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

Тема **Обработчики прерываний**

Студент Сушина А.Д.

Группа ИУ7-61б

Оценка (баллы) _____

Преподаватель Рязанова Н.Ю,

Москва.
2020 г

Задание на лабораторную работу

Задание 1:

- 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/).

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/).

Задание 1.

Код программы представлен на листинге 1.

Листинг1. tasklet.c

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

MODULE_LICENSE("GPL");

static int my_irq = 1, my_dev_id, irq_cnt = 0;

void tasklet_function(unsigned long data);

char my_tasklet_data[] = "my_tasklet_function was called";

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

void tasklet_function(unsigned long data) {
    printk(KERN_INFO "++ tasklet_function was called. state '%ld' count '%i'
data '%s'\n", my_tasklet.state, my_tasklet.count, (char*)data);
}

static irqreturn_t my_irq_handler(int irq, void *dev) {
    if (irq == my_irq) {
        printk(KERN_INFO "++ my_irq_handler was called %d time(s)\n", +
+irq_cnt);
```

```
        tasklet_schedule(&my_tasklet);
        return IRQ_HANDLED;
    }
    return IRQ_NONE;
}

static int __init my_tasklet_init(void) {
    if (request_irq(my_irq, my_irq_handler, IRQF_SHARED, "my_tasklet",
&my_dev_id)) {
        printk(KERN_ERR "++ can't get assigned IRQ %i\n", my_irq);
        return 1;
    }
    printk(KERN_INFO "++ Successfully loaded handler for IRQ %d\n", my_irq);
    return 0;
}

static void __exit my_tasklet_exit(void) {
    tasklet_kill(&my_tasklet);
    synchronize_irq(my_irq);
    free_irq(my_irq, &my_dev_id);
    printk(KERN_INFO "++ tasklet unloaded, irq_counter = %d\n", irq_cnt);
    return;
}

module_init(my_tasklet_init);

module_exit(my_tasklet_exit);
```

Был написан обработчик прерываний для irq 1 (клавиатура).

Загрузим модуль и проверим журнал. Результаты представлены на рисунке 1.

