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

Дисциплина Операционные системы

Тема Обработчики прерываний

Студент Ильясов И. М.

Группа ИУ7-63Б

Оценка (баллы) _____

Преподаватель Рязанова Н. Ю.

Москва, 2020 г.

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.

Пример работы программы

На приведенных ниже рисунках продемонстрирована работа программы из лабораторной работы.

На рисунке 1 приведен результат сборки загружаемого модуля ядра my_tasklet при помощи утилиты make.

```
parallels@parallels-Parallels-Virtual-Platform:/media/psf/Home/University/sem_06/OS/lab_09/tasklet$ make
make -C /lib/modules/4.15.0-34-generic/build M=/media/psf/Home/University/sem_06/OS/lab_09/tasklet modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-34-generic'
  CC [M]  /media/psf/Home/University/sem_06/OS/lab_09/tasklet/my_tasklet.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /media/psf/Home/University/sem_06/OS/lab_09/tasklet/my_tasklet.mod.o
  LD [M]  /media/psf/Home/University/sem_06/OS/lab_09/tasklet/my_tasklet.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-34-generic'
```

Рисунок 1 – сборка загружаемого модуля ядра my_tasklet при помощи make

На рисунке 2 приведена команда загрузки модуля и демонстрация успешности этой загрузки.

```
parallels@parallels-Parallels-Virtual-Platform:/media/psf/Home/University/sem_06/OS/lab_09/tasklet$ sudo insmod my_tasklet.ko
parallels@parallels-Parallels-Virtual-Platform:/media/psf/Home/University/sem_06/OS/lab_09/tasklet$ lsmod | grep my_tasklet
my_tasklet             16384  0
```

Рисунок 2 – загрузка модуля my_tasklet

Далее выгрузим модуль и просмотрим сообщения, записанные в логи.

```
parallels@parallels-Parallels-Virtual-Platform:/media/psf/Home/University/sem_06/OS/lab_09/tasklet$ sudo rmmod my_tasklet
parallels@parallels-Parallels-Virtual-Platform:/media/psf/Home/University/sem_06/OS/lab_09/tasklet$ dmesg
```

Рисунок 3 – выгрузка модуля my_tasklet

```
[ 3096.547839] Successfully loading ISR handler on IRQ 1
[ 3096.547839] Module is now loaded!
[ 3096.634254] + TASKLET --- TIME: 22:09:09
[ 3099.441149] + TASKLET --- TIME: 22:09:12
[ 3099.552901] + TASKLET --- TIME: 22:09:12
[ 3099.568770] + TASKLET --- TIME: 22:09:12
[ 3099.680851] + TASKLET --- TIME: 22:09:12
[ 3099.730717] + TASKLET --- TIME: 22:09:12
[ 3099.823079] + TASKLET --- TIME: 22:09:12
[ 3099.892789] + TASKLET --- TIME: 22:09:12
[ 3099.977279] + TASKLET --- TIME: 22:09:12
[ 3100.022613] + TASKLET --- TIME: 22:09:12
[ 3100.135111] + TASKLET --- TIME: 22:09:12
[ 3100.395817] + TASKLET --- TIME: 22:09:13
[ 3100.479469] + TASKLET --- TIME: 22:09:13
[ 3100.491524] + TASKLET --- TIME: 22:09:13
[ 3100.563629] + TASKLET --- TIME: 22:09:13
[ 3100.633135] + TASKLET --- TIME: 22:09:13
[ 3100.725230] + TASKLET --- TIME: 22:09:13
[ 3100.901199] + TASKLET --- TIME: 22:09:13
[ 3100.988843] + TASKLET --- TIME: 22:09:13
[ 3101.027597] + TASKLET --- TIME: 22:09:13
[ 3101.128070] + TASKLET --- TIME: 22:09:13
[ 3103.139478] + TASKLET --- TIME: 22:09:15
[ 3103.261952] + TASKLET --- TIME: 22:09:15
[ 3104.159887] + TASKLET --- TIME: 22:09:16
[ 3104.253821] + TASKLET --- TIME: 22:09:16
[ 3104.633260] + TASKLET --- TIME: 22:09:17
[ 3104.744713] + TASKLET --- TIME: 22:09:17
[ 3104.770721] + TASKLET --- TIME: 22:09:17
[ 3104.875360] + TASKLET --- TIME: 22:09:17
[ 3106.212689] + TASKLET --- TIME: 22:09:18
[ 3106.221014] Successfully unloading, irq_cnt = 30
[ 3106.221015] Module is now unloaded!
```

