKLET INFO:[TIME:17:35:53]

               state:  2, count: 0, data: my_tasklet_function was called
[24999.332708] INTERRUPT! irq_counter = 11
[24999.332713] TASKLET INFO:[TIME:17:35:53]

               state:  2, count: 0, data: my_tasklet_function was called
[24999.335765] INTERRUPT! irq_counter = 12
[24999.335770] TASKLET INFO:[TIME:17:35:53]

               state:  2, count: 0, data: my_tasklet_function was called
[24999.338767] INTERRUPT! irq_counter = 13
[24999.338774] TASKLET INFO:[TIME:17:35:53]

               state:  2, count: 0, data: my_tasklet_function was called
[24999.425402] INTERRUPT! irq_counter = 14
[24999.425410] TASKLET INFO:[TIME:17:35:53]

               state:  2, count: 0, data: my_tasklet_function was called
[24999.443984] TASKLET1_MODULE: result irq_counter = 14
[24999.443986] TASKLET1_MODULE unloaded!
```

Задание 2:
· Написать загружаемый модуль ядра, в котором зарегистрировать обработчик аппаратного прерывания с флагом IRQF_SHARED.

· Инициализировать очередь работ.

· В обработчике прерывания запланировать очередь работ на выполнение.

· Вывести информацию об очереди работ используя, или printk(), или seq_file interface - <linux/seq_file.h> (Jonathan Corber: http://lwn.net//Articales//driver-porting/).

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/interrupt.h>
#include <linux/workqueue.h>
#include <linux/time.h>

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("WQ_MODULE");
MODULE_AUTHOR("Moxxx1e");

struct workqueue_struct *wq;
void workqueue_handler(struct work_struct *);
DECLARE_WORK(workname, workqueue_handler);

static struct timespec64 cur_time;

void workqueue_handler(struct work_struct* work) {
    ktime_get_real_ts64(&cur_time);
    int h = cur_time.tv_sec / 3600 % 24;
    int m = cur_time.tv_sec / 60 % 60;
    int s = cur_time.tv_sec % 60;
    printk(KERN_INFO "WORKQUEUE info: "
        "[TIME : %.2d:%.2d:%.2d]\n"
        "data: %d\n",
```

```
        h, m, s, work->data);
}


static const int def_irq = 1;
static int irq_counter = 0;
static irqreturn_t irq_handler(int irq, void* dev_id) {
    if (irq == def_irq) {
        irq_counter++;
        printk(KERN_INFO "Interrupt! irq_counter = %d\n",
irq_counter);
        queue_work(wq, &workname);
        return IRQ_HANDLED; // прерывание обработано
    }

    return IRQ_NONE; // прерывание не обработано
}

static int dev_id;
static int __init wq_module_init(void) {
    if (request_irq(def_irq, irq_handler, IRQF_SHARED,
        "WorkQueue_interrupt", &dev_id)) {
        return -1;
    }

    wq = create_workqueue("wq");
    if (wq) {
        printk(KERN_INFO "WQ_MODULE: Workqueue created\n");
    }

    printk(KERN_INFO "WQ_MODULE loaded\n");
    return 0;
}
```

```
static void __exit wq_module_exit(void) {
    flush_workqueue(wq);
    destroy_workqueue(wq);
    synchronize_irq(def_irq);
    free_irq(def_irq, &dev_id);
    printk(KERN_INFO "WQ_MODULE: result irq_cnt = %d\n", irq_counter);
    printk(KERN_INFO "WQ_MODULE unloaded\n");
}

module_init(wq_module_init);
module_exit(wq_module_exit);
```

Загрузка модуля и проверка установленного обработчика первого прерывания (линия IRQ “разделяется”):

```
oleg@Moxxx1e:~/Документы/BMSTU/OS/lab_09$ sudo insmod wq.ko
oleg@Moxxx1e:~/Документы/BMSTU/OS/lab_09$ cat /proc/interrupts
           CPU0       CPU1       CPU2       CPU3
  0:          5          0          0          0   IO-APIC   2-edge      timer
  1:          0      20860          0          0   IO-APIC   1-edge      i8042, WorkQueue_interrupt
  8:          0          0          1          0   IO-APIC   8-edge      rtc0
  9:          0         56          0          0   IO-APIC   9-fasteoi   acpi
 12:     820381          0          0          0   IO-APIC  12-edge      i8042
```

Демонстрация работы программы на следуюшей странице.

Демонстрация работы программы: