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»

Отчет

к лабораторной работе №9
«Обработчики прерываний»

Студент: Батбилэг Н.

Группа: ИУ7-71Б

Оценка (баллы) _____

Преподаватель: Рязанова Н.Ю.

Москва.
2021 г.

1. 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.

2. Реализация задания 1

В листинге 1 представлен код задания.

Листинг 1. Tasklet.c.

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

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Batbileg Nomuundalai");

static int irq = 1;
static int dev_id, irq_cnt = 0;

char tasklet_data[] = "tasklet_data";
void tasklet_handler(unsigned long data);
DECLARE_TASKLET(tasklet, tasklet_handler, (unsigned long) &tasklet_data);
static struct timespec64 cur_time;

void tasklet_handler(unsigned long data) {
    printk(KERN_INFO "[TASKLET] state: %ld, count: %d, data: %s\n", tasklet.state,
                                                        tasklet.count, tasklet.data);
}

static irqreturn_t test_interrupt(int irq, void *dev_id) {
    if (irq == 1) {
        irq_cnt++;
        printk(KERN_INFO "[interrupt] irq_count = %d\n", irq_cnt);
        tasklet_schedule(&tasklet);
        return IRQ_HANDLED;
    }

    return IRQ_NONE;
}

static int __init test_tasklet_init(void) {
    if (request_irq(irq, test_interrupt, IRQF_SHARED,
        "test_interrupt", &dev_id)) {
        return -1;
    }
    printk(KERN_INFO "[MODULE] Module is loaded\n");
    return 0;
}

static void __exit test_tasklet_exit(void) {
    tasklet_kill(&tasklet);
    synchronize_irq(irq);
    free_irq(irq, &dev_id);
    printk(KERN_INFO "[MODULE] Result irq_count = %d\n", irq_cnt);
```

```
48      printk(KERN_INFO "[MODULE] Module is unloaded\n");
49  }
50
51  module_init(test_tasklet_init);
    module_exit(test_tasklet_exit);
```

В листинге 2 представлено содержимое файла Makefile, необходимого для сборки программы

```
 1  CONFIG_MODULE_SIG=n
 2
 3  ifneq ($(KERNELRELEASE),)
 4          obj-m := tasklet.o
 5
 6  else
 7          CURRENT = $(shell uname -r)
 8          KDIR = /lib/modules/$(CURRENT)/build
 9          PWD = $(shell pwd)
10
11  default:
12          sudo $(MAKE) -C $(KDIR) M=$(PWD) modules
13          sudo make clean
14
15  clean:
16          rm -rf .tmp_versions
17          rm .tasklet.*
18          rm *.o
19          rm *.mod.c
20          rm *.symvers
21          rm *.order
22
23  endif
```

На рисунках 1 - 2 представлена загрузка модуля и проверка установленного обработчика 1-го прерывания.

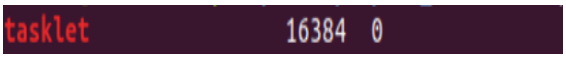

Рисунок 1.

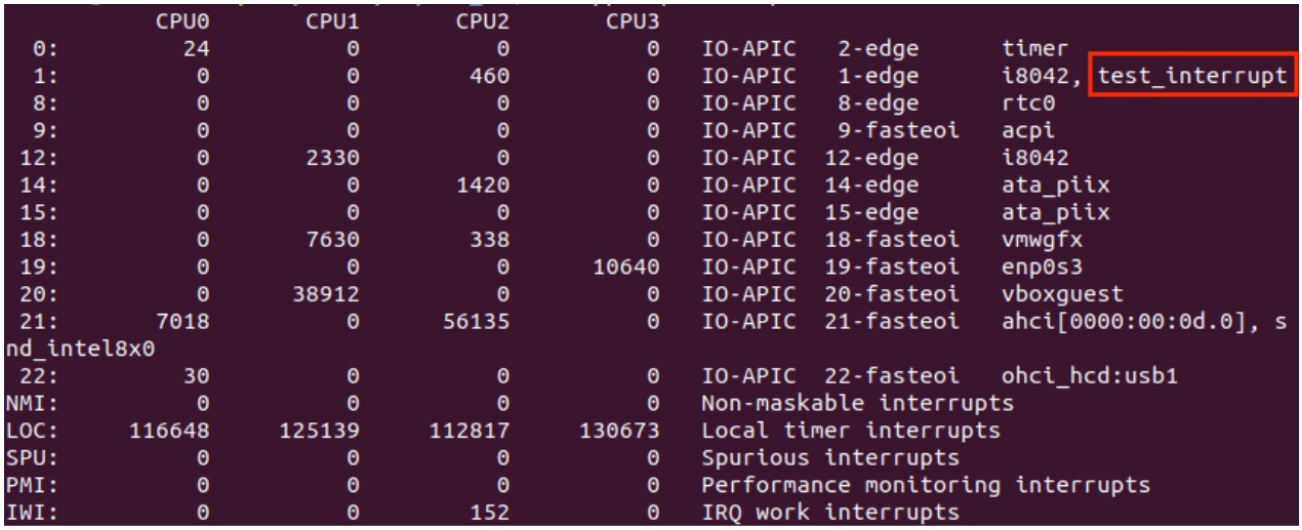

Рисунок 2.

На рисунке 3 представлена информация о работе модуля.

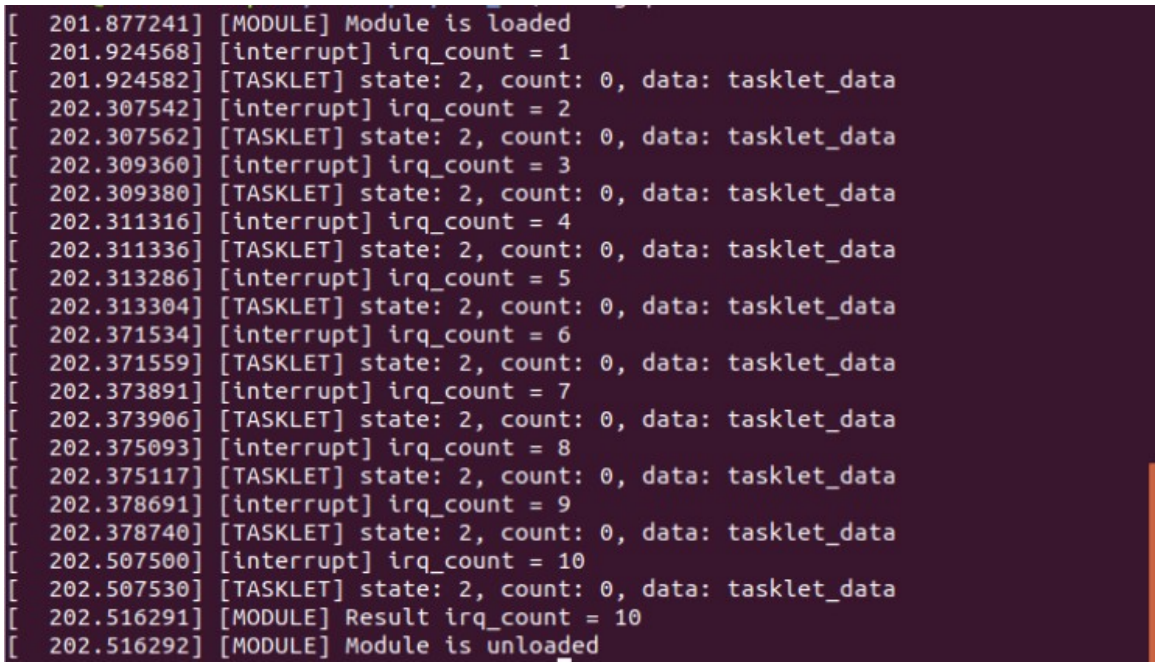

Рисунок 3.

3. 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.

4. Реализация задания 2

В листинге 3 представлен код задания.

Листинг 3. Queue.c.

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


MODULE_LICENSE("GPL");
MODULE_AUTHOR("Batbileg Nomuundalai");

static int irq = 1;
static int dev_id, irq_cnt = 0;

struct workqueue_struct *workqueue_test;
void workqueue_handler(struct work_struct *);

DECLARE_WORK(workname, workqueue_handler);

void workqueue_handler(struct work_struct *work) {
    printk(KERN_INFO "[WORKQUEUE] data: %d\n", work->data);
}

static irqreturn_t test_interrupt(int irq, void *dev_id) {
    if (irq == 1) {
        irq_cnt++;
        printk(KERN_INFO "[interrupt] irq_count = %d\n", irq_cnt);
        queue_work(workqueue_test, &workname);
        return IRQ_HANDLED;
    }

    return IRQ_NONE;
}

static int __init test_workqueue_init(void) {
    if (request_irq(irq, test_interrupt, IRQF_SHARED, "WQ_test_interrupt", &dev_id))
        return -1;

    workqueue_test = create_workqueue("workqueue_test");
    if (workqueue_test)
        printk (KERN_INFO "[MODULE] Workqueue created\n");

    printk(KERN_INFO "[MODULE] Module is loaded\n");
    return 0;
}

static void __exit test_workqueue_exit(void) {
    flush_workqueue(workqueue_test);
```

```
48     destroy_workqueue(workqueue_test);
49     synchronize_irq(irq);
50     free_irq (irq, &dev_id);
51     printk(KERN_INFO "[MODULE] Result irq_count = %d\n", irq_cnt);
52     printk(KERN_INFO "[MODULE] Module is unloaded\n");
53 }
54
55 module_init(test_workqueue_init);
56 module_exit(test_workqueue_exit);
```