Рисунок 4 – просмотр логов

Также покажем, что линия IRQ разделяется при помощи команды cat /proc/interrupts.

```
parallels@parallels-Parallels-Virtual-Platform:/media/psf/Home/University/sem_06/OS/lab_09/tasklet$ cat /proc/interrupts
           CPU0       CPU1
  0:          2          0   IO-APIC   2-edge      timer
  1:          0       9915   IO-APIC   1-edge      i8042, my_interrupt
  8:          1          0   IO-APIC   8-edge      rtc0
  9:          0      42928   IO-APIC   9-fasteoi   acpi
 12:        143          0   IO-APIC  12-edge      i8042
 14:          0          0   IO-APIC  14-edge      ata_piix
 15:          0          0   IO-APIC  15-edge      ata_piix
 17:          0       7414   IO-APIC  17-fasteoi   snd_intel8x0
 18:      21169      22252   IO-APIC  18-fasteoi   uhci_hcd:usb2
 19:        175         35   IO-APIC  19-fasteoi   ehci_hcd:usb1
 22:          0          0   IO-APIC  22-fasteoi   virtio1
 24:        103        324   PCI-MSI 487424-edge      xhci_hcd
 25:          0      45050   PCI-MSI 49152-edge      prl_tg
 26:          3          9   PCI-MSI 81920-edge      virtio0-config
 27:        383      50349   PCI-MSI 81921-edge      virtio0-input.0
 28:          5          5   PCI-MSI 81922-edge      virtio0-output.0
 29:     334847      53642   PCI-MSI 512000-edge      ahci[0000:00:1f.2]
 30:       5472      87956   PCI-MSI 524288-edge      prl_drm
NMI:          0          0   Non-maskable interrupts
LOC:     890266     630530   Local timer interrupts
SPU:          0          0   Spurious interrupts
PMI:          0          0   Performance monitoring interrupts
IWI:          0        513   IRQ work interrupts
RTR:          0          0   APIC ICR read retries
RES:    1413418    1417962   Rescheduling interrupts
CAL:      46252      41090   Function call interrupts
TLB:      16558      10140   TLB shootdowns
TRM:          0          0   Thermal event interrupts
THR:          0          0   Threshold APIC interrupts
DFR:          0          0   Deferred Error APIC interrupts
MCE:          0          0   Machine check exceptions
MCP:         36         36   Machine check polls
HYP:          0          0   Hypervisor callback interrupts
ERR:          0
MIS:          0
PIN:          0          0   Posted-interrupt notification event
NPI:          0          0   Nested posted-interrupt event
PIW:          0          0   Posted-interrupt wakeup event
```

Рисунок 5 – разделение линии IRQ

Листинг программы

Ниже в листингах приведен код программы и содержимое Makefile. Так, в листинге 1 показано содержимое файла Makefile.

Листинг 1 – содержимое Makefile

```
ifneq ($(KERNELRELEASE),)
	obj-m := my_tasklet.o
else
	CURRENT = $(shell uname -r)
	KDIR = /lib/modules/$(CURRENT)/build
	PWD = $(shell pwd)
default:
	$(MAKE) -C $(KDIR) M=$(PWD) modules
clean:
	rm -rf .tmp_versions
	rm *.ko
	rm *.o
	rm *.mod.c
	rm *.symvers
	rm *.order
endif
```

В листинге 2 приведено содержимое файла my_tasklet.c.

