



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 tasklet.
- В обработчике прерывания запланировать tasklet на выполнение.
- Вывести информацию о tasklete, используя или printk(), или seq_file interface.

2. Реализация задания 1

В листинге 1 представлен код задания.

Листинг 1. Tasklet.c.

```
1 #include <stdio.h>
2 #include <linux/module.h>
3 #include <linux/kernel.h>
4 #include <linux/init.h>
5 #include <linux/interrupt.h>
6
7 MODULE_LICENSE("GPL");
8 MODULE_AUTHOR("Batbileg Nomuundalai");
9
10 static int irq = 1;
11 static int dev_id, irq_cnt = 0;
12
13 char tasklet_data[] = "tasklet_data";
14 void tasklet_handler(unsigned long data);
15 DECLARE_TASKLET(tasklet, tasklet_handler, (unsigned long) &tasklet_data);
16 static struct timespec64 cur_time;
17
18 void tasklet_handler(unsigned long data) {
19     printk(KERN_INFO "[TASKLET] state: %ld, count: %d, data: %s\n", tasklet.state,
20                                     tasklet.count, tasklet.data);
21 }
22
23 static irqreturn_t test_interrupt(int irq, void *dev_id) {
24     if (irq == 1) {
25         irq_cnt++;
26         printk(KERN_INFO "[interrupt] irq_count = %d\n", irq_cnt);
27         tasklet_schedule(&tasklet);
28         return IRQ_HANDLED;
29     }
30
31     return IRQ_NONE;
32 }
33
34 static int __init test_tasklet_init(void) {
35     if (request_irq(irq, test_interrupt, IRQF_SHARED,
36         "test_interrupt", &dev_id)) {
37         return -1;
38     }
39     printk(KERN_INFO "[MODULE] Module is loaded\n");
40     return 0;
41 }
42
43 static void __exit test_tasklet_exit(void) {
44     tasklet_kill(&tasklet);
45     synchronize_irq(irq);
46     free_irq(irq, &dev_id);
47     printk(KERN_INFO "[MODULE] Result irq_count = %d\n", irq_cnt);
```

```

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

```
tasklet          16384  0
```

Рисунок 1.

	CPU0	CPU1	CPU2	CPU3			
0:	24	0	0	0	IO-APIC	2-edge	timer
1:	0	0	460	0	IO-APIC	1-edge	i8042, test_interrupt
8:	0	0	0	0	IO-APIC	8-edge	rtc0
9:	0	0	0	0	IO-APIC	9-fasteoi	acpi
12:	0	2330	0	0	IO-APIC	12-edge	i8042
14:	0	0	1420	0	IO-APIC	14-edge	ata_piix
15:	0	0	0	0	IO-APIC	15-edge	ata_piix
18:	0	7630	338	0	IO-APIC	18-fasteoi	vmwgfx
19:	0	0	0	10640	IO-APIC	19-fasteoi	enp0s3
20:	0	38912	0	0	IO-APIC	20-fasteoi	vboxguest
21:	7018	0	56135	0	IO-APIC	21-fasteoi	ahci[0000:00:0d.0], s
nd_intel8x0							
22:	30	0	0	0	IO-APIC	22-fasteoi	ohci_hcd:usb1
NMI:	0	0	0	0	Non-maskable interrupts		
LOC:	116648	125139	112817	130673	Local timer interrupts		
SPU:	0	0	0	0	Spurious interrupts		
PMI:	0	0	0	0	Performance monitoring interrupts		
IWI:	0	0	152	0	IRQ work interrupts		

Рисунок 2.

На рисунке 3 представлена информация о работе модуля.

```
[ 201.877241] [MODULE] Module is loaded
[ 201.924568] [interrupt] irq_count = 1
[ 201.924582] [TASKLET] state: 2, count: 0, data: tasklet_data
[ 202.307542] [interrupt] irq_count = 2
[ 202.307562] [TASKLET] state: 2, count: 0, data: tasklet_data
[ 202.309360] [interrupt] irq_count = 3
[ 202.309380] [TASKLET] state: 2, count: 0, data: tasklet_data
[ 202.311316] [interrupt] irq_count = 4
[ 202.311336] [TASKLET] state: 2, count: 0, data: tasklet_data
[ 202.313286] [interrupt] irq_count = 5
[ 202.313304] [TASKLET] state: 2, count: 0, data: tasklet_data
[ 202.371534] [interrupt] irq_count = 6
[ 202.371559] [TASKLET] state: 2, count: 0, data: tasklet_data
[ 202.373891] [interrupt] irq_count = 7
[ 202.373906] [TASKLET] state: 2, count: 0, data: tasklet_data
[ 202.375093] [interrupt] irq_count = 8
[ 202.375117] [TASKLET] state: 2, count: 0, data: tasklet_data
[ 202.378691] [interrupt] irq_count = 9
[ 202.378740] [TASKLET] state: 2, count: 0, data: tasklet_data
[ 202.507500] [interrupt] irq_count = 10
[ 202.507530] [TASKLET] state: 2, count: 0, data: tasklet_data
[ 202.516291] [MODULE] Result irq_count = 10
[ 202.516292] [MODULE] Module is unloaded
```

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.