```
nastya@Nastya:~/iu7/sem6/os/lab9/tasklet$ dmesg | grep ++
[ 1474.707582] ++ Successfully loaded handler for IRQ 1
[ 1474.787224] ++ my_irq_handler was called 1 time(s)
[ 1474.787265] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 1475.689171] ++ my_irq_handler was called 2 time(s)
[ 1475.689203] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 1475.691635] ++ my_irq_handler was called 3 time(s)
[ 1475.691672] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 1475.854367] ++ my_irq_handler was called 4 time(s)
[ 1475.854409] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 1475.856279] ++ my_irq_handler was called 5 time(s)
[ 1475.856295] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 1476.079410] ++ my_irq_handler was called 6 time(s)
[ 1476.079454] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 1476.079708] ++ my_irq_handler was called 7 time(s)
[ 1476.079717] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 1476.247904] ++ my_irq_handler was called 8 time(s)
[ 1476.247911] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 1476.248994] ++ my_irq_handler was called 9 time(s)
[ 1476.249016] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 1476.539339] ++ my_irq_handler was called 10 time(s)
[ 1476.539347] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 1476.632897] ++ my_irq_handler was called 11 time(s)
[ 1476.632929] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 1476.718931] ++ my_irq_handler was called 12 time(s)
[ 1476.718959] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 1476.802121] ++ my_irq_handler was called 13 time(s)
[ 1476.802150] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 1476.870532] ++ my_irq_handler was called 14 time(s)
```

Рис 1. Состояние системного журнала после загрузки модуля

При каждом прерывании от клавиатуры вызывается обработчик прерывания my_irq_handler. В нем вызывается tasklet_schedule(). При обработке тасклета выводится информация о нем.

Проверим, что my_tasklet добавилось в список прерываний. На рисунке 2 представлено содержимое файла /proc/interrupts, что говорит о том, что my_tasklet добавилось в список прерываний.

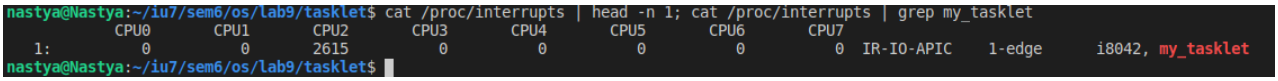

Рис 2. Содержимое файла /proc/interrupts

Выгрузим модуль и проверим содержимое системного журнала. Результаты представлены на рисунке 3.

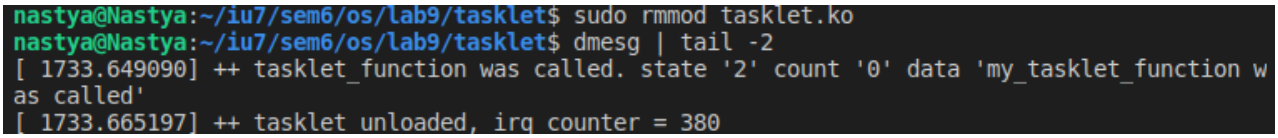

Рис 3. Состояние системного журнала после выгрузки модуля.

Покажем, что модуль удален. На рисунке 4 представлено содержание файла /proc/interrupts, что говорит о том, что my_tasklet удалилось из списка прерывания после выгрузки модуля.

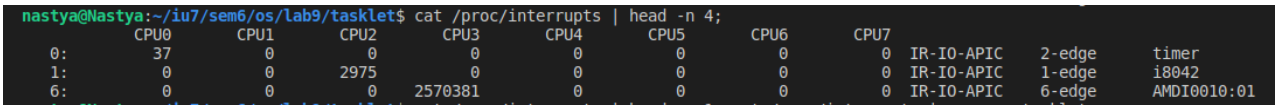

Рис 4. Содержимое файла /proc/interrupts

Задание 2.

Код программы представлен на листинге 2.

Листинг 2. queue.c

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

MODULE_LICENSE("GPL");

static int my_irq = 1, my_dev_id, irq_cnt = 0;

struct workqueue_struct *wq;

static void my_wq_function(struct work_struct *work);

DECLARE_WORK(my_work, my_wq_function);

static void my_wq_function(struct work_struct *work) {
    printk(KERN_INFO "[WORKQUEUE] handler called: data '%d'", work->data);
}

static irqreturn_t my_irq_handler(int irq, void *dev_id) {
    irq_cnt++;
    printk(KERN_INFO "[WORKQUEUE]  my_irq_handler was called %d time(s)",
irq_cnt);
    queue_work(wq, &my_work);
    return IRQ_NONE;

}

static int __init my_workqueue_init(void) {
    if (request_irq(my_irq, my_irq_handler, IRQF_SHARED, "my_int_workqueue",
&my_dev_id)) {
        printk(KERN_ERR "[WORKQUEUE] can't get assigned IRQ %i\n", my_irq);
        return 1;
    }
    printk(KERN_INFO "[WORKQUEUE] assigned IRQ %i\n", my_irq);
    if ((wq = create_workqueue("my_workqueue"))) {
        printk(KERN_INFO "[WORKQUEUE]  workqueue created\n");
    } else {
        free_irq(my_irq, &my_dev_id);
        printk(KERN_INFO "[WORKQUEUE] workqueue allocation failed\n");
        return -ENOMEM;
    }
    printk(KERN_INFO "[WORKQUEUE]  module is now loaded\n");
    return 0;
}

static void __exit my_workqueue_exit(void) {
    flush_workqueue(wq);
    destroy_workqueue(wq);
    printk(KERN_INFO "[WORKQUEUE] workqueue destroyed\n");
    synchronize_irq(my_irq);
    free_irq(my_irq, &my_dev_id);
```

```
    printk(KERN_INFO "[WORKQUEUE]  IRQ handler removed\n");
    printk(KERN_INFO "[WORKQUEUE]  module destroyed\n");
}

module_init(my_workqueue_init);

module_exit(my_workqueue_exit);
```

Был написан обработчик прерываний для irq 1.

