|  |  |
| --- | --- |
| Изображение | **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» (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

**ОТЧЕТ**

**по лабораторной работе №** 9

**Название:** Обработчики прерываний.

**Дисциплина:** Операционные системы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-62Б |  |  | Д.М. Блохин |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Н.Ю. Рязанова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва 2021

**Часть 1 (Тасклет)**

* 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 seq\_file interface.

**Листинг программы**

#include <linux/fs.h>

#include <linux/init.h>

#include <linux/kernel.h>

#include <linux/module.h>

#include <linux/proc\_fs.h>

#include <linux/version.h>

#include <linux/seq\_file.h>

#include <linux/interrupt.h>

#define SHARED\_IRQ 1

#define FILE\_NAME "lab9\_1"

MODULE\_LICENSE("GPL");

MODULE\_AUTHOR("Blokhin");

**static** **void** \*lab9\_dev\_id;

**char** tasklet\_info[50];

**static** **int** show(**struct** seq\_file \*m, **void** \*v) {

seq\_printf(m, "Tasklet info:\t%s\n", tasklet\_info);

**return** 0;

}

**static** **int** lab9\_task\_open(**struct** inode \*inode, **struct** file \*file) {

**return** single\_open(file, show, **NULL**);

}

**static** **struct** proc\_ops lab9\_task\_proc\_ops = {

.proc\_open = lab9\_task\_open,

.proc\_release = single\_release,

.proc\_read = seq\_read};

**static** **void** lab9\_task\_handler(**unsigned** **long** data);

DECLARE\_TASKLET(lab9\_task, lab9\_task\_handler, 0);

**static** **void** lab9\_task\_handler(**unsigned** **long** data) {

sprintf(tasklet\_info, "lab9\_task.state=%-4d\tlab9\_task.count=%-"

"4d\tlab9\_task.data=%-4d",

lab9\_task.state, lab9\_task.count, lab9\_task.data);

printk(KERN\_INFO "LAB9 TASKLET:\tTasklet info:\t%s", tasklet\_info);

lab9\_task.data++;

}

**static** irqreturn\_t lab9\_irq\_handler(**int** irq, **void** \*dev\_id) {

**if** (irq == SHARED\_IRQ) {

printk(KERN\_INFO "LAB9 TASKLET:\trun interrupt handler\n");

tasklet\_schedule(&lab9\_task);

**return** IRQ\_HANDLED;

}

**else** **return** IRQ\_NONE;

}

**static** **int** \_\_init lab9\_task\_init(**void**) {

**struct** proc\_dir\_entry \*entry;

entry = proc\_create(FILE\_NAME, S\_IRUGO, **NULL**, &lab9\_task\_proc\_ops);

**if** (!entry) {

printk(KERN\_ERR "LAB9 TASKLET:\tproc\_create() error\n");

**return** -ENOMEM;

}

printk(KERN\_INFO "LAB9 TASKLET:\tcreated proc file\n");

**if** (request\_irq(SHARED\_IRQ, lab9\_irq\_handler, IRQF\_SHARED, "lab9\_task",

&lab9\_dev\_id)) {

printk(KERN\_ERR "LAB9 TASKLET:\trequest\_irq() error\n");

**return** -1;

}

printk(KERN\_INFO "LAB9 TASKLET:\tregistered interrupt handler\n");

printk(KERN\_INFO "LAB9 TASKLET:\tmodule loaded\n");

**return** 0;

}

**static** **void** \_\_exit lab9\_task\_exit(**void**) {

tasklet\_kill(&lab9\_task);

printk(KERN\_INFO "LAB9 TASKLET:\tkilled tasklet\n");

free\_irq(SHARED\_IRQ, &lab9\_dev\_id);

printk(KERN\_INFO "LAB9 TASKLET:\tunregistered interrupt handler\n");

remove\_proc\_entry(FILE\_NAME, **NULL**);

printk(KERN\_INFO "LAB9 TASKLET:\tremoved proc file\n");

printk(KERN\_INFO "LAB9 TASKLET:\tmodule unloaded\n");

}

module\_init(lab9\_task\_init);

module\_exit(lab9\_task\_exit);

**Результат работы программы**

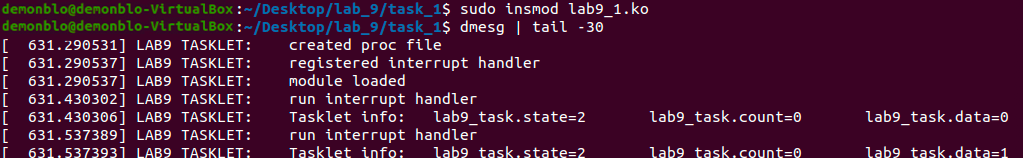
Загрузка модуля

рис. 1 (загрузка модуля)