Листинг 2 – содержимое my_tasklet.c

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


MODULE_LICENSE("GPL");
MODULE_AUTHOR("Ilyasov Idris ICS7-63B");
MODULE_DESCRIPTION("Tasklet");


static int irq_cnt = 0, irq = 1;
module_param(irq, int, S_IRUGO);
char tasklet_data[] = "tasklet func";
void tasklet_function(unsigned long data);
static struct timeval time;


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


void tasklet_function(unsigned long data)
{
    do_gettimeofday(&time);
    printk(KERN_INFO "+ TASKLET --- TIME: %.2lu:%.2lu:%.2lu\n",
                                            (time.tv_sec / 3600) % (24),
                                            (time.tv_sec / 60) % (60),
                                            time.tv_sec % 60);

    return;
}
```

```
static irqreturn_t my_interrupt(int irq, void *dev_id)
{
    if (irq == 1)
    {
        irq_cnt++;
        tasklet_schedule(&my_tasklet);
        return IRQ_HANDLED;
    }
    else
    {
        return IRQ_NONE;
    }
}

static int __init my_tasklet_init(void)
{
    if (request_irq(irq, my_interrupt, IRQF_SHARED, "my_interrupt",
                                                        (void*)my_interrupt))
    {
        return -1;
    }

    printk("Successfully loading ISR handler on IRQ %d\n", irq);
    printk("Module is now loaded!\n");
    return 0;
}

static void __exit my_tasklet_exit(void)
{
    tasklet_kill(&my_tasklet);
    free_irq(irq, (void*)my_interrupt);
    printk("Successfully unloading, irq_cnt = %d\n", irq_cnt);
    printk("Module is now unloaded!\n");
    return;
}

module_init(my_tasklet_init);
module_exit(my_tasklet_exit);
```

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.

Пример работы программы

На приведенных ниже рисунках продемонстрирована работа программы из лабораторной работы.

На рисунке 6 приведен результат сборки загружаемого модуля ядра my_workqueue при помощи утилиты make.

```
parallels@parallels-Parallels-Virtual-Platform:/media/psf/Home/University/sem_06/OS/lab_09/workqueue$ make
make -C /lib/modules/4.15.0-34-generic/build M=/media/psf/Home/University/sem_06/OS/lab_09/workqueue modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-34-generic'
  CC [M]  /media/psf/Home/University/sem_06/OS/lab_09/workqueue/my_workqueue.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /media/psf/Home/University/sem_06/OS/lab_09/workqueue/my_workqueue.mod.o
  LD [M]  /media/psf/Home/University/sem_06/OS/lab_09/workqueue/my_workqueue.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-34-generic'
```

Рисунок 6 – сборка загружаемого модуля ядра my_workqueue при помощи make

На рисунке 7 приведена команда загрузки модуля и демонстрация успешности этой загрузки.

```
parallels@parallels-Parallels-Virtual-Platform:/media/psf/Home/University/sem_06/OS/lab_09/workqueue$ sudo insmod my_workqueue.ko
parallels@parallels-Parallels-Virtual-Platform:/media/psf/Home/University/sem_06/OS/lab_09/workqueue$ lsmod | grep my_workqueue
my_workqueue           16384  0
```

Рисунок 7 – загрузка модуля my_workqueue

Далее выгрузим модуль и просмотрим сообщения, записанные в логи.