Загрузим модуль и проверим журнал. Результаты представлены на рисунке 5.

```
nastya@Nastya:~/iu7/sem6/os/lab9/workqueue$ sudo insmod queue.ko
nastya@Nastya:~/iu7/sem6/os/lab9/workqueue$ dmesg | grep WORKQUEUE
[ 2518.118845] [WORKQUEUE] assigned IRQ 1
[ 2518.118997] [WORKQUEUE]  workqueue created
[ 2518.118999] [WORKQUEUE]  module is now loaded
[ 2518.198013] [WORKQUEUE]  my_irq_handler was called 1 time(s)
[ 2518.198027] [WORKQUEUE] handler called: data '128'
[ 2518.684868] [WORKQUEUE]  my_irq_handler was called 2 time(s)
[ 2518.684887] [WORKQUEUE] handler called: data '128'
[ 2518.685606] [WORKQUEUE]  my_irq_handler was called 3 time(s)
[ 2518.685621] [WORKQUEUE] handler called: data '128'
[ 2518.842587] [WORKQUEUE]  my_irq_handler was called 4 time(s)
[ 2518.842608] [WORKQUEUE] handler called: data '128'
[ 2518.844843] [WORKQUEUE]  my_irq_handler was called 5 time(s)
[ 2518.844848] [WORKQUEUE] handler called: data '128'
[ 2518.986602] [WORKQUEUE]  my_irq_handler was called 6 time(s)
[ 2518.986635] [WORKQUEUE] handler called: data '128'
[ 2518.988107] [WORKQUEUE]  my_irq_handler was called 7 time(s)
[ 2518.988117] [WORKQUEUE] handler called: data '128'
[ 2519.106175] [WORKQUEUE]  my_irq_handler was called 8 time(s)
[ 2519.106218] [WORKQUEUE] handler called: data '128'
[ 2519.108505] [WORKQUEUE]  my_irq_handler was called 9 time(s)
[ 2519.108514] [WORKQUEUE] handler called: data '128'
[ 2519.310290] [WORKQUEUE]  my_irq_handler was called 10 time(s)
[ 2519.310331] [WORKQUEUE] handler called: data '128'
[ 2519.311539] [WORKQUEUE]  my_irq_handler was called 11 time(s)
[ 2519.311548] [WORKQUEUE] handler called: data '128'
[ 2519.442385] [WORKQUEUE]  my_irq_handler was called 12 time(s)
[ 2519.442415] [WORKQUEUE] handler called: data '128'
[ 2519.445927] [WORKQUEUE]  my_irq_handler was called 13 time(s)
```

Рис 5. Состояние системного журнала после загрузки модуля

Проверим, что my_int_worqueue добавилось в список прерываний. На рисунке 6 представлено содержимое файла /proc/interrupts, что говорит о том, что my_int_workueue добавилось в список прерываний.

```
nastya@Nastya:~/iu7/sem6/os/lab9/workqueue$ cat /proc/interrupts | head -n 1; cat /proc/interrupts | grep my_int_workqueue
           CPU0       CPU1       CPU2       CPU3       CPU4       CPU5       CPU6       CPU7
  1:          0          0       7439          0          0          0          0          0  IR-IO-APIC    1-edge      i8042, my_int_workqueue
```

Рис 6. Содержимое файла /proc/interrupts

Выгрузим модуль и проверим содержимое системного журнала. Результаты представлены на рисунке 3.

```
nastya@Nastya:~/iu7/sem6/os/lab9/workqueue$ sudo rmmod queue.ko
nastya@Nastya:~/iu7/sem6/os/lab9/workqueue$ dmesg | tail -4
[ 3168.029646] [WORKQUEUE] handler called: data '128'
[ 3168.042694] [WORKQUEUE] workqueue destroyed
[ 3168.042713] [WORKQUEUE]  IRQ handler removed
[ 3168.042713] [WORKQUEUE]  module destroyed
```

Рис 7. Состояние системного журнала после выгрузки модуля.