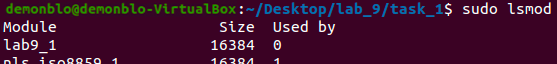


рис. 2 (загрузка модуля)

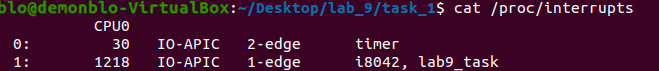


рис. 3 (загрузка модуля)

Вывод в режиме пользователя информации о тасклете (файл /proc/lab9\_1)



рис. 4 (вывод в режиме пользователя информации о тасклете)

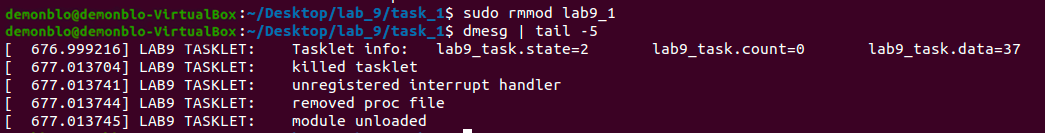
Выгрузка модуля

рис. 5 (выгрузка модуля)

**Часть 2 (Очередь работ)**

Задание

* 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 printk(),
* В очередях работ создаём 2 работы и помещаем в одну очередь (работы блокируются).

**Листинг программы**

#include <linux/init.h>

#include <linux/mutex.h>

#include <linux/module.h>

#include <linux/kernel.h>

#include <linux/interrupt.h>

#include <linux/workqueue.h>

#define SHARED\_IRQ 1

MODULE\_LICENSE("GPL");

MODULE\_AUTHOR("Blokhin");

**static** **int** lab9\_dev\_id, counter = 0;

**struct** workqueue\_struct \*wq;

DEFINE\_MUTEX(lock);

**void** lab9\_work\_function\_1(**struct** work\_struct \*work) {

mutex\_lock(&lock);

printk(KERN\_INFO "lab9 workqueue:\tWorkqueue #1\twork->data=%d\tcounter=%d\n", work->data, counter++);

mutex\_unlock(&lock);

}

**void** lab9\_work\_function\_2(**struct** work\_struct \*work) {

mutex\_lock(&lock);

printk(KERN\_INFO "lab9 workqueue:\tWorkqueue #2\twork->data=%d\tcounter=%d\n", work->data, counter++);

mutex\_unlock(&lock);

}

DECLARE\_WORK(lab9\_work1, lab9\_work\_function\_1);

DECLARE\_WORK(lab9\_work2, lab9\_work\_function\_2);

**static** irqreturn\_t lab9\_irq\_handler(**int** irq, **void** \*dev\_id) {

**if** (irq == SHARED\_IRQ) {

queue\_work(wq, &lab9\_work1);

queue\_work(wq, &lab9\_work2);

printk(KERN\_INFO "lab9 workqueue:\trun interrupt handler\n");

**return** IRQ\_HANDLED;

}

**else**

**return** IRQ\_NONE;

}

**static** **int** \_\_init lab9\_workqueue\_init(**void**) {

**if** (request\_irq(SHARED\_IRQ, lab9\_irq\_handler, IRQF\_SHARED, "lab9\_workqueue", &lab9\_dev\_id)) {

printk(KERN\_ERR "lab9 workqueue:\trequest\_irq() error\n");

**return** 1;

}

printk(KERN\_INFO "lab9 workqueue:\tregistered interrupt handler\n");

wq = create\_workqueue("lab9\_workqueue");

**if** (!wq) {

free\_irq(SHARED\_IRQ, &lab9\_dev\_id);

printk(KERN\_INFO "lab9 workqueue:\tcreate\_workqueue() error\n");

**return** -ENOMEM;

}

printk(KERN\_INFO "lab9 workqueue:\tcreate workqueue\n");

printk(KERN\_INFO "lab9 workqueue:\tmodule loaded\n");

**return** 0;

}

**static** **void** \_\_exit lab9\_workqueue\_exit(**void**) {

flush\_workqueue(wq);

printk(KERN\_INFO "lab9 workqueue:\tflush workqueue\n");

destroy\_workqueue(wq);

printk(KERN\_INFO "lab9 workqueue:\tdestroy workqueue\n");

free\_irq(SHARED\_IRQ, &lab9\_dev\_id);

printk(KERN\_INFO "lab9 workqueue:\tunregistered interrupt handler\n");

printk(KERN\_INFO "lab9 workqueue:\tmodule unloaded\n");

}

module\_init(lab9\_workqueue\_init);

module\_exit(lab9\_workqueue\_exit);

**Результат работы программы**

Загрузка модуля

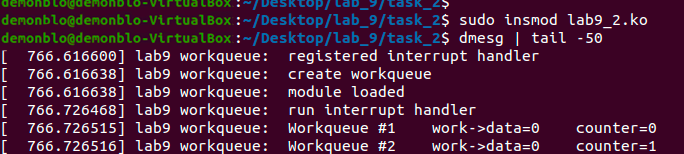


рис. 6 (загрузка модуля)