```
parallels@parallels-Parallels-Virtual-Platform:/media/psf/Home/University/sem_06/OS/lab_09/workqueue$ sudo rmmod my_workqueue
parallels@parallels-Parallels-Virtual-Platform:/media/psf/Home/University/sem_06/OS/lab_09/workqueue$ dmesg
[ 4436.519438] Successfully loading ISR handler on IRQ 1
[ 4436.521203] Workqueue created!
[ 4436.521203] Module is now loaded!
[ 4436.603183] + WORKQUEUE --- TIME: 22:31:29
[ 4438.611293] + WORKQUEUE --- TIME: 22:31:31
[ 4438.711884] + WORKQUEUE --- TIME: 22:31:31
[ 4438.718520] + WORKQUEUE --- TIME: 22:31:31
[ 4438.819475] + WORKQUEUE --- TIME: 22:31:31
[ 4438.855016] + WORKQUEUE --- TIME: 22:31:31
[ 4438.955492] + WORKQUEUE --- TIME: 22:31:31
[ 4438.955947] + WORKQUEUE --- TIME: 22:31:31
[ 4439.051697] + WORKQUEUE --- TIME: 22:31:31
[ 4439.095388] + WORKQUEUE --- TIME: 22:31:31
[ 4439.191387] + WORKQUEUE --- TIME: 22:31:31
[ 4439.351210] + WORKQUEUE --- TIME: 22:31:32
[ 4439.427085] + WORKQUEUE --- TIME: 22:31:32
[ 4439.439073] + WORKQUEUE --- TIME: 22:31:32
[ 4439.507190] + WORKQUEUE --- TIME: 22:31:32
[ 4439.572339] + WORKQUEUE --- TIME: 22:31:32
[ 4439.664359] + WORKQUEUE --- TIME: 22:31:32
[ 4440.156970] + WORKQUEUE --- TIME: 22:31:32
[ 4440.249437] + WORKQUEUE --- TIME: 22:31:32
[ 4440.291493] + WORKQUEUE --- TIME: 22:31:32
[ 4440.403221] + WORKQUEUE --- TIME: 22:31:33
[ 4440.426838] + WORKQUEUE --- TIME: 22:31:33
[ 4440.476380] + WORKQUEUE --- TIME: 22:31:33
[ 4440.568035] + WORKQUEUE --- TIME: 22:31:33
[ 4440.673844] + WORKQUEUE --- TIME: 22:31:33
[ 4440.707387] + WORKQUEUE --- TIME: 22:31:33
[ 4440.800927] + WORKQUEUE --- TIME: 22:31:33
[ 4440.859334] + WORKQUEUE --- TIME: 22:31:33
[ 4440.963326] + WORKQUEUE --- TIME: 22:31:33
[ 4441.352096] + WORKQUEUE --- TIME: 22:31:34
[ 4441.359066] Successfully unloading, irq_cnt = 30
[ 4441.359067] Module is now unloaded!
```

Рисунок 8 – выгрузка модуля my_workqueue и просмотр логов

Также покажем, что линия IRQ разделяется при помощи команды cat /proc/interrupts.