В листинге 4 представлено содержимое файла Makefile, необходимого для сборки программы

Листинг 4. Makefile

```
 1 CONFIG_MODULE_SIG=n
 2
 3 ifneq ($(KERNELRELEASE),)
 4         obj-m := queue.o
 5
 6 else
 7         CURRENT = $(shell uname -r)
 8         KDIR = /lib/modules/$(CURRENT)/build
 9         PWD = $(shell pwd)
10
11 default:
12         sudo $(MAKE) -C $(KDIR) M=$(PWD) modules
13         sudo make clean
14
15 clean:
16         rm -rf .tmp_versions
17         rm .tasklet.*
18         rm *.o
19         rm *.mod.c
20         rm *.symvers
21         rm *.order
22
23 endif
```

На рисунках 4 - 5 представлена загрузка модуля и проверка установленного обработчика 1-го прерывания.

Рисунок 4.

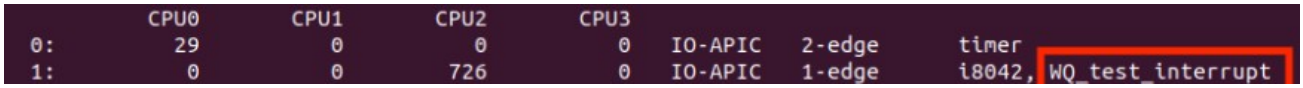

Рисунок 5. Проверка обработчика.

На рисунках 6 - 7 представлена информация о работе модуля.

Рисунок 6. Загрузка и выгрузка модуля.

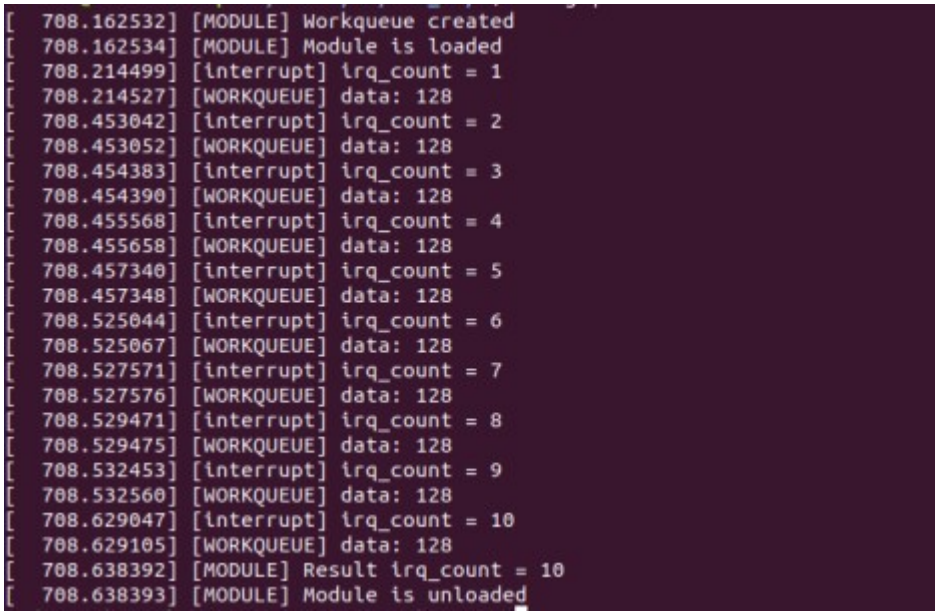

Рисунок 7. Информация о работе модуля.