```
1 #include <linux/module.h>
2 #include <linux/kernel.h>
3 #include <linux/init.h>
4 #include <linux/interrupt.h>
5 #include <linux/workqueue.h>
6
7
8 MODULE_LICENSE("GPL");
9 MODULE_AUTHOR("Batbileg Nomuundalai");
10
11 static int irq = 1;
12 static int dev_id, irq_cnt = 0;
13
14 struct workqueue_struct *workqueue_test;
15 void workqueue_handler(struct work_struct *);
16
17 DECLARE_WORK(workname, workqueue_handler);
18
19 void workqueue_handler(struct work_struct *work) {
20     printk(KERN_INFO "[WORKQUEUE] data: %d\n", work->data);
21 }
22
23 static irqreturn_t test_interrupt(int irq, void *dev_id) {
24     if (irq == 1) {
25         irq_cnt++;
26         printk(KERN_INFO "[interrupt] irq_count = %d\n", irq_cnt);
27         queue_work(workqueue_test, &workname);
28         return IRQ_HANDLED;
29     }
30
31     return IRQ_NONE;
32 }
33
34 static int __init test_workqueue_init(void) {
35     if (request_irq(irq, test_interrupt, IRQF_SHARED, "WQ_test_interrupt", &dev_id))
36         return -1;
37
38     workqueue_test = create_workqueue("workqueue_test");
39     if (workqueue_test)
40         printk(KERN_INFO "[MODULE] Workqueue created\n");
41
42     printk(KERN_INFO "[MODULE] Module is loaded\n");
43     return 0;
44 }
45
46 static void __exit test_workqueue_exit(void) {
47     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.

```
queue 16384 0
```

Рисунок 4.

	CPU0	CPU1	CPU2	CPU3			
0:	29	0	0	0	IO-APIC	2-edge	timer
1:	0	0	726	0	IO-APIC	1-edge	i8042, WQ_test interrupt

Рисунок 5. Проверка обработчика.

На рисунках 6 - 7 представлена информация о работе модуля.

```
sudo insmod queue.ko  
sudo rmmod queue
```

Рисунок 6. Загрузка и выгрузка модуля.

```
[ 708.162532] [MODULE] Workqueue created  
[ 708.162534] [MODULE] Module is loaded  
[ 708.214499] [interrupt] irq_count = 1  
[ 708.214527] [WORKQUEUE] data: 128  
[ 708.453042] [interrupt] irq_count = 2  
[ 708.453052] [WORKQUEUE] data: 128  
[ 708.454383] [interrupt] irq_count = 3  
[ 708.454390] [WORKQUEUE] data: 128  
[ 708.455568] [interrupt] irq_count = 4  
[ 708.455568] [WORKQUEUE] data: 128  
[ 708.457340] [interrupt] irq_count = 5  
[ 708.457348] [WORKQUEUE] data: 128  
[ 708.525044] [interrupt] irq_count = 6  
[ 708.525067] [WORKQUEUE] data: 128  
[ 708.527571] [interrupt] irq_count = 7  
[ 708.527576] [WORKQUEUE] data: 128  
[ 708.529471] [interrupt] irq_count = 8  
[ 708.529475] [WORKQUEUE] data: 128  
[ 708.532453] [interrupt] irq_count = 9  
[ 708.532560] [WORKQUEUE] data: 128  
[ 708.629047] [interrupt] irq_count = 10  
[ 708.629105] [WORKQUEUE] data: 128  
[ 708.638392] [MODULE] Result irq_count = 10  
[ 708.638393] [MODULE] Module is unloaded
```

Рисунок 7. Информация о работе модуля.