```
parallels@parallels-Parallels-Virtual-Platform:/media/psf/Home/University/sem_06/OS/lab_09/workqueue$ cat /proc/interrupts
           CPU0       CPU1
  0:          1          0   IO-APIC   2-edge      timer
  1:          0       1637   IO-APIC   1-edge      i8042, my_interrupt
  8:          1          0   IO-APIC   8-edge      rtc0
  9:          0       5023   IO-APIC   9-fasteoi   acpi
 12:        143          0   IO-APIC  12-edge      i8042
 14:          0          0   IO-APIC  14-edge      ata_piix
 15:          0          0   IO-APIC  15-edge      ata_piix
 17:          0       3130   IO-APIC  17-fasteoi   snd_intel8x0
 18:         96       7715   IO-APIC  18-fasteoi   uhci_hcd:usb2
 19:          0         34   IO-APIC  19-fasteoi   ehci_hcd:usb1
 22:          0          0   IO-APIC  22-fasteoi   virtio1
 24:          0         30   PCI-MSI 487424-edge      xhci_hcd
 25:          0      10811   PCI-MSI 49152-edge      prl_tg
 26:          0          0   PCI-MSI 81920-edge      virtio0-config
 27:       5709       5890   PCI-MSI 81921-edge      virtio0-input.0
 28:          1          0   PCI-MSI 81922-edge      virtio0-output.0
 29:       5428      59857   PCI-MSI 512000-edge      ahci[0000:00:1f.2]
 30:      21072          0   PCI-MSI 524288-edge      prl_drm
NMI:          0          0   Non-maskable interrupts
LOC:      92140     105928   Local timer interrupts
SPU:          0          0   Spurious interrupts
PMI:          0          0   Performance monitoring interrupts
IWI:          0          0   IRQ work interrupts
RTR:          0          0   APIC ICR read retries
RES:     253218     224602   Rescheduling interrupts
CAL:       3691       2546   Function call interrupts
TLB:       2228       1361   TLB shootdowns
TRM:          0          0   Thermal event interrupts
THR:          0          0   Threshold APIC interrupts
DFR:          0          0   Deferred Error APIC interrupts
MCE:          0          0   Machine check exceptions
MCP:          4          4   Machine check polls
HYP:          0          0   Hypervisor callback interrupts
ERR:          0
MIS:          0
PIN:          0          0   Posted-interrupt notification event
NPI:          0          0   Nested posted-interrupt event
PIW:          0          0   Posted-interrupt wakeup event
```

Рисунок 9 – разделение линии IRQ

Листинг программы

Ниже в листингах приведен код программы и содержимое Makefile. Так, в листинге 3 показано содержимое файла Makefile.

Листинг 3 – содержимое Makefile

```
ifneq ($(KERNELRELEASE),)
	obj-m := my_workqueue.o
else
	CURRENT = $(shell uname -r)
	KDIR = /lib/modules/$(CURRENT)/build
	PWD = $(shell pwd)
default:
	$(MAKE) -C $(KDIR) M=$(PWD) modules
clean:
	rm -rf .tmp_versions
	rm *.ko
	rm *.o
	rm *.mod.c
	rm *.symvers
	rm *.order
endif
```

В листинге 4 приведено содержимое файла my_workqueue.c.

Листинг 4 – содержимое my_workqueue.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("Ilyasov Idris ICS7-63B");
MODULE_DESCRIPTION("Workqueue");

static int irq_cnt = 0, irq = 1;
module_param(irq, int, S_IRUGO);
struct workqueue_struct *wq;
void hardwork_function(struct work_struct *work);
static struct timeval time;

DECLARE_WORK(hardwork, hardwork_function);

void hardwork_function(struct work_struct *work)
{
    do_gettimeofday(&time);
    printk(KERN_INFO "+ TASKLET --- TIME: %.2lu:%.2lu:%.2lu\n",
                                                (time.tv_sec / 3600) % (24),
                                                (time.tv_sec / 60) % (60),
                                                time.tv_sec % 60);

    return;
}

static irqreturn_t my_interrupt(int irq, void *dev_id)
{
    if (irq == 1)
    {
        irq_cnt++;
        queue_work(wq, &hardwork);
        return IRQ_HANDLED;
    }
    else
    {
        return IRQ_NONE;
    }
}

static int __init my_workqueue_init(void)
{
    if (request_irq(irq, my_interrupt, IRQF_SHARED, "my_interrupt",
                                                        (void*)my_interrupt))
    {
        return -1;
    }

    printk(KERN_INFO "Successfully loading ISR handler on IRQ %d\n", irq);
    wq = create_workqueue("workqueue");

    if (wq)
    {
        printk(KERN_INFO "Workqueue created!\n");
    }

    printk(KERN_INFO "Module is now loaded!\n");
    return 0;
}

static void __exit my_workqueue_exit(void)
{
    flush_workqueue(wq);
```

```
    destroy_workqueue(wq);
    free_irq(irq, (void*)my_interrupt);
    printk("Successfully unloading, irq_cnt = %d\n", irq_cnt);
    printk("Module is now unloaded!\n");
    return;
}

module_init(my_workqueue_init);
module_exit(my_workqueue_exit);
```