Покажем, что модуль удален. На рисунке 4 представлено содержание файла /proc/interrupts, что говорит о том, что my_tasklet удалилось из списка прерывания после выгрузки модуля.

```
nastya@Nastya:~/iu7/sem6/os/lab9/workqueue$ cat /proc/interrupts | head -n 4;
           CPU0       CPU1       CPU2       CPU3       CPU4       CPU5       CPU6       CPU7
  0:         39          0          0          0          0          0          0          0  IR-IO-APIC    2-edge      timer
  1:          0          0          0          0          0          0          0          0  IR-IO-APIC    1-edge      i8042
  6:          0          0          0    4101421          0          0          0          0  IR-IO-APIC    6-edge      AMDI0010:01
```

Рис 8. Содержимое файла /proc/interrupts

Загрузим оба модуля и посмотрим содержимое системного журнала.

```
nastya@Nastya:~/iu7/sem6/os/lab9/tasklet$ dmesg | tail -40
[ 3360.057176] [WORKQUEUE] handler called: data '128'
[ 3360.181563] [WORKQUEUE]  my_irq_handler was called 155 time(s)
[ 3360.181567] ++ my_irq_handler was called 37 time(s)
[ 3360.181589] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 3360.181606] [WORKQUEUE] handler called: data '128'
[ 3360.317288] [WORKQUEUE]  my_irq_handler was called 156 time(s)
[ 3360.317292] ++ my_irq_handler was called 38 time(s)
[ 3360.317309] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 3360.317323] [WORKQUEUE] handler called: data '128'
[ 3360.420165] [WORKQUEUE]  my_irq_handler was called 157 time(s)
[ 3360.420170] ++ my_irq_handler was called 39 time(s)
[ 3360.420184] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 3360.420197] [WORKQUEUE] handler called: data '128'
[ 3360.733473] [WORKQUEUE]  my_irq_handler was called 158 time(s)
[ 3360.733478] ++ my_irq_handler was called 40 time(s)
[ 3360.733497] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 3360.733510] [WORKQUEUE] handler called: data '128'
[ 3360.853872] [WORKQUEUE]  my_irq_handler was called 159 time(s)
[ 3360.853877] ++ my_irq_handler was called 41 time(s)
[ 3360.853900] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 3360.853917] [WORKQUEUE] handler called: data '128'
[ 3360.879658] [WORKQUEUE]  my_irq_handler was called 160 time(s)
[ 3360.879661] ++ my_irq_handler was called 42 time(s)
[ 3360.879676] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 3360.879682] [WORKQUEUE] handler called: data '128'
[ 3361.001001] [WORKQUEUE]  my_irq_handler was called 161 time(s)
[ 3361.001006] ++ my_irq_handler was called 43 time(s)
[ 3361.001025] ++ tasklet_function was called. state '2' count '0' data 'my_tasklet_function was called'
[ 3361.001042] [WORKQUEUE] handler called: data '128'
[ 3361.861991] [WORKQUEUE]  my_irq_handler was called 162 time(s)
```

Рис 9. Состояние системного журнала после выгрузки модуля.

Содержание файла /proc/interrupts:

```
nastya@Nastya:~/iu7/sem6/os/lab9/tasklet$ cat /proc/interrupts | head -n 4;
           CPU0       CPU1       CPU2       CPU3       CPU4       CPU5       CPU6       CPU7
  0:         39          0          0          0          0          0          0          0  IR-IO-APIC    2-edge      timer
  1:          0          0       8636          0          0          0          0          0  IR-IO-APIC    1-edge      i8042, my_int_workqueue, my_tasklet
  6:          0          0          0    4266320          0          0          0          0  IR-IO-APIC    6-edge      AMDI0010:01
```

Рис 10. Содержимое файла /proc/interrupts