рис. 7 (загрузка модуля)

Демонстрация разделения линии IRQ

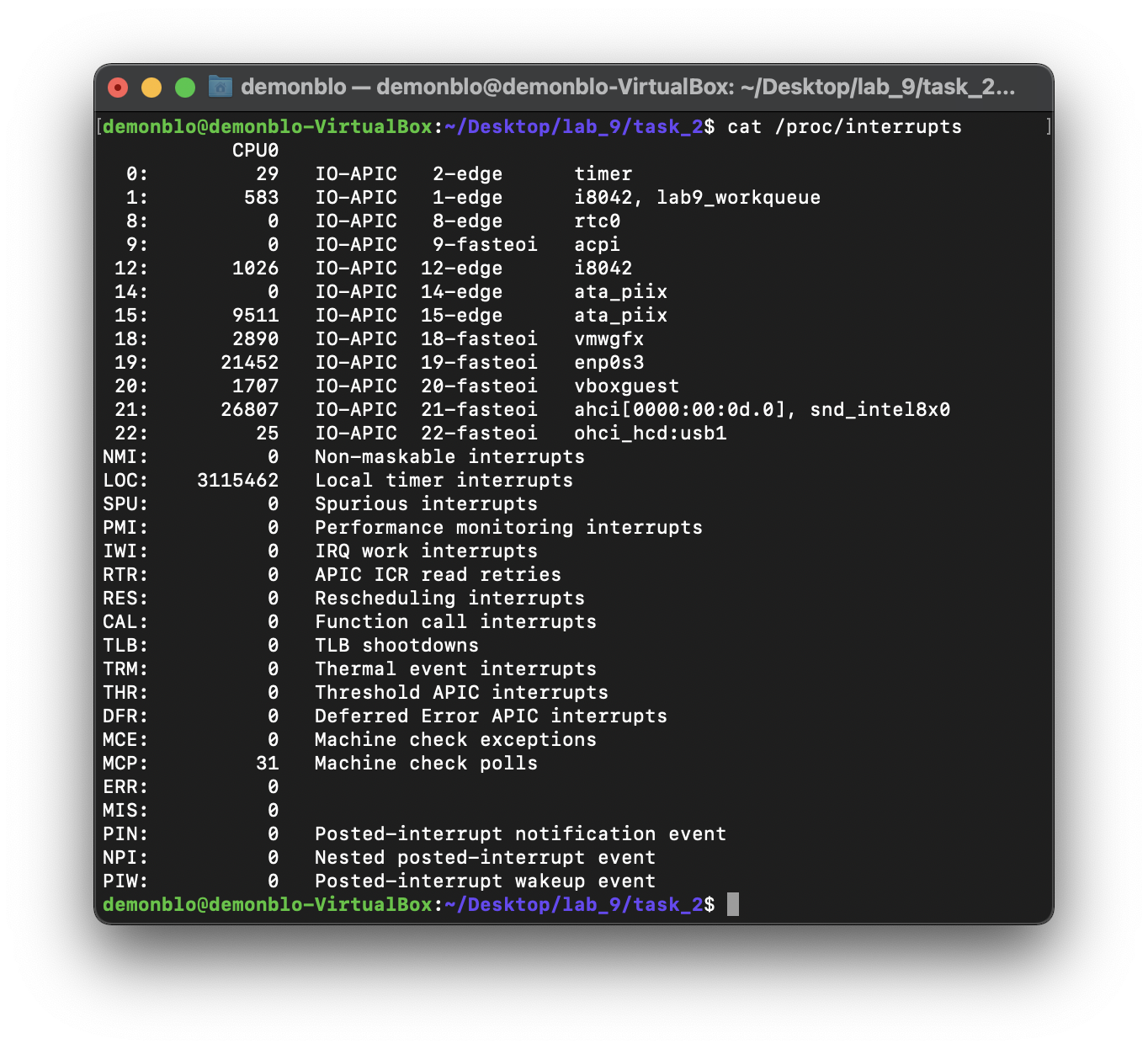


рис. 8 (Демонстрация разделения линии IRQ)

Выгрузка модуля

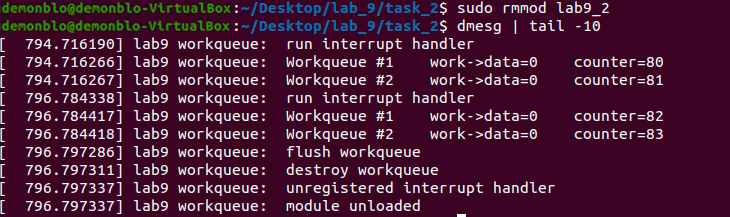


рис. 9 (Выгрузка модуля)